Question 1:

You are a software engineer working for an IT company and are asked to contribute to a growing internal application that includes dashboards for data visualization. You are provisioning your AWS DynamoDB table and need to perform 10 strongly consistent reads per second of 4 KB in size each. How many Read Capacity Units (RCUs) are needed?

- 5
- 40
- 10(Correct)
- 20

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.

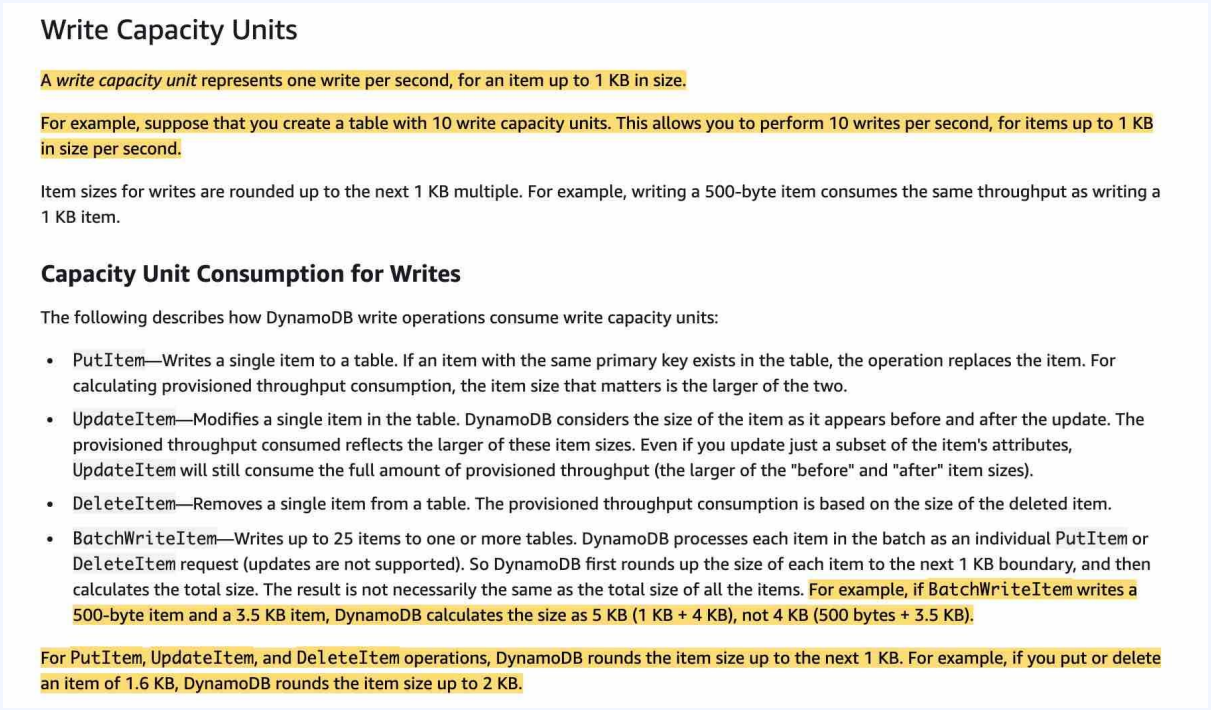
Item sizes for reads are rounded up to the next 4 KB multiple. For example, reading a 3,500-byte item consumes the same throughput as reading a 4 KB item.

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

10

One 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. If you need to read an item that is larger than 4 KB, DynamoDB will need to consume additional read capacity units. The total number of Read Capacity Units required depends on the item size, and whether you want an eventually consistent or strongly consistent read.

1) Item Size / 4KB, rounding to the nearest whole number.

So, in the above case, 4KB / 4 KB = 1 read capacity unit.

2) 1 read capacity unit per item (since strongly consistent read) × No of reads per second

So, in the above case, 1 x 10 = 10 read capacity units.

Incorrect options:

40

20

5

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

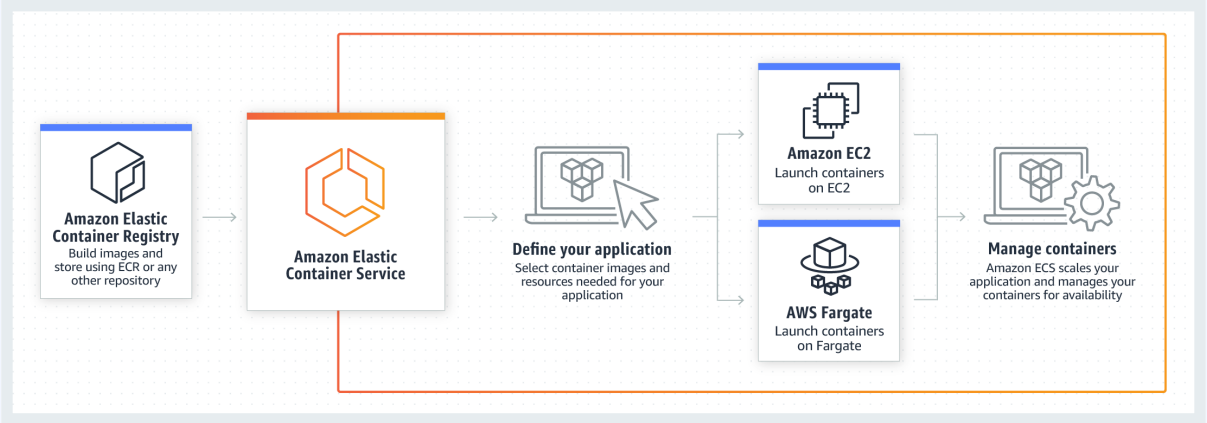
Reference:

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

Question 2:
A company that specializes in cloud communications platform as a service allows software developers to programmatically use their services to send and receive text messages. The initial platform did not have a scalable architecture as all components were hosted on one server and should be redesigned for high availability and scalability.
Which of the following options can be used to implement the new architecture? (select two)

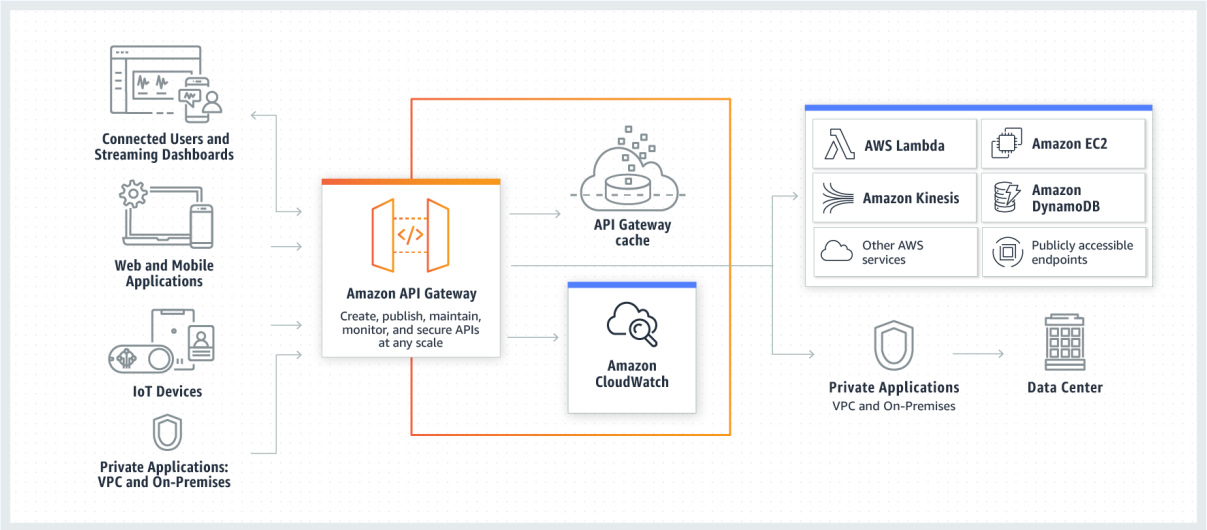
- ALB + ECS(Correct)
- CloudWatch + CloudFront
- EBS + RDS
- API Gateway + Lambda(Correct)
- SES + S3

Explanation
Correct options:
ALB + ECS
Amazon Elastic Container Service (ECS) is a highly scalable, high-performance container management service that supports Docker containers and allows you to easily run applications on a managed cluster of Amazon EC2 instances.
How ECS Works:



via - https://aws.amazon.com/ecs/
Elastic Load Balancing automatically distributes incoming application traffic across multiple targets, such as Amazon EC2 instances, containers, IP addresses, and Lambda functions. It can handle the varying load of your application traffic in a single Availability Zone or across multiple Availability Zones.
When you use ECS with a load balancer such as ALB deployed across multiple Availability Zones, it helps provide a scalable and highly available REST API.
API Gateway + Lambda
Amazon API Gateway is a fully managed service that makes it easy for developers to publish, maintain, monitor, and secure APIs at any scale. Using API Gateway, you can create an API that acts as a “front door” for applications to access data, business logic, or functionality from your back-end services, such as EC2 or Lambda functions.

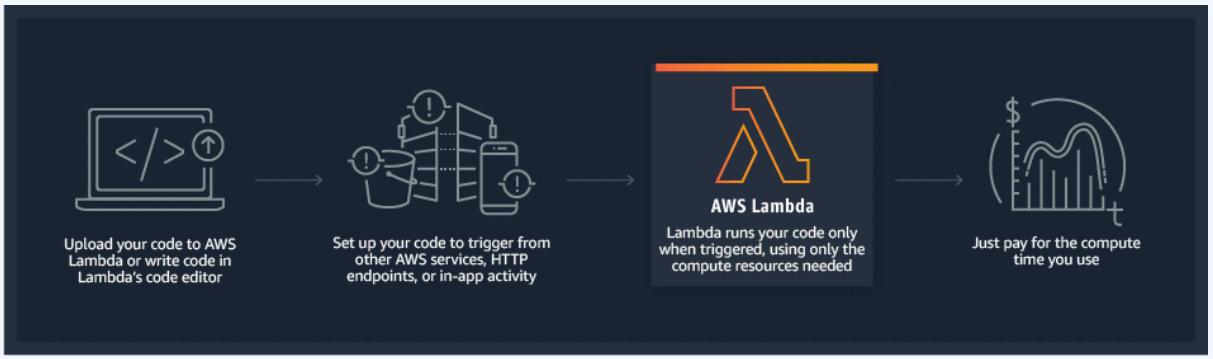
How API Gateway Works:



via - https://aws.amazon.com/api-gateway/

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/

API Gateway and Lambda help achieve the same purpose integrating some capabilities such as authentication in a serverless fashion, with fully scalable and highly available architectures.

Incorrect options:

SES + S3 - The combination of these services only provide email and object storage services.

CloudWatch + CloudFront - The combination of these services only provide monitoring and fast content delivery network (CDN) services.

EBS + RDS - The combination of these services only provide elastic block storage and database services.

References:

https://aws.amazon.com/getting-started/projects/build-serverless-web-app-lambda-apigateway-s3-dynamodb-cognito/module-4/

https://aws.amazon.com/blogs/compute/microservice-delivery-with-amazon-ecs-and-application-load-balancers/

Question 3:
A developer is creating a RESTful API service using an Amazon API Gateway with AWS Lambda integration. The service must support different API versions for testing purposes.
As a Developer Associate, which of the following would you suggest as the best way to accomplish this?

- Use an API Gateway Lambda authorizer to route API clients to the correct API version
- Use an X-Version header to identify which version is being called and pass that header to the Lambda function
- Deploy the API versions as unique stages with unique endpoints and use stage variables to provide the context to identify the API versions(Correct)
- Set up an API Gateway resource policy to identify the API versions and provide context to the Lambda function

Explanation
Correct option:
Deploy the API versions as unique stages with unique endpoints and use stage variables to provide the context to identify the API versions - A stage is a named reference to a deployment, which is a snapshot of the API. You use a stage to manage and optimize a particular deployment. For example, you can configure stage settings to enable caching, customize request throttling, configure logging, define stage variables, or attach a canary release for testing.
Stage variables are name-value pairs that you can define as configuration attributes associated with a deployment stage of a REST API. They act like environment variables and can be used in your API setup and mapping templates.
With deployment stages in API Gateway, you can manage multiple release stages for each API, such as alpha, beta, and production. Using stage variables you can configure an API deployment stage to interact with different backend endpoints.
For example, your API can pass a GET request as an HTTP proxy to the backend web host (for example, http://example.com). In this case, the backend web host is configured in a stage variable so that when developers call your production endpoint, API Gateway calls example.com. When you call your beta endpoint, API Gateway uses the value configured in the stage variable for the beta stage, and calls a different web host (for example, beta.example.com). Similarly, stage variables can be used to specify a different AWS Lambda function name for each stage in your API.
Incorrect options:
Use an X-Version header to identify which version is being called and pass that header to the Lambda function - This is an incorrect option and has been added as a distractor.
Use an API Gateway Lambda authorizer to route API clients to the correct API version - A Lambda authorizer is an API Gateway feature that uses a Lambda function to control access to your API. A Lambda authorizer is useful if you want to implement a custom authorization scheme that uses a bearer token authentication strategy such as OAuth or SAML, or that uses request parameters to determine the caller's identity.
Set up an API Gateway resource policy to identify the API versions and provide context to the Lambda function - Amazon API Gateway resource policies are JSON policy documents that you attach

to an API to control whether a specified principal (typically an IAM user or role) can invoke the API. You can use API Gateway resource policies to allow your API to be securely invoked requestors. They are not meant for choosing the version of APIs.

References:

https://docs.aws.amazon.com/apigateway/latest/developerguide/stage-variables.html

https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-resource-policies.html

https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html

Question 4:
A development team has configured an Elastic Load Balancer for host-based routing. The idea is to support multiple subdomains and different top-level domains.
The rule *.sample.com matches which of the following?

- SAMPLE.COM
- sample.test.com
- test.sample.com(Correct)
- Sample.com

Explanation
Correct option:
test.sample.com - You can use host conditions to define rules that route requests based on the hostname in the host header (also known as host-based routing). This enables you to support multiple subdomains and different top-level domains using a single load balancer.
A hostname is not case-sensitive, can be up to 128 characters in length, and can contain any of the following characters: 1. A–Z, a–z, 0–9 2. - . 3. * (matches 0 or more characters) 4. ? (matches exactly 1 character)
You must include at least one "." character. You can include only alphabetical characters after the final "." character.
The rule *.sample.com matches test.sample.com but doesn't match sample.com.
Incorrect options:
sample.com
sample.test.com
SAMPLE.COM
These three options contradict the explanation provided above, so these options are incorrect.
Reference:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-listeners.html

Question 5:

The development team at an e-commerce company wants to run a serverless data store service on two docker containers that share resources.

Which of the following ECS configurations can be used to facilitate this use-case?

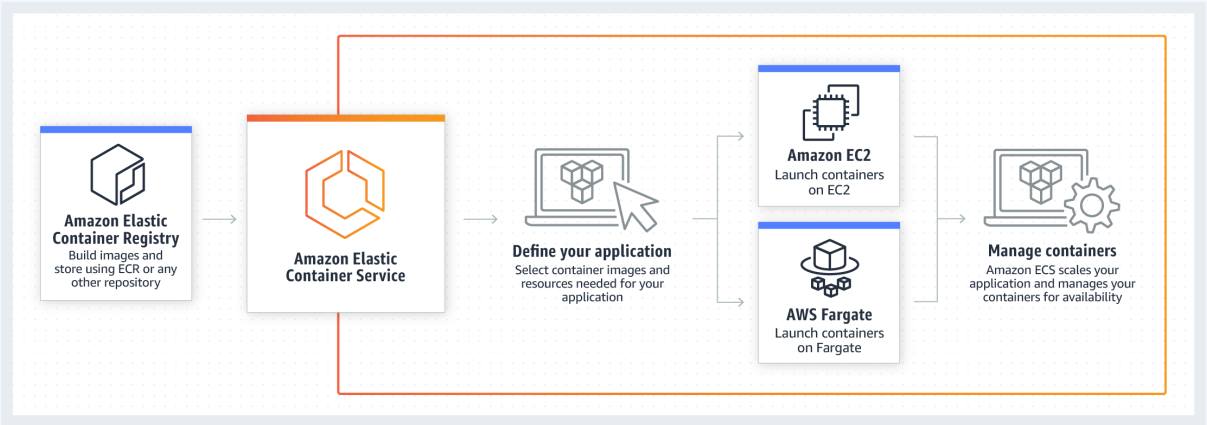
- Put the two containers into two separate task definitions using a Fargate Launch Type
- Put the two containers into two separate task definitions using an EC2 Launch Type
- Put the two containers into a single task definition using an EC2 Launch Type
- Put the two containers into a single task definition using a Fargate Launch Type(Correct)

Explanation

Correct option:

Put the two containers into a single task definition using a Fargate Launch Type

Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast, container management service that makes it easy to run, stop, and manage Docker containers on a cluster. You can host your cluster on a serverless infrastructure that is managed by Amazon ECS by launching your services or tasks using the Fargate launch type. For more control over your infrastructure, you can host your tasks on a cluster of Amazon Elastic Compute Cloud (Amazon EC2) instances that you manage by using the EC2 launch type.



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

As the development team is looking for a serverless data store service, therefore the two containers should be launched into a single task definition using a Fargate Launch Type. Using a single task definition allows the two containers to share resources. Please see these use-cases for Fargate Launch type when you should put multiple containers into the same task definition:

Using the Fargate launch type

When architecting your application using the Fargate launch type for your tasks, the main question is when should you put multiple containers into the same task definition versus deploying containers separately in multiple task definitions.

You should put multiple containers in the same task definition if:

- Containers share a common lifecycle (that is, they should be launched and terminated together).
- Containers are required to be run on the same underlying host (that is, one container references the other on a localhost port).
- You want your containers to share resources.
- Your containers share data volumes.

Otherwise, you should define your containers in separate tasks definitions so that you can scale, provision, and deprovision them separately.

via - https://docs.aws.amazon.com/AmazonECS/latest/developerguide/application_architecture.html

For a deep-dive on understanding how Amazon ECS manages CPU and memory resources, please review this excellent blog-
https://aws.amazon.com/blogs/containers/how-amazon-ecs-manages-cpu-and-memory-resources/

Incorrect options:

Put the two containers into two separate task definitions using a Fargate Launch Type - This option contradicts the details provided in the explanation above, so this option is ruled out.

Put the two containers into two separate task definitions using an EC2 Launch Type

Put the two containers into a single task definition using an EC2 Launch Type

As the development team is looking for a serverless data store service, therefore EC2 Launch Type is ruled out. So both these options are incorrect.

References:

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/application_architecture.html

https://aws.amazon.com/blogs/containers/how-amazon-ecs-manages-cpu-and-memory-resources/

Question 6:
A developer is configuring an Amazon EC2 Auto Scaling Group that has to launch both Spot and On-Demand instances based on the requirement. Also, the CodeDeploy agent has to be automatically installed on these EC2 instances. All the EC2 instances are running on the Amazon Linux operating system.
What is the most operationally efficient way to configure this requirement?

- Configure AWS Resource Access Manager(RAM) to schedule the automatic install of CodeDeploy agent on the EC2 instances. RAM automatic schedules work on only Linux machines and not on Windows operating systems
- Use launch templates to configure the EC2 Auto Scaling Group for On-Demand and spot instances. When you create a launch template use the User data field to add a configuration script that runs when the instance starts. This shell script can, in turn, install the CodeDeploy agent(Correct)
- Use launch configurations to configure the EC2 Auto Scaling Group for On-Demand and spot instances. Add the shell script to the Launch configuration tab on the AWS console. This shell script will install the CodeDeploy agent
- Use AWS Systems Manager for installing and updating the CodeDeploy agent automatically for Spot and On-Demand instances

Explanation
Correct option:
Use launch templates to configure the EC2 Auto Scaling Group for On-Demand and spot instances. When you create a launch template use the User data field to add a configuration script that runs when the instance starts. This shell script can, in turn, install the CodeDeploy agent
A launch template specifies instance configuration information that includes the ID of the Amazon Machine Image (AMI), the instance type, a key pair, security groups, and other parameters used to launch EC2 instances. Defining a launch template instead of a launch configuration allows you to have multiple versions of a launch template.
When you create a launch template, you can use the User data field to add a configuration script that runs when the instance starts. This shell script installs the CodeDeploy agent for all AWS Regions and supported Amazon Linux and Ubuntu distributions. You can configure CodeDeploy to auto-update on boot by setting the AUTOUPDATE variable to true.
Incorrect options:
Use launch configurations to configure the EC2 Auto Scaling Group for On-Demand and spot instances. Add the shell script to the Launch configuration tab on the AWS console. This shell script will install the CodeDeploy agent - This statement is incorrect. Not all Amazon EC2 Auto Scaling features are available when you use launch configurations. For example, you cannot create an Auto Scaling group that launches both Spot and On-Demand Instances or that specifies multiple instance types. You must use a launch template to configure these features. AWS strongly recommends not using launch configurations anymore.

Use AWS Systems Manager for installing and updating the CodeDeploy agent automatically for Spot and On-Demand instances - AWS Systems Manager also uses launch templates to automatically deploy CodeDeploy agent. However, for this use case, SSM is not operationally efficient since SSM agent needs to be installed first on all EC2 instances and then you can install the CodeDeploy agent.
Configure AWS Resource Access Manager(RAM) to schedule the automatic installation of CodeDeploy agent on the EC2 instances. RAM automatic schedules work on only Linux machines and not on Windows operating systems - AWS RAM helps you securely share your resources across AWS accounts, within your organization or organizational units (OUs), and with IAM roles and users for supported resource types. You can use AWS RAM to share resources with other AWS accounts. This eliminates the need to provision and manage resources in every account. RAM cannot be used to install the CodeDeploy agent, so this option is incorrect.
References:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/launch-templates.html
https://aws.amazon.com/premiumsupport/knowledge-center/codedeploy-agent-launch-template/
https://docs.aws.amazon.com/autoscaling/ec2/userguide/launch-configurations.html
https://docs.aws.amazon.com/codedeploy/latest/userguide/codedeploy-agent-operations-install-ssm.html

Question 7:

A company is looking at storing their less frequently accessed files on AWS that can be concurrently accessed by hundreds of EC2 instances. The company needs the most cost-effective file storage service that provides immediate access to data whenever needed.

Which of the following options represents the best solution for the given requirements?

- Amazon Elastic Block Store (EBS)
- Amazon S3 Standard-Infrequent Access (S3 Standard-IA) storage class
- Amazon Elastic File System (EFS) Standard–IA storage class(Correct)
- Amazon Elastic File System (EFS) Standard storage class

Explanation

Correct option:

Amazon Elastic File System (EFS) Standard–IA storage class - Amazon EFS is a file storage service for use with Amazon compute (EC2, containers, serverless) and on-premises servers. Amazon EFS provides a file system interface, file system access semantics (such as strong consistency and file locking), and concurrently accessible storage for up to thousands of Amazon EC2 instances.

The Standard–IA storage class reduces storage costs for files that are not accessed every day. It does this without sacrificing the high availability, high durability, elasticity, and POSIX file system access that Amazon EFS provides. AWS recommends Standard-IA storage if you need your full dataset to be readily accessible and want to automatically save on storage costs for files that are less frequently accessed.

Incorrect options:

Amazon S3 Standard-Infrequent Access (S3 Standard-IA) storage class - Amazon S3 is an object storage service. Amazon S3 makes data available through an Internet API that can be accessed anywhere. It is not a file storage service, as is needed in the use case.

Amazon Elastic File System (EFS) Standard storage class - Amazon EFS Standard storage classes are ideal for workloads that require the highest levels of durability and availability. The EFS Standard storage class is used for frequently accessed files. It is the storage class to which customer data is initially written for Standard storage classes. The company is also looking at cutting costs by optimally storing the infrequently accessed data. Hence, EFS standard storage class is not the right solution for the given use case.

Amazon Elastic Block Store (EBS) - Amazon EBS is a block-level storage service for use with Amazon EC2. Amazon EBS can deliver performance for workloads that require the lowest latency access to data from a single EC2 instance. EBS volume cannot be accessed by hundreds of EC2 instances concurrently. It is not a file storage service, as is needed in the use case.

Reference:

https://docs.aws.amazon.com/efs/latest/ug/storage-classes.html

Question 8:
An organization with high data volume workloads have successfully moved to DynamoDB after having many issues with traditional database systems. However, a few months into production, DynamoDB tables are consistently recording high latency.
As a Developer Associate, which of the following would you suggest to reduce the latency? (Select two)

- Use DynamoDB Accelerator (DAX) for businesses with heavy write-only workloads
- Use eventually consistent reads in place of strongly consistent reads whenever possible(Correct)
- Consider using Global tables if your application is accessed by globally distributed users(Correct)
- Increase the request timeout settings, so the client gets enough time to complete the requests, thereby reducing retries on the system
- Reduce connection pooling, which keeps the connections alive even when user requests are not present, thereby, blocking the services

Explanation
Correct option:
Amazon DynamoDB is a key-value and document database that delivers single-digit millisecond performance at any scale. It's a fully managed, multi-Region, multi-master, durable database with built-in security, backup, and restore and in-memory caching for internet-scale applications.
Consider using Global tables if your application is accessed by globally distributed users - If you have globally dispersed users, consider using global tables. With global tables, you can specify the AWS Regions where you want the table to be available. This can significantly reduce latency for your users. So, reducing the distance between the client and the DynamoDB endpoint is an important performance fix to be considered.
Use eventually consistent reads in place of strongly consistent reads whenever possible - If your application doesn't require strongly consistent reads, consider using eventually consistent reads. Eventually consistent reads are cheaper and are less likely to experience high latency.
Incorrect options:
Increase the request timeout settings, so the client gets enough time to complete the requests, thereby reducing retries on the system - This statement is incorrect. The right way is to reduce the request timeout settings. This causes the client to abandon high latency requests after the specified time period and then send a second request that usually completes much faster than the first.
Reduce connection pooling, which keeps the connections alive even when user requests are not present, thereby, blocking the services - This is not correct. When you're not making requests, consider having the client send dummy traffic to a DynamoDB table. Alternatively, you can reuse client connections or use connection pooling. All of these techniques keep internal caches warm, which helps keep latency low.
Use DynamoDB Accelerator (DAX) for businesses with heavy write-only workloads - This is not correct. If your traffic is read-heavy, consider using a caching service such as DynamoDB Accelerator (DAX). 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.

References:

https://aws.amazon.com/premiumsupport/knowledge-center/dynamodb-high-latency/

https://aws.amazon.com/dynamodb/

Question 9:
Your e-commerce company needs to improve its software delivery process and is moving away from the waterfall methodology. You decided that every application should be built using the best CI/CD practices and every application should be packaged and deployed as a Docker container. The Docker images should be stored in ECR and pushed with AWS CodePipeline and AWS CodeBuild.
When you attempt to do this, the last step fails with an authorization issue. What is the most likely issue?

- The ECR repository is stale, you must delete and re-create it
- CodeBuild cannot talk to ECR because of security group issues
- The ECS instances are misconfigured and must contain additional data in /etc/ecs/ecs.config
- The IAM permissions are wrong for the CodeBuild service(Correct)

Explanation
Correct option:
The IAM permissions are wrong for the CodeBuild service
You can push your Docker or Open Container Initiative (OCI) images to an Amazon ECR repository with the docker push command.
Amazon ECR users require permission to call ecr:GetAuthorizationToken before they can authenticate to a registry and push or pull any images from any Amazon ECR repository. Amazon ECR provides several managed policies to control user access at varying levels
Incorrect options:
The ECR repository is stale, you must delete and re-create it - You can delete a repository when you are done using it, stale is not a concept within ECR. This option has been added as a distractor.
CodeBuild cannot talk to ECR because of security group issues - A security group acts as a virtual firewall at the instance level and it is not related to pushing Docker images, so this option does not fit the given use-case.
The ECS instances are misconfigured and must contain additional data in /etc/ecs/ecs.config - The error Authorization is an indication that there is an access issue, therefore you should not look at your configuration first but rather permissions.
References:
https://docs.aws.amazon.com/AmazonECR/latest/userguide/docker-push-ecr-image.html
https://docs.aws.amazon.com/AmazonECR/latest/userguide/ecr_managed_policies.html

Question 10:
You are working for a technology startup building web and mobile applications. You would like to pull Docker images from the ECR repository called demo so you can start running local tests against the latest application version.
Which of the following commands must you run to pull existing Docker images from ECR? (Select two)

- aws docker push 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest
- $(aws ecr get-login --no-include-email)(Correct)
- docker pull 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest(Correct)
- docker login -u $AWS_ACCESS_KEY_ID -p $AWS_SECRET_ACCESS_KEY
- docker build -t 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest

Explanation
Correct options:
$(aws ecr get-login --no-include-email)
docker pull 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest
The get-login command retrieves a token that is valid for a specified registry for 12 hours, and then it prints a docker login command with that authorization token. You can execute the printed command to log in to your registry with Docker, or just run it automatically using the $() command wrapper. After you have logged in to an Amazon ECR registry with this command, you can use the Docker CLI to push and pull images from that registry until the token expires. The docker pull command is used to pull an image from the ECR registry.
Incorrect options:
docker login -u $AWS_ACCESS_KEY_ID -p $AWS_SECRET_ACCESS_KEY - You cannot login to AWS ECR this way. AWS_ACCESS_KEY_ID and AWS_SECRET_ACCESS_KEY are only used by the CLI and not by docker.
aws docker push 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest - docker push here is the wrong answer, you need to use docker pull.
docker build -t 1234567890.dkr.ecr.eu-west-1.amazonaws.com/demo:latest - This is a docker command that is used to build Docker images from a Dockerfile.
Reference:
https://docs.aws.amazon.com/cli/latest/reference/ecr/get-login.html

Question 11:

A development team is configuring Kinesis Data Streams for ingesting real-time data from various appliances. The team has declared a shard capacity of one to test the configuration.

What happens if the capacity limits of an Amazon Kinesis data stream are exceeded while the data producer adds data to the data stream?

- The put data calls will be rejected with a ProvisionedThroughputExceeded exception(Correct)
- The put data calls will be rejected with a AccessDeniedException exception once the limit is reached
- Contact AWS support to request an increase in the number of shards
- Data is lost unless the partition key of the data records is changed in order to write data to a different shard in the stream

Explanation

Correct option:

The put data calls will be rejected with a ProvisionedThroughputExceeded exception

The capacity limits of an Amazon Kinesis data stream are defined by the number of shards within the data stream. The limits can be exceeded by either data throughput or the number of PUT records. While the capacity limits are exceeded, the put data call will be rejected with a ProvisionedThroughputExceeded exception. If this is due to a temporary rise of the data stream's input data rate, retry by the data producer will eventually lead to completion of the requests. If this is due to a sustained rise of the data stream's input data rate, you should increase the number of shards within your data stream to provide enough capacity for the put data calls to consistently succeed.

Incorrect options:

The put data calls will be rejected with a AccessDeniedException exception once the limit is reached - Access Denied error is thrown when the accessing system does not have enough permissions. Since data was getting ingested into Data Streams before reaching the capacity, this error is not possible.

Data is lost unless the partition key of the data records is changed in order to write data to a different shard in the stream - Partition key is used to segregate and route records to different shards of a data stream. A partition key is specified by your data producer while adding data to an Amazon Kinesis data stream. The use case talks about provisioning only one shard. It is not possible to set up more shards by simply changing the partition key. Hence, this choice is incorrect.

Contact AWS support to request an increase in the number of shards - This is a made-up option that acts as a distractor.

Reference:

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

Question 12:

A company has AWS Lambda functions where each is invoked by other AWS services such as Amazon Kinesis Data Firehose, Amazon API Gateway, Amazon Simple Storage Service, or Amazon CloudWatch Events. What these Lambda functions have in common is that they process heavy workloads such as big data analysis, large file processing, and statistical computations.

What should you do to improve the performance of your AWS Lambda functions without changing your code?

- Change your Lambda function runtime to use Golang
- Increase the RAM assigned to your Lambda function(Correct)
- Change the instance type for your Lambda function
- Increase the Lambda function timeout

Explanation

Correct option:

Increase the RAM assigned to your Lambda function

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

In the AWS Lambda resource model, you choose the amount of memory you want for your function which allocates proportional CPU power and other resources. This means you will have access to more compute power when you choose one of the new larger settings. To configure the memory for your function, set a value between 128 MB and 10,240 MB in 1-MB increments. At 1,769 MB, a function has the equivalent of one vCPU (one vCPU-second of credits per second). You access these settings when you create a function or update its configuration. The settings are available using the AWS Management Console, AWS CLI, or SDKs.

Therefore, by increasing the amount of memory available to the Lambda functions, you can run the compute-heavy workflows.

Incorrect options:

Change the instance type for your Lambda function - Instance types apply to the EC2 service and not to Lambda function as its a serverless service.

Change your Lambda function runtime to use Golang - This changes programming language which requires code changes, so this option is not correct. Besides, changing the runtime may not even address the performance issues.

Increase the Lambda function timeout - This option would increase the amount of time for which the Lambda function executes, which may help in case you have some heavy processing, but won't help with the actual performance of your Lambda function.

Reference:

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

Question 13:
As an AWS Certified Developer Associate, you are writing a CloudFormation template in YAML. The template consists of an EC2 instance creation and one RDS resource. Once your resources are created you would like to output the connection endpoint for the RDS database.
Which intrinsic function returns the value needed?

- !Sub
- !FindInMap
- !Ref
- !GetAtt(Correct)

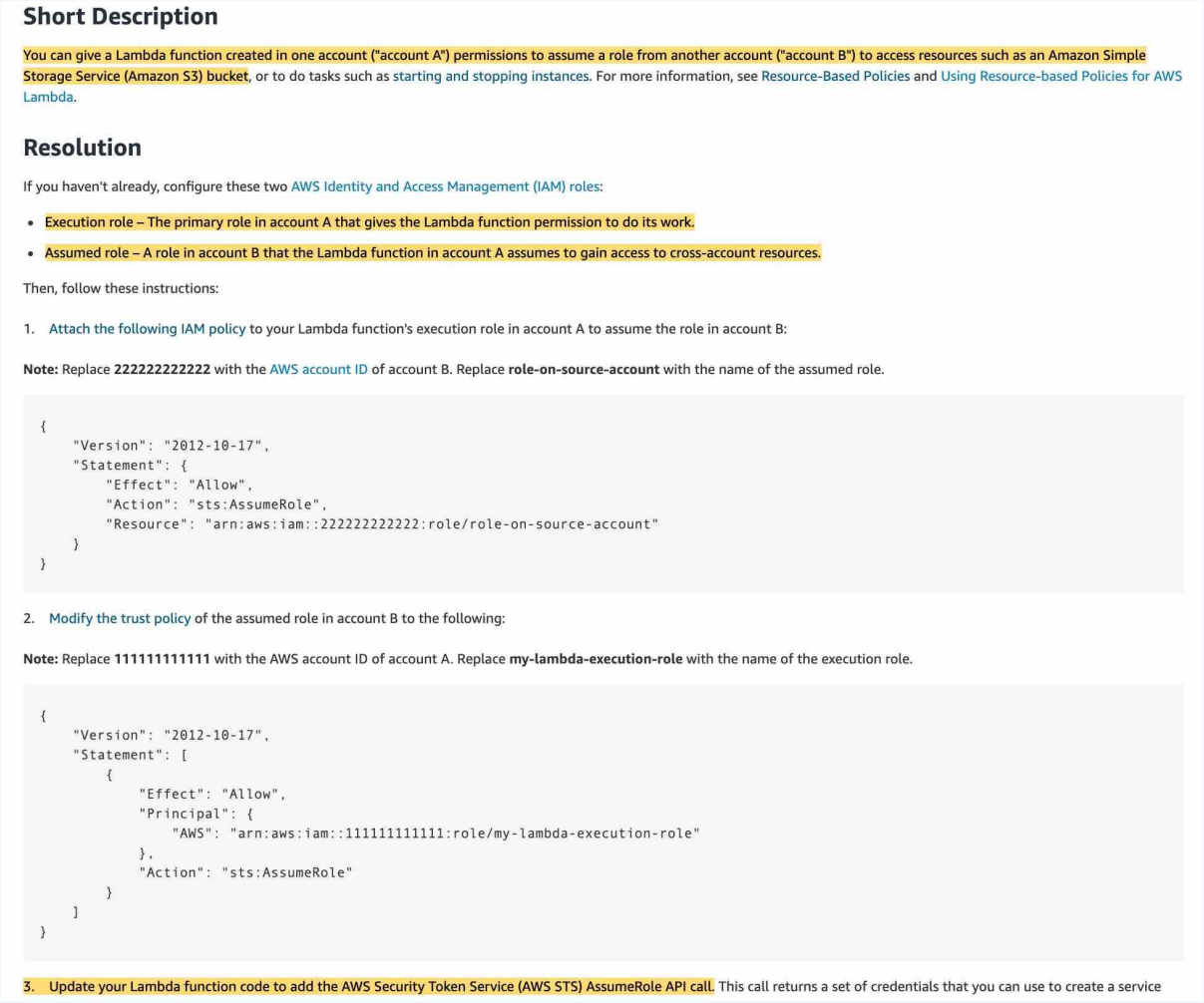
Explanation
Correct option:
AWS CloudFormation provides several built-in functions that help you manage your stacks. Intrinsic functions are used in templates to assign values to properties that are not available until runtime.
!GetAtt - The Fn::GetAtt intrinsic function returns the value of an attribute from a resource in the template. This example snippet returns a string containing the DNS name of the load balancer with the logical name myELB - YML : !GetAtt myELB.DNSName JSON : "Fn::GetAtt" : ["myELB" , "DNSName"]
Incorrect options:
!Sub - The intrinsic function Fn::Sub substitutes variables in an input string with values that you specify. In your templates, you can use this function to construct commands or outputs that include values that aren't available until you create or update a stack.
!Ref - The intrinsic function Ref returns the value of the specified parameter or resource.
!FindInMap - The intrinsic function Fn::FindInMap returns the value corresponding to keys in a two-level map that is declared in the Mappings section. For example, you can use this in the Mappings section that contains a single map, RegionMap, that associates AMIs with AWS regions.
References:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference.html
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-getatt.html

Question 14:
The development team at a retail organization wants to allow a Lambda function in its AWS Account A to access a DynamoDB table in another AWS Account B.
As a Developer Associate, which of the following solutions would you recommend for the given use-case?

- Add a resource policy to the DynamoDB table in AWS Account B to give access to the Lambda function in Account A
- Create an IAM role in Account B with access to DynamoDB. Modify the trust policy of the role in Account B to allow the execution role of Lambda to assume this role. Update the Lambda function code to add the AssumeRole API call(Correct)
- Create an IAM role in Account B with access to DynamoDB. Modify the trust policy of the execution role in Account A to allow the execution role of Lambda to assume the IAM role in Account B. Update the Lambda function code to add the AssumeRole API call
- Create a clone of the Lambda function in AWS Account B so that it can access the DynamoDB table in the same account

Explanation
Correct option:
Create an IAM role in account B with access to DynamoDB. Modify the trust policy of the role in Account B to allow the execution role of Lambda to assume this role. Update the Lambda function code to add the AssumeRole API call
You can give a Lambda function created in one account ("account A") permissions to assume a role from another account ("account B") to access resources such as DynamoDB or S3 bucket. You need to create an execution role in Account A that gives the Lambda function permission to do its work. Then you need to create a role in account B that the Lambda function in account A assumes to gain access to the cross-account DynamoDB table. Make sure that you modify the trust policy of the role in Account B to allow the execution role of Lambda to assume this role. Finally, update the Lambda function code to add the AssumeRole API call.

Sample use-case to configure a Lambda function to assume a role from another AWS account:



Short Description

You can give a Lambda function created in one account ("account A") permissions to assume a role from another account ("account B") to access resources such as an Amazon Simple Storage Service (Amazon S3) bucket, or to do tasks such as starting and stopping instances. For more information, see Resource-Based Policies and Using Resource-based Policies for AWS Lambda.

Resolution

If you haven't already, configure these two AWS Identity and Access Management (IAM) roles:

- Execution role – The primary role in account A that gives the Lambda function permission to do its work.
- Assumed role – A role in account B that the Lambda function in account A assumes to gain access to cross-account resources.

Then, follow these instructions:

1. Attach the following IAM policy to your Lambda function's execution role in account A to assume the role in account B:

Note: Replace **222222222222** with the AWS account ID of account B. Replace **role-on-source-account** with the name of the assumed role.

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::222222222222:role/role-on-source-account"
    }
}
```

2. Modify the trust policy of the assumed role in account B to the following:

Note: Replace **111111111111** with the AWS account ID of account A. Replace **my-lambda-execution-role** with the name of the execution role.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:role/my-lambda-execution-role"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

3. Update your Lambda function code to add the AWS Security Token Service (AWS STS) AssumeRole API call. This call returns a set of credentials that you can use to create a service

via - https://aws.amazon.com/premiumsupport/knowledge-center/lambda-function-assume-iam-role/

Incorrect options:

Create a clone of the Lambda function in AWS Account B so that it can access the DynamoDB table in the same account - Creating a clone of the Lambda function is a distractor as this does not solve the use-case outlined in the problem statement.

Add a resource policy to the DynamoDB table in AWS Account B to give access to the Lambda function in Account A - You cannot attach a resource policy to a DynamoDB table, so this option is incorrect.

Create an IAM role in Account B with access to DynamoDB. Modify the trust policy of the execution role in Account A to allow the execution role of Lambda to assume the IAM role in Account B. Update the Lambda function code to add the AssumeRole API call - As mentioned in the explanation above, you need to modify the trust policy of the IAM role in Account B so that it allows the execution role of Lambda function in account A to assume the IAM role in Account B.

Reference:

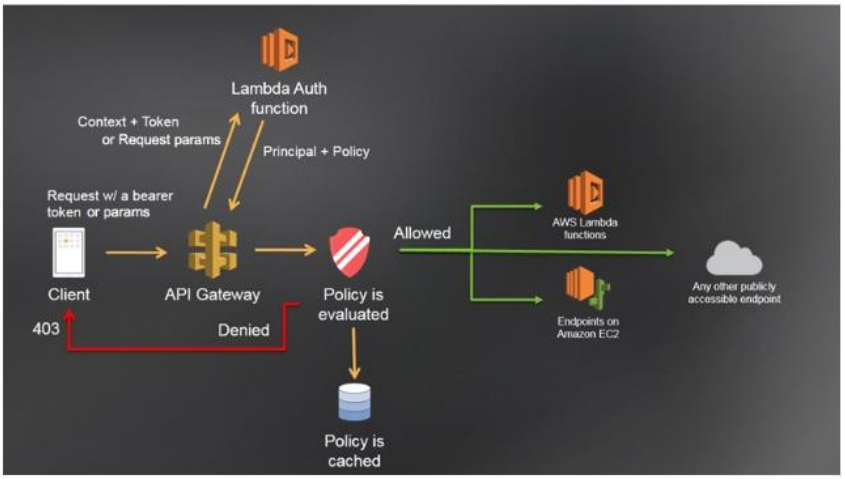
https://aws.amazon.com/premiumsupport/knowledge-center/lambda-function-assume-iam-role/

Question 15:
A company wants to implement authentication for its new RESTful API service that uses Amazon API Gateway. To authenticate the calls, each request must include HTTP headers with a client ID and user ID. These credentials must be compared to the authentication data in a DynamoDB table.
As an AWS Certified Developer Associate, which of the following would you recommend for implementing this authentication in API Gateway?

- Update the API Gateway integration requests to require the credentials, then grant API Gateway access to the authentication table in DynamoDB
- Set up an API Gateway Model that requires the credentials, then grant API Gateway access to the authentication table in DynamoDB
- Develop an AWS Lambda authorizer that references the authentication data in the DynamoDB table(Correct)
- Authorize using Amazon Cognito that will reference the authentication table of DynamoDB

Explanation
Correct option:
Develop an AWS Lambda authorizer that references the DynamoDB authentication table - A Lambda authorizer is an API Gateway feature that uses a Lambda function to control access to your API.
A Lambda authorizer is useful if you want to implement a custom authorization scheme that uses a bearer token authentication strategy such as OAuth or SAML, or that uses request parameters to determine the caller's identity.
When a client makes a request to one of your API's methods, API Gateway calls your Lambda authorizer, which takes the caller's identity as input and returns an IAM policy as output.
There are two types of Lambda authorizers:
A token-based Lambda authorizer (also called a TOKEN authorizer) receives the caller's identity in a bearer token, such as a JSON Web Token (JWT) or an OAuth token.
A request parameter-based Lambda authorizer (also called a REQUEST authorizer) receives the caller's identity in a combination of headers, query string parameters, state variables, and $context variables.

API Gateway Lambda authorization workflow:



API Gateway Lambda authorization workflow

1. The client calls a method on an API Gateway API method, passing a bearer token or request parameters.
2. API Gateway checks whether a Lambda authorizer is configured for the method. If it is, API Gateway calls the Lambda function.
3. The Lambda function authenticates the caller by means such as the following:
   - Calling out to an OAuth provider to get an OAuth access token.
   - Calling out to a SAML provider to get a SAML assertion.
   - Generating an IAM policy based on the request parameter values.
   - Retrieving credentials from a database.
4. If the call succeeds, the Lambda function grants access by returning an output object containing at least an IAM policy and a principal identifier.
5. API Gateway evaluates the policy.
   - If access is denied, API Gateway returns a suitable HTTP status code, such as `403 ACCESS_DENIED`.
   - If access is allowed, API Gateway executes the method. If caching is enabled in the authorizer settings, API Gateway also caches the policy so that the Lambda authorizer function doesn't need to be invoked again.

via - https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html

Incorrect options:

Set up an API Gateway Model that requires the credentials, then grant API Gateway access to the authentication table in DynamoDB - In API Gateway, a model defines the data structure of a payload. In API Gateway models are defined using the JSON schema draft 4. Models are not mandatory.

Update the API Gateway integration requests to require the credentials, then grant API Gateway access to the authentication table in DynamoDB - After setting up an API method, you must integrate it with an endpoint in the backend. A backend endpoint is also referred to as an integration endpoint and can be a Lambda function, an HTTP webpage, or an AWS service action.

An integration request is an HTTP request that API Gateway submits to the backend, passing along the client-submitted request data, and transforming the data, if necessary. The HTTP method (or verb) and URI of the integration request are dictated by the backend (that is, the integration endpoint). They can be the same as or different from the method request's HTTP method and URI, respectively.

Authorize using Amazon Cognito that will reference the authentication table of DynamoDB - As an alternative to using IAM roles and policies or Lambda authorizers (formerly known as custom authorizers), you can use an Amazon Cognito user pool to control who can access your API in Amazon API Gateway.

To use an Amazon Cognito user pool with your API, you must first create an authorizer of the COGNITO_USER_POOLS type and then configure an API method to use that authorizer. After the API is deployed, the client must first sign the user in to the user pool, obtain an identity or access token for the user, and then call the API method with one of the tokens, which are typically set to the request's Authorization header. The API call succeeds only if the required token is supplied and the supplied token is valid, otherwise, the client isn't authorized to make the call because the client did not have credentials that could be authorized.
References:
https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-lambda-authorizer.html
https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-integrate-with-cognito.html
https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-integration-settings.html
https://docs.aws.amazon.com/apigateway/latest/developerguide/models-mappings.html

Question 16:
A developer has just integrated an AWS Lambda function to an Amazon API Gateway API. The integration has led to errors that the developer is unable to troubleshoot. The developer has decided to enable CloudWatch logging at the method level for the API Gateway API.
What are the key points of consideration while configuring configuring method-level logging for the API Gateway? (Select two)

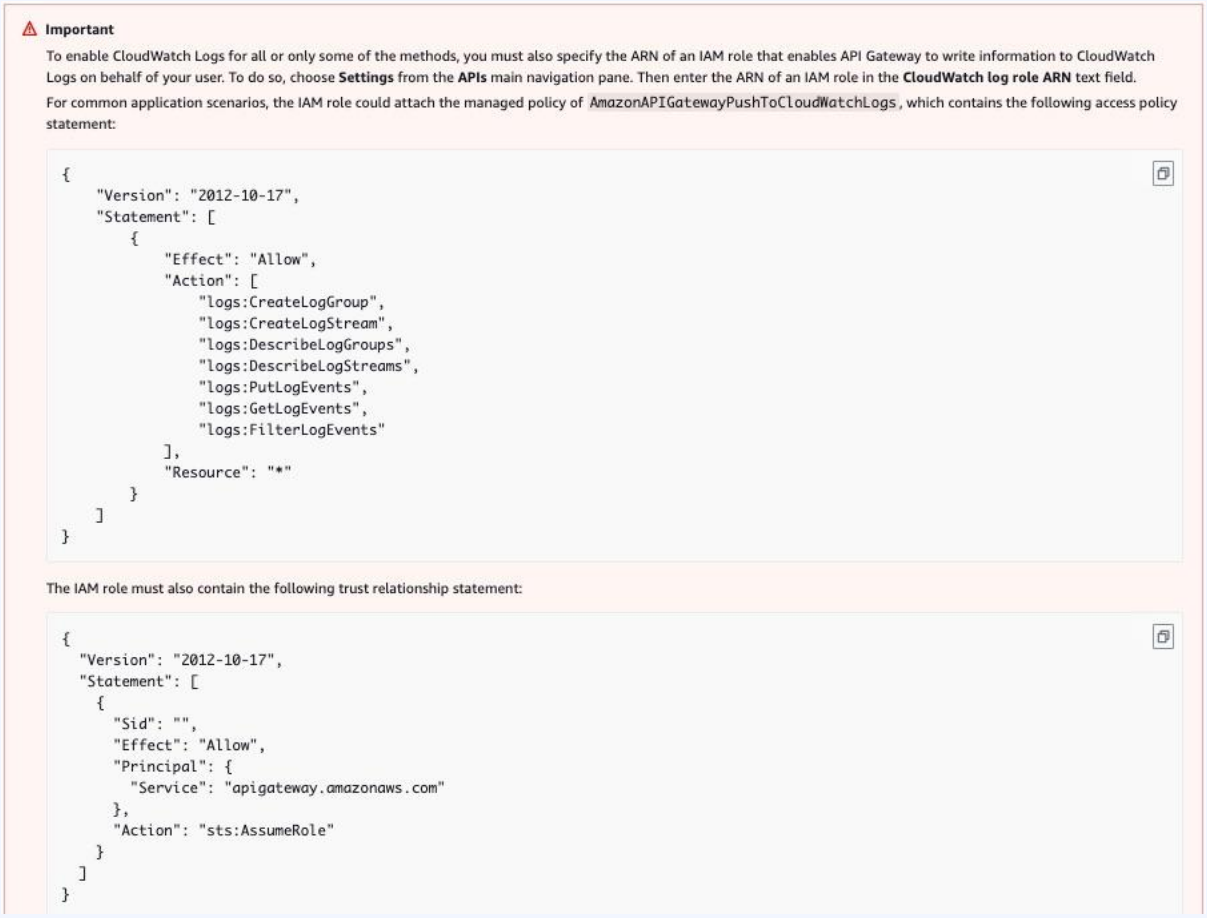
- AWS Security Token Service(STS) is used by API Gateway for logging data to CloudWatch logs. Hence, AWS STS has to be enabled for the Region that you're using(Correct)

- To enable CloudWatch Logs for all or only some of the methods, you must also specify the ARN of an IAM role that enables API Gateway to write information to CloudWatch Logs on behalf of your user. The IAM role must also contain the following trust relationship statement(Correct)

- You are charged for accessing method-level and stage-level CloudWatch metrics, but not for API-level metrics
- API Gateway API log groups or streams can only be deleted and recreated by redeploying the API
- In access logging, only $context and $input variables are supported

Explanation
Correct options:
AWS Security Token Service(STS) is used by API Gateway for logging data to CloudWatch logs. Hence, AWS STS has to be enabled for the Region that you're using
API Gateway calls AWS Security Token Service to assume the IAM role, so make sure that AWS STS is enabled for the Region. If you receive an error when setting the IAM role ARN, check your AWS Security Token Service account settings to make sure that AWS STS is enabled in the Region that you're using.
To enable CloudWatch Logs for all or only some of the methods, you must also specify the ARN of an IAM role that enables API Gateway to write information to CloudWatch Logs on behalf of your user. The IAM role must also contain the following trust relationship statement
To enable CloudWatch Logs for all or only some of the methods, you must also specify the ARN of an IAM role that enables API Gateway to write information to CloudWatch Logs on behalf of your user. To do so, choose Settings from the APIs main navigation pane. Then enter the ARN of an IAM role in the CloudWatch log role ARN text field. The IAM role must also contain the trust relationship statement.

Policy of AmazonAPIGatewayPushToCloudWatchLogs for IAM role:



Important

To enable CloudWatch Logs for all or only some of the methods, you must also specify the ARN of an IAM role that enables API Gateway to write information to CloudWatch Logs on behalf of your user. To do so, choose **Settings** from the **APIs** main navigation pane. Then enter the ARN of an IAM role in the **CloudWatch log role ARN** text field.

For common application scenarios, the IAM role could attach the managed policy of `AmazonAPIGatewayPushToCloudWatchLogs`, which contains the following access policy statement:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogGroups",
                "logs:DescribeLogStreams",
                "logs:PutLogEvents",
                "logs:GetLogEvents",
                "logs:FilterLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

The IAM role must also contain the following trust relationship statement:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Service": "apigateway.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

via - https://docs.aws.amazon.com/apigateway/latest/developerguide/stages.html#how-to-stage-settings-console

Incorrect options:

In access logging, only $context and $input variables are supported - This statement is incorrect. In access logging, you, as an API developer, want to log who has accessed your API and how the caller accessed the API. You can create your own log group or choose an existing log group that could be managed by API Gateway. To specify the access details, you select $context variables and choose a log group as the destination. Only $context variables are supported (not $input, and so on).

You are charged for accessing method-level and stage-level CloudWatch metrics, but not for API-level metrics - This statement is incorrect. Your account is charged for accessing method-level CloudWatch metrics, but not the API-level or stage-level metrics.

API Gateway API log groups or streams can only be deleted and recreated by redeploying the API - API Gateway API log groups or streams can be deleted from the CloudWatch console. But, it is not recommended.

Do not manually delete API Gateway API log groups or streams; let API Gateway manage these resources. Manually deleting log groups or streams may cause API requests and responses not to be logged. If that happens, you can delete the entire log group for the API and redeploy the API. This is because API Gateway creates log groups or log streams for an API stage at the time when it is deployed.

References:
https://docs.aws.amazon.com/apigateway/latest/developerguide/stages.html#how-to-stage-settings-console
https://docs.aws.amazon.com/apigateway/latest/developerguide/set-up-logging.html#apigateway-cloudwatch-log-formats
https://docs.aws.amazon.com/apigateway/latest/developerguide/view-cloudwatch-log-events-in-cloudwatch-console.html

Question 17:
A developer is configuring Amazon ECS container instances to send log information to CloudWatch Logs. For the container instances to be able to send log data to CloudWatch Logs, an IAM policy needs to be created that will allow the container instances to use the CloudWatch Logs APIs.
Which policy is the right fit for the given requirement?

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:DescribeLogGroups"
         ],
         "Resource": [
            "arn:aws:logs:*:*:*"
         ]
      }
   ]
}
```

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:DescribeLogStreams"
         ],
         "Resource": [
            "arn:aws:logs:*:*:*"
         ]
      }
   ]
}
```

(Correct)

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents,
            "ecs:DescribeServices"
         ],
         "Resource": [
            "arn:aws:logs:<ARN of the Log Group>"
         ]
      }
   ]
}

{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents"
         ],
         "Resource": [
            "arn:aws:logs:*:*:*"
         ]
      }
   ]
}
```

Explanation

Correct option:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
```

```
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents",
            "logs:DescribeLogStreams"
         ],
         "Resource": [
            "arn:aws:logs:*:*:*"
         ]
      }
   ]
}
```

Before your container instances can send log data to CloudWatch Logs, you must create an IAM policy to allow your container instances to use the CloudWatch Logs APIs, and then you must attach that policy to ecsInstanceRole.
This policy has one statement that grants permissions to create log groups and log streams, to upload log events to log streams, and to list details about log streams.
The wildcard character (*) at the end of the Resource value means that the statement allows permission for the logs:CreateLogGroup, logs:CreateLogStream, logs:PutLogEvents, and logs:DescribeLogStreams actions on any log group. To limit this permission to a specific log group, replace the wildcard character (*) in the resource ARN with the specific log group ARN
Incorrect options:

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Effect": "Allow",
         "Action": [
            "logs:CreateLogGroup",
            "logs:CreateLogStream",
            "logs:PutLogEvents"
         ],
         "Resource": [
            "arn:aws:logs:*:*:*"
         ]
      }
   ]
}
```

Permission to list details of the log stream needs to be attached to this policy.

```
{
   "Version": "2012-10-17",
   "Statement": [
```

```
      {
        "Effect": "Allow",
        "Action": [
          "logs:CreateLogGroup",
          "logs:CreateLogStream",
          "logs:PutLogEvents,
          "ecs:DescribeServices"
        ],
        "Resource": [
          "arn:aws:logs:<ARN of the Log Group>"
        ]
      }
   ]
}```

- ecs:DescribeServices permission is not needed, but logs:DescribeLogStreams permissions are needed for the policy to perform as expected.

{ "Version": "2012-10-17", "Statement": [ { "Effect": "Allow", "Action": [ "logs:CreateLogGroup", "logs:CreateLogStream", "logs:PutLogEvents", "logs:DescribeLogGroups" ], "Resource": [ "arn:aws:logs:::*" ] } ] } ```
logs:DescribeLogGroups is an erroneous permission here.
References:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/using_cloudwatch_logs.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/iam-identity-based-access-control-cwl.html
```

Question 18:

An e-commerce company uses Amazon SQS queues to decouple their application architecture. The development team has observed message processing failures for an edge case scenario when a user places an order for a particular product ID, but the product ID is deleted, thereby causing the application code to fail.

As a Developer Associate, which of the following solutions would you recommend to address such message failures?

- Use a temporary queue to handle message processing failures
- Use long polling to handle message processing failures
- Use short polling to handle message processing failures
- Use a dead-letter queue to handle message processing failures(Correct)

Explanation

Correct option:

Use a dead-letter queue to handle message processing failures

Dead-letter queues can be used by other queues (source queues) as a target for messages that can't be processed (consumed) successfully. Dead-letter queues are useful for debugging your application or messaging system because they let you isolate problematic messages to determine why their processing doesn't succeed.

Sometimes, messages can’t be processed because of a variety of possible issues, such as when a user comments on a story but it remains unprocessed because the original story itself is deleted by the author while the comments were being posted. In such a case, the dead-letter queue can be used to handle message processing failures.

How do dead-letter queues work?

How do dead-letter queues work?

Sometimes, messages can't be processed because of a variety of possible issues, such as erroneous conditions within the producer or consumer application or an unexpected state change that causes an issue with your application code. For example, if a user places a web order with a particular product ID, but the product ID is deleted, the web store's code fails and displays an error, and the message with the order request is sent to a dead-letter queue.

Occasionally, producers and consumers might fail to interpret aspects of the protocol that they use to communicate, causing message corruption or loss. Also, the consumer's hardware errors might corrupt message payload.

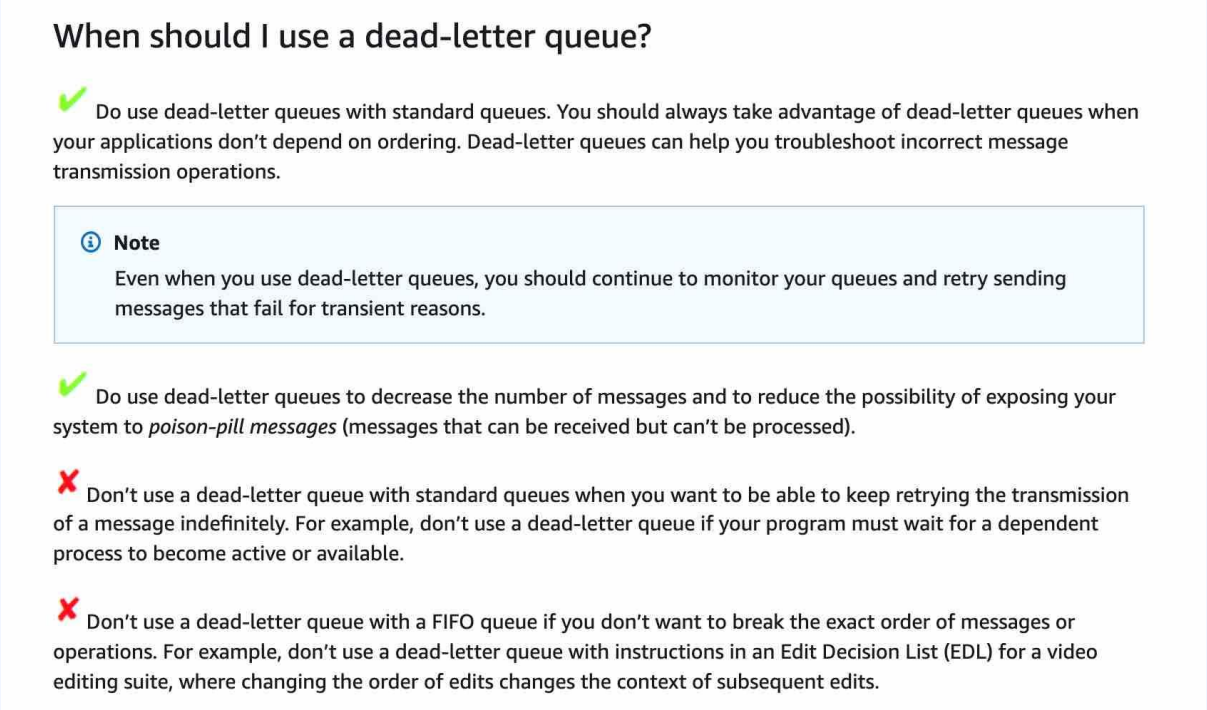
The *redrive policy* specifies the *source queue*, the *dead-letter queue*, and the conditions under which Amazon SQS moves messages from the former to the latter if the consumer of the source queue fails to process a message a specified number of times. When the `ReceiveCount` for a message exceeds the `maxReceiveCount` for a queue, Amazon SQS moves the message to a dead-letter queue (with its original message ID). For example, if the source queue has a redrive policy with `maxReceiveCount` set to 5, and the consumer of the source queue receives a message 6 times without ever deleting it, Amazon SQS moves the message to the dead-letter queue.

To specify a dead-letter queue, you can use the console or the AWS SDK for Java. You must do this for each queue that sends messages to a dead-letter queue. Multiple queues of the same type can target a single dead-letter queue. For more information, see Configuring a dead-letter queue (console) and the `RedrivePolicy` attribute of the `CreateQueue` or `SetQueueAttributes` action.

> **Important**
>
> The dead-letter queue of a FIFO queue must also be a FIFO queue. Similarly, the dead-letter queue of a standard queue must also be a standard queue.
>
> You must use the same AWS account to create the dead-letter queue and the other queues that send messages to the dead-letter queue. Also, dead-letter queues must reside in the same region as the other queues that use the dead-letter queue. For example, if you create a queue in the US East (Ohio) region and you want to use a dead-letter queue with that queue, the second queue must also be in the US East (Ohio) region.
>
> The expiration of a message is always based on its original enqueue timestamp. When a message is moved to a dead-letter queue, the enqueue timestamp remains unchanged. For example, if a message spends 1 day in the original queue before being moved to a dead-letter queue, and the retention period of the dead-letter queue is set to 4 days, the message is deleted from the dead-letter queue after 3 days. Thus, it is a best practice to always set the retention period of a dead-letter queue to be longer than the retention period of the original queue.

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

Use-cases for dead-letter queues:



When should I use a dead-letter queue?

✔ Do use dead-letter queues with standard queues. You should always take advantage of dead-letter queues when your applications don't depend on ordering. Dead-letter queues can help you troubleshoot incorrect message transmission operations.

Note

Even when you use dead-letter queues, you should continue to monitor your queues and retry sending messages that fail for transient reasons.

✔ Do use dead-letter queues to decrease the number of messages and to reduce the possibility of exposing your system to *poison-pill messages* (messages that can be received but can't be processed).

✘ Don't use a dead-letter queue with standard queues when you want to be able to keep retrying the transmission of a message indefinitely. For example, don't use a dead-letter queue if your program must wait for a dependent process to become active or available.

✘ Don't use a dead-letter queue with a FIFO queue if you don't want to break the exact order of messages or operations. For example, don't use a dead-letter queue with instructions in an Edit Decision List (EDL) for a video editing suite, where changing the order of edits changes the context of subsequent edits.

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

Incorrect options:

Use a temporary queue to handle message processing failures - The most common use case for temporary queues is the request-response messaging pattern (for example, processing a login request), where a requester creates a temporary queue for receiving each response message. To avoid creating an Amazon SQS queue for each response message, the Temporary Queue Client lets you create and delete multiple temporary queues without making any Amazon SQS API calls. Temporary queues cannot be used to handle message processing failures.

Use short polling to handle message processing failures

Use long polling to handle message processing failures

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. Neither short polling nor long polling can be used to handle message processing failures.

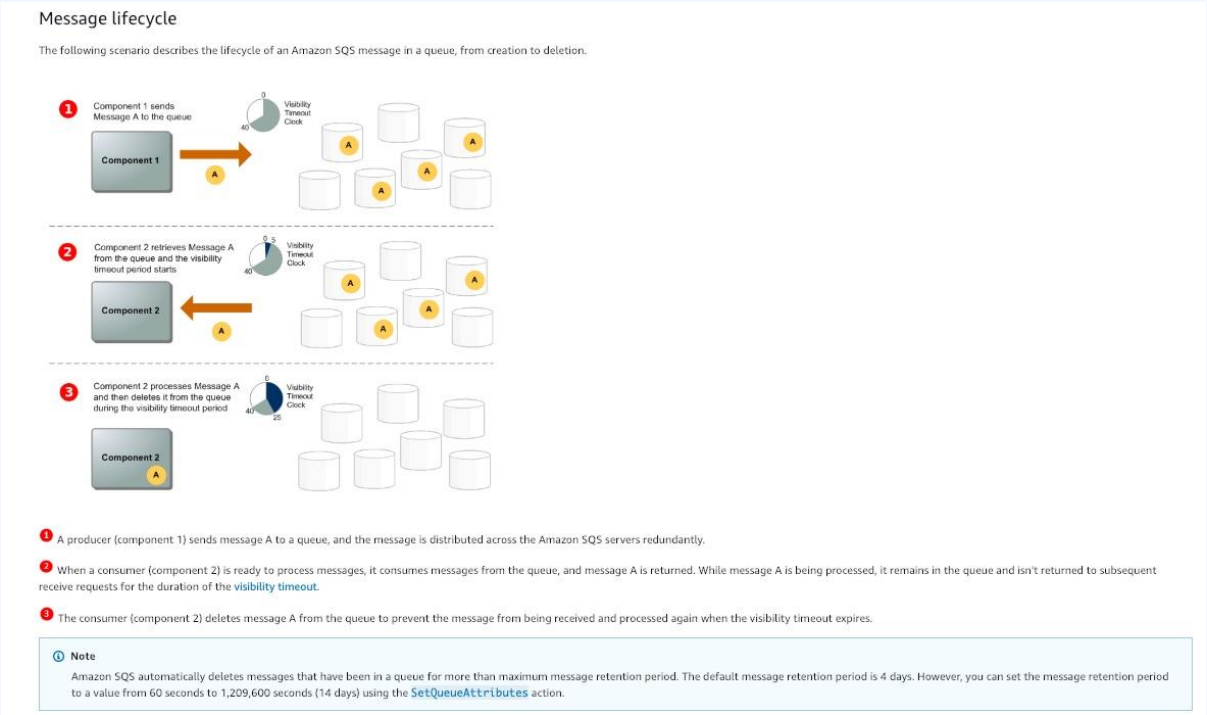
Reference:

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

Question 19:
As a Developer, you are working on a mobile application that utilizes Amazon Simple Queue Service (SQS) for sending messages to downstream systems for further processing. One of the requirements is that the messages should be stored in the queue for a period of 12 days.
How will you configure this requirement?

- Use a FIFO SQS queue
- Change the queue message retention setting(Correct)
- The maximum retention period of SQS messages is 7 days, therefore retention period of 12 days is not possible
- Enable Long Polling for the SQS queue

Explanation
Correct option:
Change the queue message retention setting - Amazon SQS automatically deletes messages that have been in a queue for more than the maximum message retention period. The default message retention period is 4 days. However, you can set the message retention period to a value from 60 seconds to 1,209,600 seconds (14 days) using the SetQueueAttributes action.
More info here:



via -
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-basic-architecture.html
Incorrect options:
Enable Long Polling for the SQS queue - Amazon SQS provides short polling and long polling to receive messages from a queue. By default, queues use short 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). This feature is not useful for the current use case.
The maximum retention period of SQS messages is 7 days, therefore retention period of 12 days is not possible - This is an incorrect statement. Retention period of up to 14 days is possible.
Use a FIFO SQS queue - FIFO (First-In-First-Out) queues are designed to enhance messaging between applications when the order of operations and events is critical, or where duplicates can't be tolerated. This is not useful for the current scenario.
References:
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-basic-architecture.html
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-short-and-long-polling.html
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues.html

Question 20:

You're a developer for 'Movie Gallery', a company that just migrated to the cloud. A database must be created using NoSQL technology to hold movies that are listed for public viewing. You are taking an important step in designing the database with DynamoDB and need to choose the appropriate partition key.

Which of the following unique attributes satisfies this requirement?

- producer_name
- movie_language
- lead_actor_name
- movie_id(Correct)

Explanation

Correct option:

DynamoDB stores data as groups of attributes, known as items. Items are similar to rows or records in other database systems. DynamoDB stores and retrieves each item based on the primary key value, which must be unique. Items are distributed across 10-GB storage units, called partitions (physical storage internal to DynamoDB).

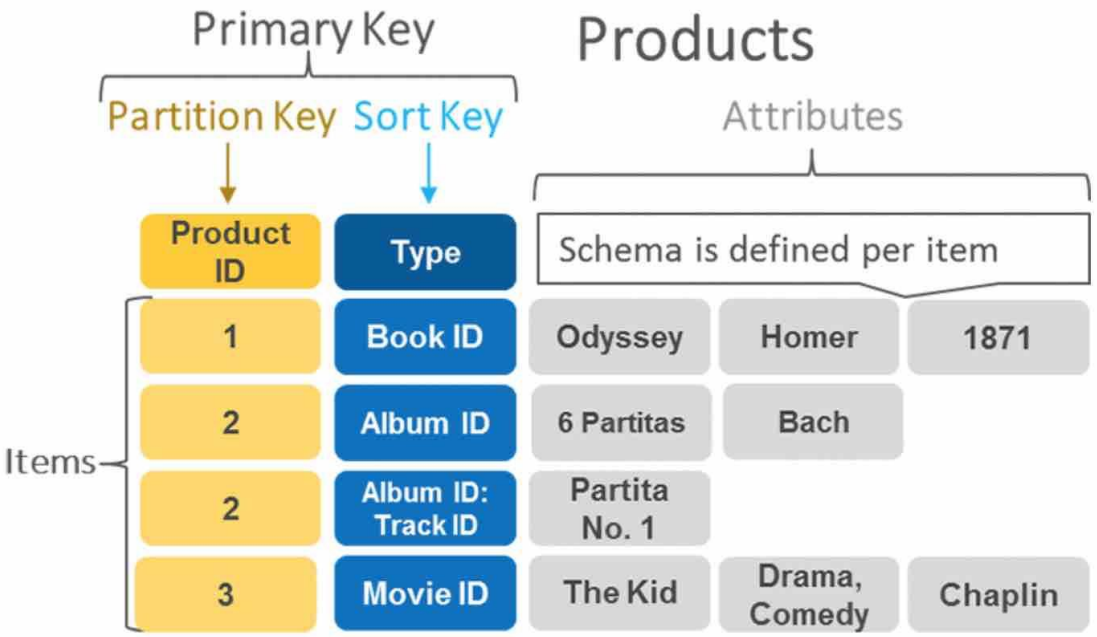
DynamoDB uses the partition key's value as an input to an internal hash function. The output from the hash function determines the partition in which the item is stored. Each item's location is determined by the hash value of its partition key.

Please see these details for the DynamoDB Partition Keys:

What is a partition key?

DynamoDB supports two types of primary keys:

- Partition key: A simple primary key, composed of one attribute known as the *partition key*. Attributes in DynamoDB are similar in many ways to fields or columns in other database systems.
- Partition key and sort key: Referred to as a *composite primary key*, this type of key is composed of two attributes. The first attribute is the *partition key*, and the second attribute is the *sort key*. Following is an example.



via - https://aws.amazon.com/blogs/database/choosing-the-right-dynamodb-partition-key/

Recommendations for partition keys

Use high-cardinality attributes. These are attributes that have distinct values for each item, like `e-mailid`, `employee_no`, `customerid`, `sessionid`, `orderid`, and so on.

Use composite attributes. Try to combine more than one attribute to form a unique key, if that meets your access pattern. For example, consider an orders table with `customerid+productid+countrycode` as the partition key and `order_date` as the sort key.

Cache the popular items when there is a high volume of read traffic using Amazon DynamoDB Accelerator (DAX). The cache acts as a low-pass filter, preventing reads of unusually popular items from swamping partitions. For example, consider a table that has deals information for products. Some deals are expected to be more popular than others during major sale events like Black Friday or Cyber Monday. DAX is a fully managed, in-memory cache for DynamoDB that doesn't require developers to manage cache invalidation, data population, or cluster management. DAX also is compatible with DynamoDB API calls, so developers can incorporate it more easily into existing applications.

Add random numbers or digits from a predetermined range for write-heavy use cases. Suppose that you expect a large volume of writes for a partition key (for example, greater than 1000 1 K writes per second). In this case, use an additional prefix or suffix (a fixed number from predetermined range, say 1–10) and add it to the partition key.

For example, consider a table of invoice transactions. A single invoice can contain thousands of transactions per client. How do we enforce uniqueness and ability to query and update the invoice details for high-volumetric clients?

Following is the recommended table layout for this scenario:

- Partition key: Add a random suffix (1–10 or 1–100) with the `InvoiceNumber`, depending on the number of transactions per `InvoiceNumber`. For example, assume that a single `InvoiceNumber` contains up to 50,000 1K items and that you expect 5000 writes per second. In this case, you can use the following formula to estimate the suffix range: (Number of writes per second * (roundup (item size in KB),0)* 1KB) /1000). Using this formula requires a minimum of five partitions to distribute writes, and hence you might want to set the range as 1-5.
- Sort key: `ClientTransactionid`

Partition Key	Sort Key	Attribute1
InvoiceNumber+Randomsuffix	ClientTransactionid	Invoice_Date
121212-1	Client1_trans1	2016-05-17 01.36.45
121212-1	Client1-trans2	2016-05-18 01.36.30
121212-2	Client2_trans1	2016-06-15 01.36.20
121212-2	Client2_trans2	2016-07-1 01.36.15

via - https://aws.amazon.com/blogs/database/choosing-the-right-dynamodb-partition-key/

movie_id

The movie_id attribute has high-cardinality across the entire collection of the movie database, hence it is the most suitable candidate for the partition key in this use case.

Incorrect options:

producer_name - Does not qualify because the attribute will have duplicate values (and therefore low cardinality)

lead_actor_name - Does not qualify because the attribute will have duplicate values (and therefore low cardinality)

movie_language - Does not qualify because the attribute will have duplicate values (and therefore low cardinality)

Reference:
https://aws.amazon.com/blogs/database/choosing-the-right-dynamodb-partition-key/

Question 21:

A development team has created AWS CloudFormation templates that are reusable by taking advantage of input parameters to name resources based on client names.

You would like to save your templates on the cloud, which storage option should you choose?

- ECR
- EFS
- S3(Correct)
- EBS

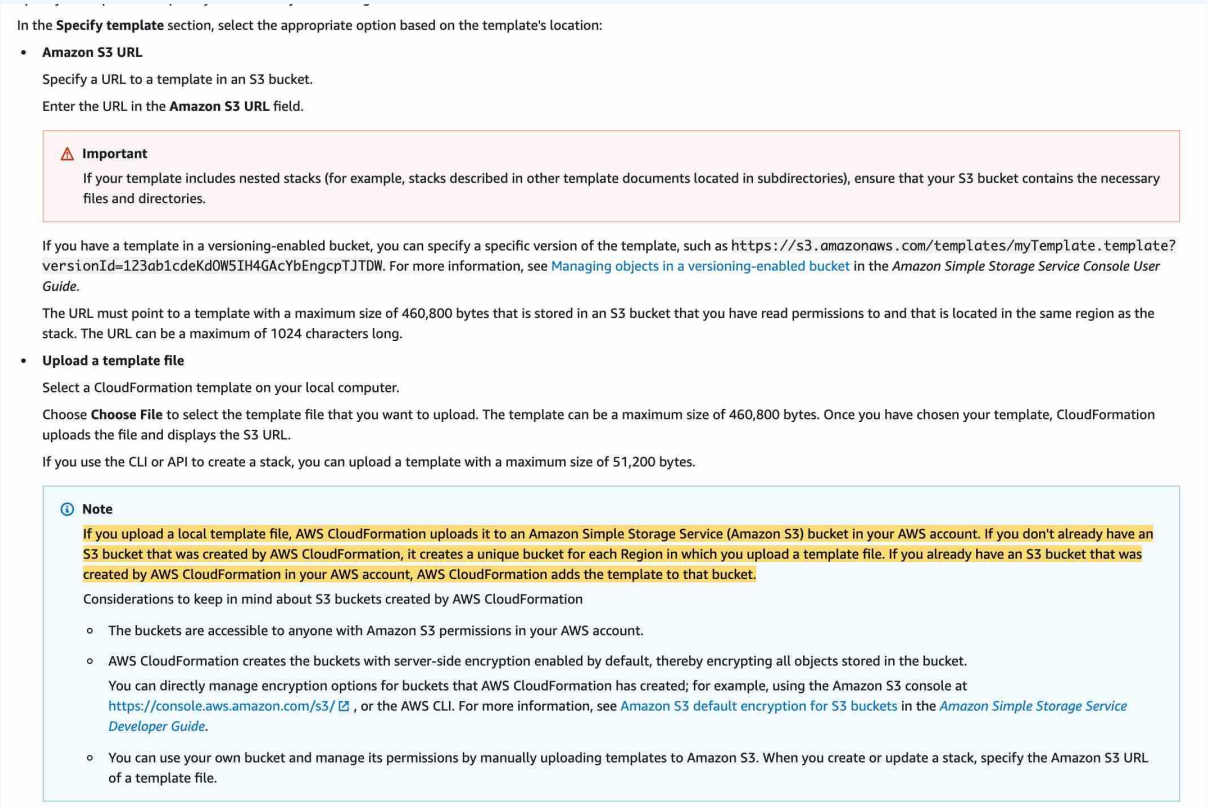
Explanation

Correct option:

S3

If you upload a local template file, AWS CloudFormation uploads it to an Amazon Simple Storage Service (Amazon S3) bucket in your AWS account. If you don't already have an S3 bucket that was created by AWS CloudFormation, it creates a unique bucket for each region in which you upload a template file. If you already have an S3 bucket that was created by AWS CloudFormation in your AWS account, AWS CloudFormation adds the template to that bucket.

Selecting a stack template for CloudFormation:



In the **Specify template** section, select the appropriate option based on the template's location:

- **Amazon S3 URL**

  Specify a URL to a template in an S3 bucket.

  Enter the URL in the **Amazon S3 URL** field.

  Important

  If your template includes nested stacks (for example, stacks described in other template documents located in subdirectories), ensure that your S3 bucket contains the necessary files and directories.

  If you have a template in a versioning-enabled bucket, you can specify a specific version of the template, such as `https://s3.amazonaws.com/templates/myTemplate.template?versionId=123ab1cdeKdOW5IH4GAcYbEngcpTJTDW`. For more information, see Managing objects in a versioning-enabled bucket in the *Amazon Simple Storage Service Console User Guide*.

  The URL must point to a template with a maximum size of 460,800 bytes that is stored in an S3 bucket that you have read permissions to and that is located in the same region as the stack. The URL can be a maximum of 1024 characters long.

- **Upload a template file**

  Select a CloudFormation template on your local computer.

  Choose **Choose File** to select the template file that you want to upload. The template can be a maximum size of 460,800 bytes. Once you have chosen your template, CloudFormation uploads the file and displays the S3 URL.

  If you use the CLI or API to create a stack, you can upload a template with a maximum size of 51,200 bytes.

  Note

  If you upload a local template file, AWS CloudFormation uploads it to an Amazon Simple Storage Service (Amazon S3) bucket in your AWS account. If you don't already have an S3 bucket that was created by AWS CloudFormation, it creates a unique bucket for each Region in which you upload a template file. If you already have an S3 bucket that was created by AWS CloudFormation in your AWS account, AWS CloudFormation adds the template to that bucket.

  Considerations to keep in mind about S3 buckets created by AWS CloudFormation

  - The buckets are accessible to anyone with Amazon S3 permissions in your AWS account.
  - AWS CloudFormation creates the buckets with server-side encryption enabled by default, thereby encrypting all objects stored in the bucket.

    You can directly manage encryption options for buckets that AWS CloudFormation has created; for example, using the Amazon S3 console at https://console.aws.amazon.com/s3/ , or the AWS CLI. For more information, see Amazon S3 default encryption for S3 buckets in the *Amazon Simple Storage Service Developer Guide*.
  - You can use your own bucket and manage its permissions by manually uploading templates to Amazon S3. When you create or update a stack, specify the Amazon S3 URL of a template file.

via - https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-using-console-create-stack-template.html

Incorrect options:

EBS - An Amazon EBS volume is a durable, block-level storage device that you can attach to your instances. After you attach a volume to an instance, you can use it as you would use a physical hard

drive. EBS volumes are flexible. Amazon EBS is a recommended storage option when data must be quickly accessible and requires long-term persistence. EBS cannot be used for selecting a stack template for CloudFormation.
EFS - EFS is a file storage service where you mount the file system on an Amazon EC2 Linux-based instance which is not an option for CloudFormation.
ECR - Amazon ECR eliminates the need to operate your container repositories or worry about scaling the underlying infrastructure which does not apply to CloudFormation.
Reference:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/cfn-using-console-create-stack-template.html

Question 22:
Your web application front end consists of 5 EC2 instances behind an Application Load Balancer. You have configured your web application to capture the IP address of the client making requests. When viewing the data captured you notice that every IP address being captured is the same, which also happens to be the IP address of the Application Load Balancer.
What should you do to identify the true IP address of the client?

- Look into the X-Forwarded-For header in the backend(Correct)
- Look into the client's cookie
- Look into the X-Forwarded-Proto header in the backend
- Modify the front-end of the website so that the users send their IP in the requests

Explanation
Correct option:
Look into the X-Forwarded-For header in the backend
The X-Forwarded-For request header helps you identify the IP address of a client when you use an HTTP or HTTPS load balancer. Because load balancers intercept traffic between clients and servers, your server access logs contain only the IP address of the load balancer. To see the IP address of the client, use the X-Forwarded-For request header. Elastic Load Balancing stores the IP address of the client in the X-Forwarded-For request header and passes the header to your server.
Incorrect options:
Modify the front-end of the website so that the users send their IP in the requests - When a user makes a request the IP address is sent with the request to the server and the load balancer intercepts it. There is no need to modify the application.
Look into the X-Forwarded-Proto header in the backend - The X-Forwarded-Proto request header helps you identify the protocol (HTTP or HTTPS) that a client used to connect to your load balancer.
Look into the client's cookie - For this, we would need to modify the client-side logic and server-side logic, which would not be efficient.
Reference:
https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/x-forwarded-headers.html

Question 23:

You are a DynamoDB developer for an aerospace company that requires you to write 6 objects per second of 4.5KB in size each.

What write capacity unit is needed for your project?

- 46
- 15
- 30(Correct)
- 24

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.

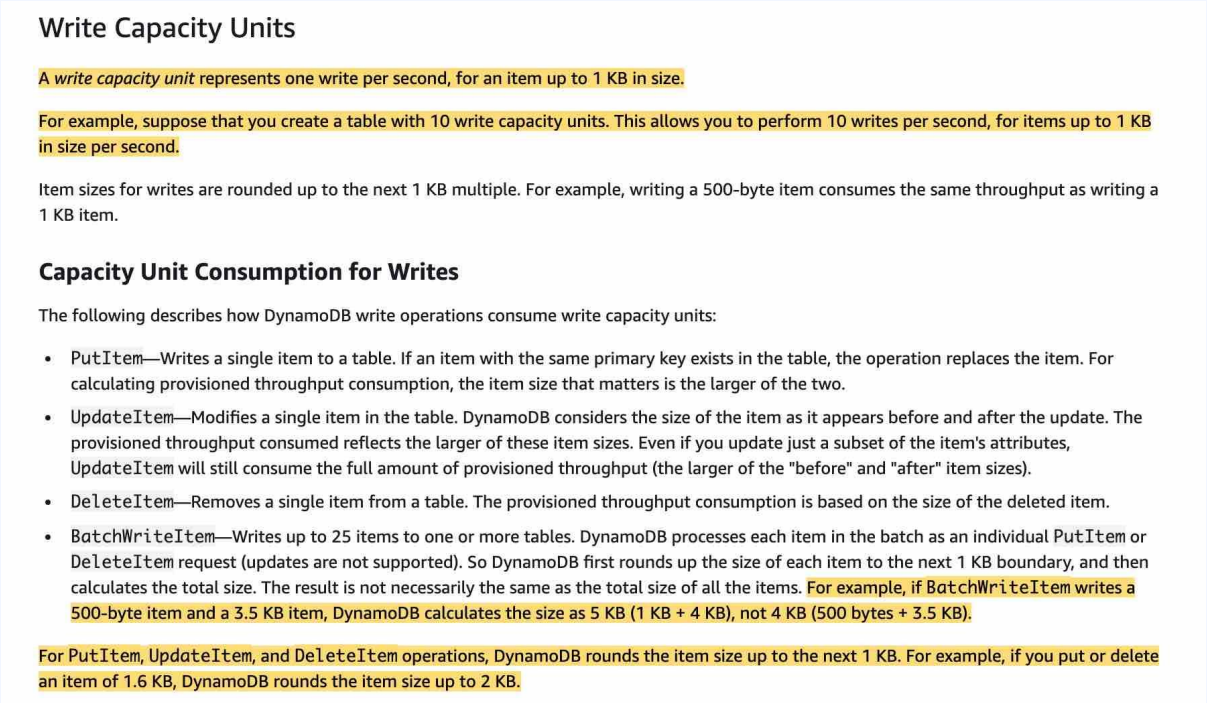
Item sizes for reads are rounded up to the next 4 KB multiple. For example, reading a 3,500-byte item consumes the same throughput as reading a 4 KB item.

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

30

A write capacity unit represents one write per second, for an item up to 1 KB in size.

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. So, for the given use-case, each object is of size 4.5 KB, which will be rounded up to 5KB.

Therefore, for 6 objects, you need 6x5 = 30 WCUs.

Incorrect options:

24

15

46

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

Reference:

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

Question 24:
A company stores confidential data on an Amazon Simple Storage Service (S3) bucket. New regulatory 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?

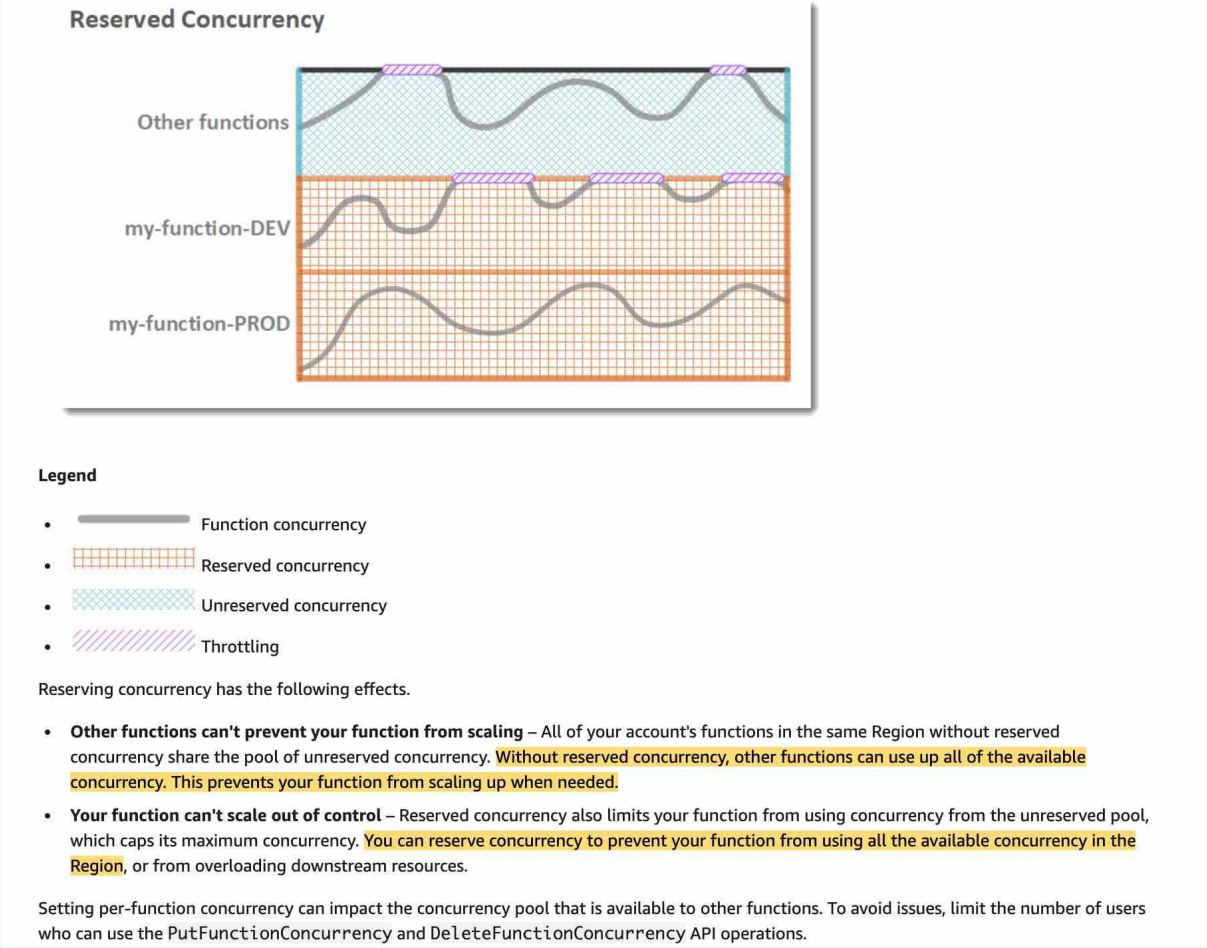
- SSE-KMS
- SSE-C
- SSE-S3(Correct)
- Client Side Encryption

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 25:
The development team at an IT company has configured an Application Load Balancer (ALB) with a Lambda function A as the target but the Lambda function A is not able to process any request from the ALB. Upon investigation, the team finds that there is another Lambda function B in the AWS account that is exceeding the concurrency limits.
How can the development team address this issue?

- Use a Cloudfront Distribution instead of an Application Load Balancer (ALB) for Lambda function A
- Use an API Gateway instead of an Application Load Balancer (ALB) for Lambda function A
- Set up provisioned concurrency for the Lambda function B so that it throttles if it goes above a certain concurrency limit
- Set up reserved concurrency for the Lambda function B so that it throttles if it goes above a certain concurrency limit(Correct)

Explanation
Correct option:
Set up reserved concurrency for the Lambda function B so that it throttles if it goes above a certain concurrency limit
Concurrency is the number of requests that a Lambda function is serving at any given time. If a Lambda function is invoked again while a request is still being processed, another instance is allocated, which increases the function's concurrency.
To ensure that a function can always reach a certain level of concurrency, you can configure the function with reserved concurrency. When a function has reserved concurrency, no other function can use that concurrency. More importantly, reserved concurrency also limits the maximum concurrency for the function, and applies to the function as a whole, including versions and aliases.
Please review this note to understand how reserved concurrency works:



via - https://docs.aws.amazon.com/lambda/latest/dg/configuration-concurrency.html#configuration-concurrency-reserved

Therefore using reserved concurrency for Lambda function B would limit its maximum concurrency and allow Lambda function A to execute without getting throttled.

Incorrect options:

Set up provisioned concurrency for the Lambda function B so that it throttles if it goes above a certain concurrency limit - You should use provisioned concurrency to enable your function to scale without fluctuations in latency. By allocating provisioned concurrency before an increase in invocations, you can ensure that all requests are served by initialized instances with very low latency. Provisioned concurrency is not used to limit the maximum concurrency for a given Lambda function, so this option is incorrect.

Use an API Gateway instead of an Application Load Balancer (ALB) for Lambda function A - This has been added as a distractor as using an API Gateway for Lambda function A has no bearing on limiting the concurrency of Lambda function B, so this option is incorrect.

Use a Cloudfront Distribution instead of an Application Load Balancer (ALB) for Lambda function A - When you associate a CloudFront distribution with a Lambda function (known as Lambda@Edge), CloudFront intercepts requests and responses at CloudFront edge locations and runs the function. Again, this has no bearing on limiting the concurrency of Lambda function B, so this option is incorrect.

References:

https://docs.aws.amazon.com/lambda/latest/dg/configuration-concurrency.html#configuration-concurrency-reserved
https://aws.amazon.com/blogs/networking-and-content-delivery/lambda-functions-as-targets-for-application-load-balancers/

Question 26:
A development team has noticed that one of the EC2 instances has been wrongly configured with the 'DeleteOnTermination' attribute set to True for its root EBS volume.
As a developer associate, can you suggest a way to disable this flag while the instance is still running?

- The attribute cannot be updated when the instance is running. Stop the instance from Amazon EC2 console and then update the flag
- Update the attribute using AWS management console. Select the EC2 instance and then uncheck the Delete On Termination check box for the root EBS volume
- Set the DisableApiTermination attribute of the instance using the API
- Set the DeleteOnTermination attribute to False using the command line(Correct)

Explanation
Correct option:
When an instance terminates, the value of the DeleteOnTermination attribute for each attached EBS volume determines whether to preserve or delete the volume. By default, the DeleteOnTermination attribute is set to True for the root volume and is set to False for all other volume types.
Set the DeleteOnTermination attribute to False using the command line - If the instance is already running, you can set DeleteOnTermination to False using the command line.
Incorrect options:
Update the attribute using AWS management console. Select the EC2 instance and then uncheck the Delete On Termination check box for the root EBS volume - You can set the DeleteOnTermination attribute to False when you launch a new instance. It is not possible to update this attribute of a running instance from the AWS console.
Set the DisableApiTermination attribute of the instance using the API - By default, you can terminate your instance using the Amazon EC2 console, command-line interface, or API. To prevent your instance from being accidentally terminated using Amazon EC2, you can enable termination protection for the instance. The DisableApiTermination attribute controls whether the instance can be terminated using the console, CLI, or API. This option cannot be used to control the delete status for the EBS volume when the instance terminates.
The attribute cannot be updated when the instance is running. Stop the instance from Amazon EC2 console and then update the flag - This statement is wrong and given only as a distractor.
References:
https://aws.amazon.com/premiumsupport/knowledge-center/deleteontermination-ebs/
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/terminating-instances.html#delete-on-termination-running-instance

Question 27:
A retail company manages its IT infrastructure on AWS Cloud via Elastic Beanstalk. The development team at the company is planning to deploy the next version with MINIMUM application downtime and the ability to rollback quickly in case deployment goes wrong.
As a Developer Associate, which of the following options would you recommend to the development team?

- Deploy the new application version using 'Rolling with additional batch' deployment policy
- Deploy the new application version using 'All at once' deployment policy
- Deploy the new application version using 'Rolling' deployment policy
- Deploy the new version to a separate environment via Blue/Green Deployment, and then swap Route 53 records of the two environments to redirect traffic to the new version(Correct)

Explanation
Correct option:
Deploy the new version to a separate environment via Blue/Green Deployment, and then swap Route 53 records of the two environments to redirect traffic to the new version
With deployment policies such as 'All at once', AWS Elastic Beanstalk performs an in-place update when you update your application versions and your application can become unavailable to users for a short period of time. You can avoid this downtime by performing a blue/green deployment, where you deploy the new version to a separate environment, and then swap CNAMEs (via Route 53) of the two environments to redirect traffic to the new version instantly. In case of any deployment issues, the rollback process is very quick via swapping the URLs for the two environments.

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:

Deploy the new application version using 'All at once' deployment policy - Although 'All at once' is the quickest deployment method, but the application may become unavailable to users (or have low availability) for a short time. So this option is not correct.

Deploy the new application version using 'Rolling' deployment policy - This policy avoids downtime and minimizes reduced availability, at a cost of a longer deployment time. However rollback process is via manual redeploy, so it's not as quick as the Blue/Green deployment.

Deploy the new application version using 'Rolling with additional batch' deployment policy - This policy 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. However rollback process is via manual redeploy, so it's not as quick as the Blue/Green deployment.

Reference:

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

Question 28:

As a site reliability engineer, you are responsible for improving the company's deployment by scaling and automating applications. As new application versions are ready for production you ensure that the application gets deployed to different sets of EC2 instances at different times allowing for a smooth transition.

Using AWS CodeDeploy, which of the following options will allow you to do this?

- Define multiple CodeDeploy Applications
- CodeDeploy Deployment Groups(Correct)
- CodeDeploy Agent
- CodeDeploy Hooks

Explanation

Correct option:

CodeDeploy Deployment Groups

You can specify one or more deployment groups for a CodeDeploy application. The deployment group contains settings and configurations used during the deployment. Most deployment group settings depend on the compute platform used by your application. Some settings, such as rollbacks, triggers, and alarms can be configured for deployment groups for any compute platform.

In an EC2/On-Premises deployment, a deployment group is a set of individual instances targeted for deployment. A deployment group contains individually tagged instances, Amazon EC2 instances in Amazon EC2 Auto Scaling groups, or both.

Incorrect options:

CodeDeploy Agent - The CodeDeploy agent is a software package that, when installed and configured on an instance, makes it possible for that instance to be used in CodeDeploy deployments. The agent connects the EC2 instances to the CodeDeploy service.

CodeDeploy Hooks - Hooks are found in the AppSec file used by AWS CodeDeploy to manage deployment. Hooks correspond to lifecycle events such as ApplicationStart, ApplicationStop, etc. to which you can assign a script.

Define multiple CodeDeploy Applications - This option has been added as a distractor. Instead, you want to use deployment groups to use the same deployment and maybe separate the times when a group of instances receives the software updates.

Reference:

https://docs.aws.amazon.com/codedeploy/latest/userguide/deployment-groups.html

Question 29:
A multi-national company runs its technology operations on AWS Cloud. As part of their storage solution, they use a large number of EBS volumes, with AWS Config and CloudTrail activated. A manager has tried to find the user name that created an EBS volume by searching CloudTrail events logs but wasn't successful.
As a Developer Associate, which of the following would you recommend as the correct solution?

- EBS volume status checks are disabled
- Amazon EBS CloudWatch metrics are disabled
- AWS CloudTrail event logs for 'ManageVolume' aren't available for EBS volumes created during an Amazon EC2 launch
- AWS CloudTrail event logs for 'CreateVolume' aren't available for EBS volumes created during an Amazon EC2 launch(Correct)

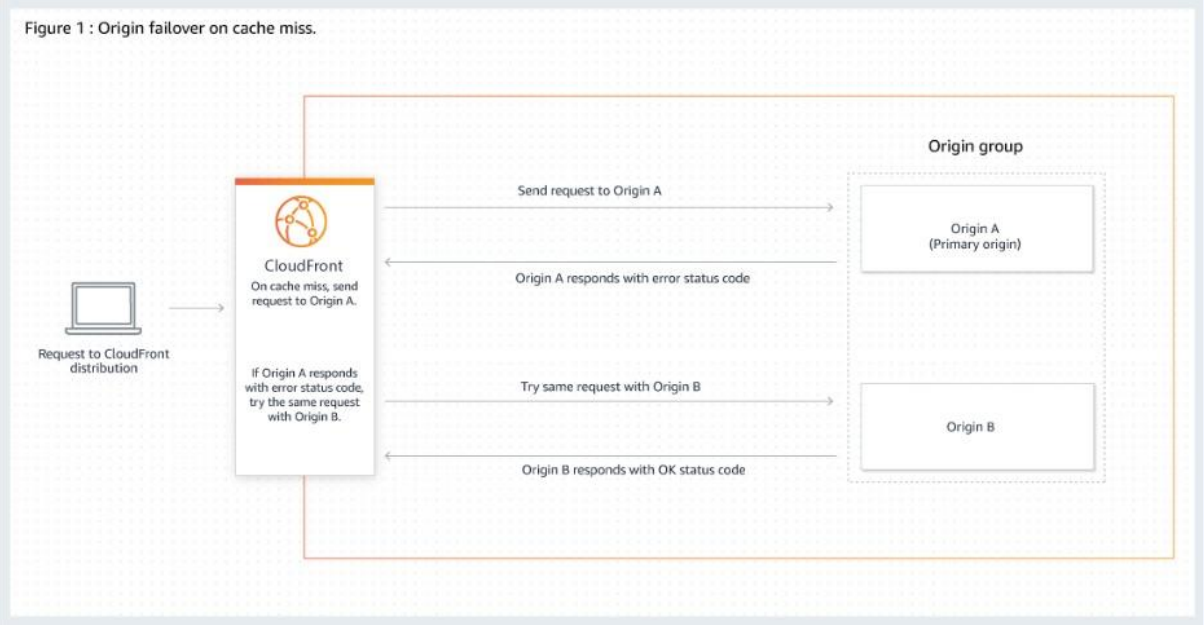
Explanation
Correct option:
AWS CloudTrail event logs for 'CreateVolume' aren't available for EBS volumes created during an Amazon EC2 launch - AWS CloudTrail event logs for 'CreateVolume' aren't available for EBS volumes created during an Amazon Elastic Compute Cloud (Amazon EC2) launch.
Incorrect options:
AWS CloudTrail event logs for 'ManageVolume' aren't available for EBS volumes created during an Amazon EC2 launch - Event 'ManageVolume' is a made-up option and has been added as a distractor.
Amazon EBS CloudWatch metrics are disabled - Amazon Elastic Block Store (Amazon EBS) sends data points to CloudWatch for several metrics. Data is only reported to CloudWatch when the volume is attached to an instance. CloudWatch metrics are useful in tracking the status or life cycle changes of an EBS volume, they are not useful in knowing about the metadata of EBS volumes.
EBS volume status checks are disabled - Volume status checks enable you to better understand, track and manage potential inconsistencies in the data on an Amazon EBS volume. They are designed to provide you with the information that you need to determine whether your Amazon EBS volumes are impaired, and to help you control how a potentially inconsistent volume is handled. Our current use case requires us to pull data about EBS volume metadata, which is not possible with this feature.
References:
https://aws.amazon.com/premiumsupport/knowledge-center/find-ebs-user-config-cloudtrail/
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using_cloudwatch_ebs.html
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/monitoring-volume-status.html

Question 30:
A video streaming application uses Amazon CloudFront for its data distribution. The development team has decided to use CloudFront with origin failover for high availability.
Which of the following options are correct while configuring CloudFront with Origin Groups? (Select two)

- To set up origin failover, you must have a distribution with at least three origins
- CloudFront fails over to the secondary origin only when the HTTP method of the viewer request is GET, HEAD or OPTIONS(Correct)
- CloudFront routes all incoming requests to the primary origin, even when a previous request failed over to the secondary origin(Correct)
- When there's a cache hit, CloudFront routes the request to the primary origin in the origin group
- In the Origin Group of your distribution, all the origins are defined as primary for automatic failover in case an origin fails

Explanation
Correct options:
CloudFront routes all incoming requests to the primary origin, even when a previous request failed over to the secondary origin
CloudFront routes all incoming requests to the primary origin, even when a previous request failed over to the secondary origin. CloudFront only sends requests to the secondary origin after a request to the primary origin fails.
CloudFront fails over to the secondary origin only when the HTTP method of the viewer request is GET, HEAD or OPTIONS
CloudFront fails over to the secondary origin only when the HTTP method of the viewer request is GET, HEAD, or OPTIONS. CloudFront does not failover when the viewer sends a different HTTP method (for example POST, PUT, and so on).

How origin failover works:



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

Incorrect options:

When there's a cache hit, CloudFront routes the request to the primary origin in the origin group - When there's a cache miss, CloudFront routes the request to the primary origin in the origin group.

When there's a cache hit, CloudFront returns the requested file.

To set up origin failover, you must have a distribution with at least three origins - Two origins are enough to set up an origin failover.

In the Origin Group of your distribution, all the origins are defined as primary for automatic failover in case an origin fails - To set up origin failover, you must have a distribution with at least two origins. Next, you create an origin group for your distribution that includes two origins, setting one as the primary. Only one origin can be set as primary.

Reference:

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

Question 31:
A developer is configuring the redirect actions for an Application Load Balancer. The developer stumbled upon the following snippet of code.
Which of the following is an example of a query string condition that the developer can use on AWS CLI?

```
[
 {
    "Field": "query-string",
    "QueryStringConfig": {
       "Values": [
        {
           "Key": "version",
           "Value": "v1"
        },
        {
           "Value": "*example*"
        }
       ]
    }
 }
]
```

(Correct)

```
[
 {
    "Field": "query-string",
    "StringHeaderConfig": {
       "Values": ["*.example.com"]
    }
 }
]
```

```
[
 {
    "Field": "query-string",
    "PathPatternConfig": {
       "Values": ["/img/*"]
    }
 }
]
```

```
[
```

```
 {
    "Type": "redirect",
    "RedirectConfig": {
       "Protocol": "HTTPS",
       "Port": "443",
       "Host": "#{host}",
       "Path": "/#{path}",
       "Query": "#{query}",
       "StatusCode": "HTTP_301"
    }
 }
]
```

Explanation
Correct option:
**

```
[
 {
    "Field": "query-string",
    "QueryStringConfig": {
       "Values": [
        {
           "Key": "version",
           "Value": "v1"
        },
        {
           "Value": "*example*"
        }
       ]
    }
 }
]
```

**
You can use query string conditions to configure rules that route requests based on key/value pairs or values in the query string. The match evaluation is not case-sensitive. The following wildcard characters are supported: * (matches 0 or more characters) and ? (matches exactly 1 character). You can specify conditions when you create or modify a rule.
Query parameters are often used along with the path component of the URL for applying a special logic to the resource being fetched.

The query string component starts after the first "?" in a URI. Typically query strings contain key-value pairs separated by a delimiter "&". Example: http://example.com/path/to/page?version=A&gender=female
The example condition given in the question is satisfied by requests with a query string that includes either a key/value pair of "version=v1" or any key set to "example".
Incorrect options:
**

```
[
 {
    "Field": "query-string",
    "PathPatternConfig": {
       "Values": ["/img/*"]
    }
 }
]
```

**
**

```
[
 {
    "Field": "query-string",
    "StringHeaderConfig": {
       "Values": ["*.example.com"]
    }
 }
]
```

**
These two options are malformed and are incorrect.
**

```
[
 {
    "Type": "redirect",
    "RedirectConfig": {
       "Protocol": "HTTPS",
       "Port": "443",
       "Host": "#{host}",
       "Path": "/#{path}",
       "Query": "#{query}",
       "StatusCode": "HTTP_301"
    }
 }
]
```

** - This action redirects an HTTP request to an HTTPS request on port 443, with the same hostname, path, and query string as the HTTP request.
Reference:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-listeners.html#query-string-conditions

Question 32:
Your company has a load balancer in a VPC configured to be internet facing. The public DNS name assigned to the load balancer is myDns-1234567890.us-east-1.elb.amazonaws.com. When your client applications first load they capture the load balancer DNS name and then resolve the IP address for the load balancer so that they can directly reference the underlying IP.
It is observed that the client applications work well but unexpectedly stop working after a while.
What is the reason for this?

- You need to disable multi-AZ deployments
- The load balancer is highly available and its public IP may change. The DNS name is constant(Correct)
- Your security groups are not stable
- You need to enable stickiness

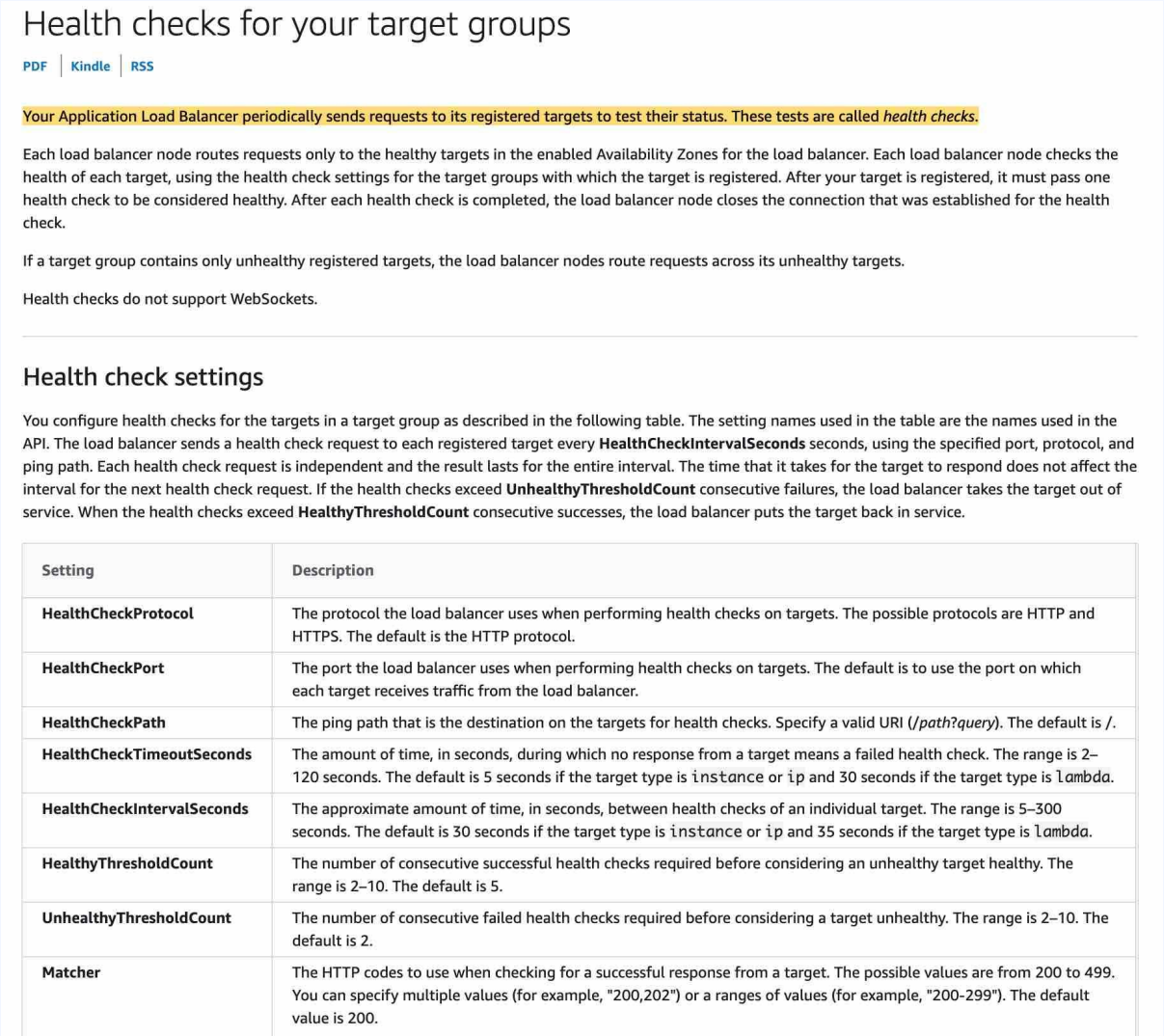
Explanation
Correct option:
The load balancer is highly available and its public IP may change. The DNS name is constant
When your load balancer is created, it receives a public DNS name that clients can use to send requests. The DNS servers resolve the DNS name of your load balancer to the public IP addresses of the load balancer nodes for your load balancer. Never resolve the IP of a load balancer as it can change with time. You should always use the DNS name.
Incorrect options:
Your security groups are not stable - You security groups to allow your load balancer to work with registered instances. It is stable if set correctly. If your application is working and stops after a while, the issue is not with the security groups.
You need to enable stickiness - This enables the load balancer to bind a user's session to a specific instance, so this has no impact on the issue described in the given use-case.
You need to disable multi-AZ deployments - This has been added as a distractor and this has no bearing on the use-case. The change is happening with the IP of the load balancer.
Reference:
https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-internet-facing-load-balancers.html

Question 33:
A development team has inherited a web application running in the us-east-1 region with three availability zones (us-east-1a, us-east1-b, and us-east-1c) whose incoming web traffic is routed by a load balancer. When one of the EC2 instances hosting the web application crashes, the team realizes that the load balancer continues to route traffic to that instance causing intermittent issues.
Which of the following should the development team do to minimize this problem?

- Enable SSL
- Enable Multi AZ deployments
- Enable Stickiness
- Enable Health Checks(Correct)

Explanation
Correct option:
Enable Health Checks
To discover the availability of your EC2 instances, a load balancer periodically sends pings, attempts connections, or sends requests to test the EC2 instances. These tests are called health checks. The status of the instances that are healthy at the time of the health check is InService. The status of any instances that are unhealthy at the time of the health check is OutOfService.

Load Balancer Health Checks:



Health checks for your target groups

PDF | Kindle | RSS

Your Application Load Balancer periodically sends requests to its registered targets to test their status. These tests are called *health checks*.

Each load balancer node routes requests only to the healthy targets in the enabled Availability Zones for the load balancer. Each load balancer node checks the health of each target, using the health check settings for the target groups with which the target is registered. After your target is registered, it must pass one health check to be considered healthy. After each health check is completed, the load balancer node closes the connection that was established for the health check.

If a target group contains only unhealthy registered targets, the load balancer nodes route requests across its unhealthy targets.

Health checks do not support WebSockets.

Health check settings

You configure health checks for the targets in a target group as described in the following table. The setting names used in the table are the names used in the API. The load balancer sends a health check request to each registered target every **HealthCheckIntervalSeconds** seconds, using the specified port, protocol, and ping path. Each health check request is independent and the result lasts for the entire interval. The time that it takes for the target to respond does not affect the interval for the next health check request. If the health checks exceed **UnhealthyThresholdCount** consecutive failures, the load balancer takes the target out of service. When the health checks exceed **HealthyThresholdCount** consecutive successes, the load balancer puts the target back in service.

Setting	Description
HealthCheckProtocol	The protocol the load balancer uses when performing health checks on targets. The possible protocols are HTTP and HTTPS. The default is the HTTP protocol.
HealthCheckPort	The port the load balancer uses when performing health checks on targets. The default is to use the port on which each target receives traffic from the load balancer.
HealthCheckPath	The ping path that is the destination on the targets for health checks. Specify a valid URI (*/path?query*). The default is /.
HealthCheckTimeoutSeconds	The amount of time, in seconds, during which no response from a target means a failed health check. The range is 2–120 seconds. The default is 5 seconds if the target type is `instance` or `ip` and 30 seconds if the target type is `lambda`.
HealthCheckIntervalSeconds	The approximate amount of time, in seconds, between health checks of an individual target. The range is 5–300 seconds. The default is 30 seconds if the target type is `instance` or `ip` and 35 seconds if the target type is `lambda`.
HealthyThresholdCount	The number of consecutive successful health checks required before considering an unhealthy target healthy. The range is 2–10. The default is 5.
UnhealthyThresholdCount	The number of consecutive failed health checks required before considering a target unhealthy. The range is 2–10. The default is 2.
Matcher	The HTTP codes to use when checking for a successful response from a target. The possible values are from 200 to 499. You can specify multiple values (for example, "200,202") or a ranges of values (for example, "200-299"). The default value is 200.

via - https://docs.aws.amazon.com/elasticloadbalancing/latest/application/target-group-health-checks.html

Incorrect options:

Enable Stickiness - Stickiness enables the load balancer to bind a user's session to a specific instance, it cannot be used for gauging the health of an instance.

Enable Multi-AZ deployments - It's a good practice to provision instances in more than one availability zone however you still need a way to check the health status of the instances, so this option is incorrect.

Enable SSL - This option has been added as a distractor. SSL encrypts the transmission of data between a web server and a browser.

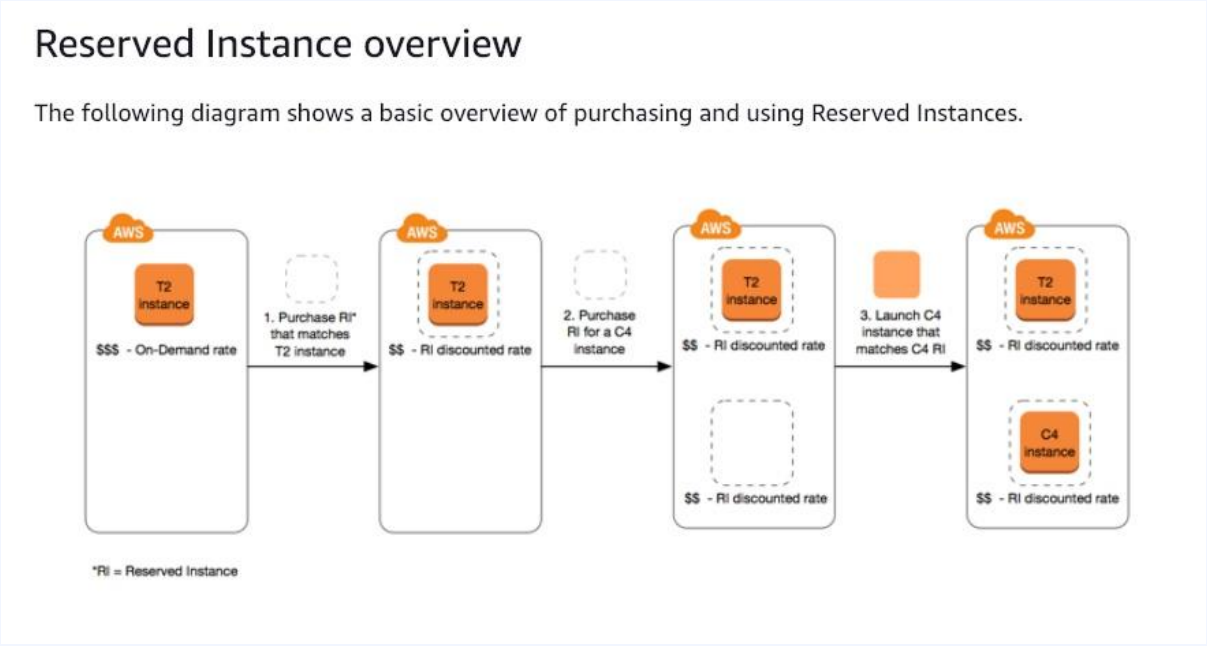
Reference:

https://docs.aws.amazon.com/elasticloadbalancing/latest/application/target-group-health-checks.html

Question 34:
Your team has just signed up an year-long contract with a client maintaining a three-tier web application, that needs to be moved to AWS Cloud. The application has steady traffic throughout the day and needs to be on a reliable system with no down-time or access issues. The solution needs to be cost-optimal for this startup.
Which of the following options should you choose?

- On-premise EC2 instance
- Amazon EC2 Spot Instances
- Amazon EC2 On Demand Instances
- Amazon EC2 Reserved Instances(Correct)

Explanation
Correct option:
Amazon EC2 Reserved Instances - Reserved instances can provide a capacity reservation, offering additional confidence in your ability to launch the number of instances you have reserved when you need them. You save money going with Reserved instances vs on-demand especially in a year's worth of time.
Reserved Instances are not physical instances, but rather a billing discount applied to the use of On-Demand Instances in your account. These On-Demand Instances must match certain attributes, such as instance type and Region, to benefit from the billing discount. So, there is no performance difference between an On-Demand instance or a Reserved instance.
How RIs work:



via - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-reserved-instances.html
Incorrect options:
Amazon EC2 Spot Instances - A Spot Instance is an unused EC2 instance that is available for less than the On-Demand price. Because Spot Instances enable you to request unused EC2 instances at steep discounts, you can lower your Amazon EC2 costs significantly. Spot instances are useful if your applications can be interrupted, like data analysis, batch jobs, background processing, and optional

tasks. Spot instances can be pulled down anytime without prior notice. Hence, not the right choice for the current scenario.

Amazon EC2 On-Demand Instances - With On-Demand Instances, you pay for compute capacity by the second with no long-term commitments. You have full control over its lifecycle—you decide when to launch, stop, hibernate, start, reboot, or terminate it. But, On-Demand instances cost a lot more than Reserved instances. Here, in our use case, we already know that the systems are required for a complete year, so making use of Reserved Instances discount makes a lot more sense.

On-premise EC2 instance - On-premise implies the client has to maintain the physical machines, their capacity provisioning and maintenance. Not an option when the client is planning to move to AWS Cloud.

References:

https://aws.amazon.com/ec2/pricing/reserved-instances/

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-reserved-instances.html

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

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-on-demand-instances.html

Question 35:

A Company uses a large set of EBS volumes for their fleet of Amazon EC2 instances. As an AWS Certified Developer Associate, your help has been requested to understand the security features of the EBS volumes. The company does not want to build or maintain their own encryption key management infrastructure.

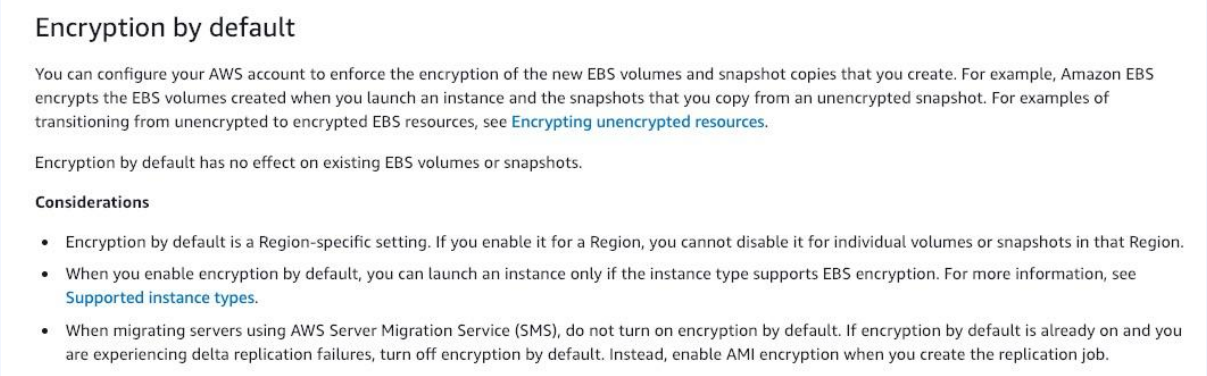
Can you help them understand what works for Amazon EBS encryption? (Select two)

- A volume restored from an encrypted snapshot, or a copy of an encrypted snapshot is always encrypted(Correct)
- Encryption by default is a Region-specific setting. If you enable it for a Region, you cannot disable it for individual volumes or snapshots in that Region(Correct)
- A snapshot of an encrypted volume can be encrypted or unencrypted
- Encryption by default is an AZ specific setting. If you enable it for an AZ, you cannot disable it for individual volumes or snapshots in that AZ
- You can encrypt an existing unencrypted volume or snapshot by using AWS Key Management Service (KMS) AWS SDKs

Explanation

Correct option:

Encryption by default is a Region-specific setting. If you enable it for a Region, you cannot disable it for individual volumes or snapshots in that Region - You can configure your AWS account to enforce the encryption of the new EBS volumes and snapshot copies that you create. Encryption by default is a Region-specific setting. If you enable it for a Region, you cannot disable it for individual volumes or snapshots in that Region.



Encryption by default

You can configure your AWS account to enforce the encryption of the new EBS volumes and snapshot copies that you create. For example, Amazon EBS encrypts the EBS volumes created when you launch an instance and the snapshots that you copy from an unencrypted snapshot. For examples of transitioning from unencrypted to encrypted EBS resources, see Encrypting unencrypted resources.

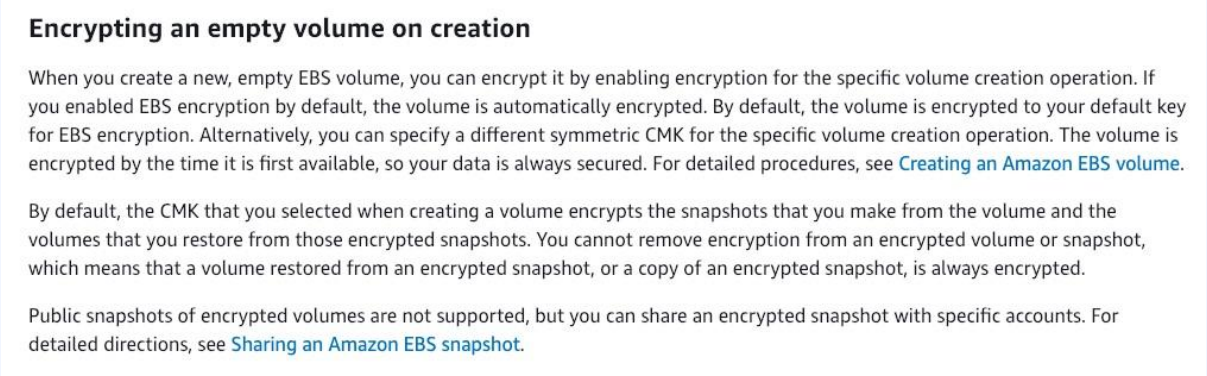
Encryption by default has no effect on existing EBS volumes or snapshots.

Considerations

- Encryption by default is a Region-specific setting. If you enable it for a Region, you cannot disable it for individual volumes or snapshots in that Region.
- When you enable encryption by default, you can launch an instance only if the instance type supports EBS encryption. For more information, see Supported instance types.
- When migrating servers using AWS Server Migration Service (SMS), do not turn on encryption by default. If encryption by default is already on and you are experiencing delta replication failures, turn off encryption by default. Instead, enable AMI encryption when you create the replication job.

via - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html

A volume restored from an encrypted snapshot, or a copy of an encrypted snapshot, is always encrypted - By default, the CMK that you selected when creating a volume encrypts the snapshots that you make from the volume and the volumes that you restore from those encrypted snapshots. You cannot remove encryption from an encrypted volume or snapshot, which means that a volume restored from an encrypted snapshot, or a copy of an encrypted snapshot is always encrypted.



Encrypting an empty volume on creation

When you create a new, empty EBS volume, you can encrypt it by enabling encryption for the specific volume creation operation. If you enabled EBS encryption by default, the volume is automatically encrypted. By default, the volume is encrypted to your default key for EBS encryption. Alternatively, you can specify a different symmetric CMK for the specific volume creation operation. The volume is encrypted by the time it is first available, so your data is always secured. For detailed procedures, see Creating an Amazon EBS volume.

By default, the CMK that you selected when creating a volume encrypts the snapshots that you make from the volume and the volumes that you restore from those encrypted snapshots. You cannot remove encryption from an encrypted volume or snapshot, which means that a volume restored from an encrypted snapshot, or a copy of an encrypted snapshot, is always encrypted.

Public snapshots of encrypted volumes are not supported, but you can share an encrypted snapshot with specific accounts. For detailed directions, see Sharing an Amazon EBS snapshot.

via - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html

Incorrect options:

You can encrypt an existing unencrypted volume or snapshot by using AWS Key Management Service (KMS) AWS SDKs - This is an incorrect statement. There is no direct way to encrypt an existing unencrypted volume or snapshot. You can encrypt an unencrypted snapshot by copying and enabling encryption while copying the snapshot. To encrypt an EBS volume, you need to create a snapshot and then encrypt the snapshot as described earlier. From this new encrypted snapshot, you can then create an encrypted volume.

A snapshot of an encrypted volume can be encrypted or unencrypted - This is an incorrect statement. You cannot remove encryption from an encrypted volume or snapshot, which means that a volume restored from an encrypted snapshot, or a copy of an encrypted snapshot is always encrypted.

Encryption by default is an AZ specific setting. If you enable it for an AZ, you cannot disable it for individual volumes or snapshots in that AZ - This is an incorrect statement. Encryption by default is a Region-specific setting. If you enable it for a Region, you cannot disable it for individual volumes or snapshots in that Region.

References:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSEncryption.html#encrypt-unencrypted

Question 36:
As a site reliability engineer, you work on building and running large-scale, distributed, fault-tolerant systems in the cloud using automation. You have just replaced the company's Jenkins based CI/CD platform with AWS CodeBuild and would like to programmatically define your build steps.
Which of the following options should you choose?

- Define a buildspec.yml file in the codebuild/ directory
- Define an appspec.yml file in the codebuild/ directory
- Define an appspec.yml file in the root directory
- Define a buildspec.yml file in the root directory(Correct)

Explanation
Correct option:
Define a buildspec.yml file in the root directory
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.
A build spec is a collection of build commands and related settings, in YAML format, that AWS CodeBuild uses to run a build. You can include a build spec as part of the source code or you can define a build spec when you create a build project.

Buildspec file name and storage location

If you include a buildspec as part of the source code, by default, the buildspec file must be named `buildspec.yml` and placed in the root of your source directory.

You can override the default buildspec file name and location. For example, you can:

- Use a different buildspec file for different builds in the same repository, such as `buildspec_debug.yml` and `buildspec_release.yml`.
- Store a buildspec file somewhere other than the root of your source directory, such as `config/buildspec.yml` or in an S3 bucket. The S3 bucket must be in the same AWS Region as your build project. Specify the buildspec file using its ARN (for example, `arn:aws:s3:::my-codebuild-sample2/buildspec.yml`).

You can specify only one buildspec for a build project, regardless of the buildspec file's name.

To override the default buildspec file name, location, or both, do one of the following:

- Run the AWS CLI `create-project` or `update-project` command, setting the `buildspec` value to the path to the alternate buildspec file relative to the value of the built-in environment variable `CODEBUILD_SRC_DIR`. You can also do the equivalent with the `create project` operation in the AWS SDKs. For more information, see Create a build project or Change a build project's settings.
- Run the AWS CLI `start-build` command, setting the `buildspecOverride` value to the path to the alternate buildspec file relative to the value of the built-in environment variable `CODEBUILD_SRC_DIR`. You can also do the equivalent with the `start build` operation in the AWS SDKs. For more information, see Run a build.
- In an AWS CloudFormation template, set the `BuildSpec` property of `Source` in a resource of type `AWS::CodeBuild::Project` to the path to the alternate buildspec file relative to the value of the built-in environment variable `CODEBUILD_SRC_DIR`. For more information, see the BuildSpec property in AWS CodeBuild project source in the *AWS CloudFormation User Guide*.

via - https://docs.aws.amazon.com/codebuild/latest/userguide/build-spec-ref.html

Incorrect options:

Define an appspec.yml file in the root directory - The AppSpec file is used for deployment in the CodeDeploy service.

Define a buildspec.yml file in the codebuild/ directory - The file is correct but must be in the root directory.

Define an appspec.yml file in the codebuild/ directory - The AppSpec file is used for deployment in the CodeDeploy service.

Reference:

https://docs.aws.amazon.com/codebuild/latest/userguide/build-spec-ref.html

Question 37:

An e-commerce company has a fleet of EC2 based web servers running into very high CPU utilization issues. The development team has determined that serving secure traffic via HTTPS is a major contributor to the high CPU load.

Which of the following steps can take the high CPU load off the web servers? (Select two)

- Create an HTTP listener on the Application Load Balancer with SSL pass-through
- Create an HTTP listener on the Application Load Balancer with SSL termination
- Create an HTTPS listener on the Application Load Balancer with SSL termination(Correct)
- Create an HTTPS listener on the Application Load Balancer with SSL pass-through
- Configure an SSL/TLS certificate on an Application Load Balancer via AWS Certificate Manager (ACM)(Correct)

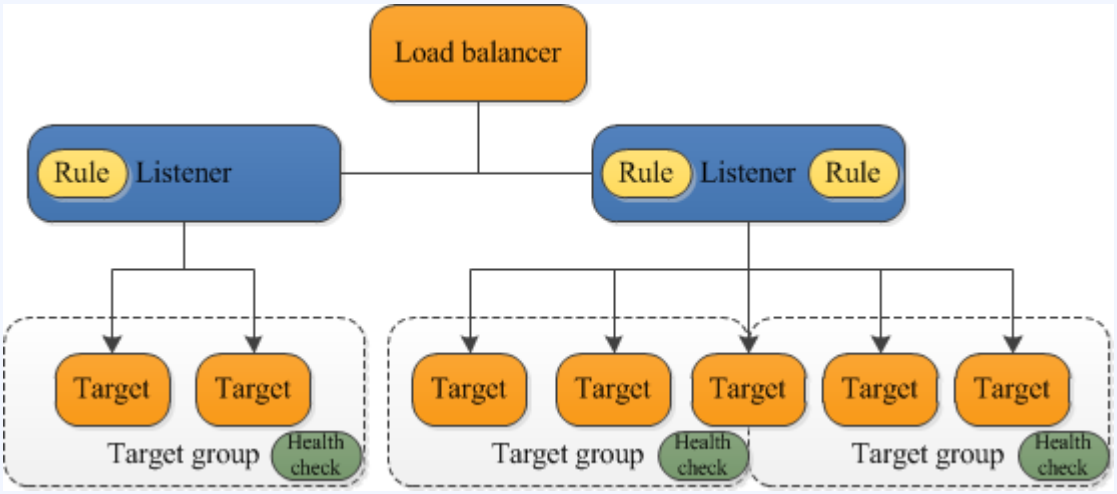
Explanation

Correct option:

"Configure an SSL/TLS certificate on an Application Load Balancer via AWS Certificate Manager (ACM)"

"Create an HTTPS listener on the Application Load Balancer with SSL termination"

An Application load balancer distributes incoming application traffic across multiple targets, such as EC2 instances, in multiple Availability Zones. A listener checks for connection requests from clients, using the protocol and port that you configure. The rules that you define for a listener determine how the load balancer routes requests to its registered targets. Each rule consists of a priority, one or more actions, and one or more conditions.



via - https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html

To use an HTTPS listener, you must deploy at least one SSL/TLS server certificate on your load balancer. You can create an HTTPS listener, which uses encrypted connections (also known as SSL offload). This feature enables traffic encryption between your load balancer and the clients that initiate SSL or TLS sessions. As the EC2 instances are under heavy CPU load, the load balancer will use the server certificate to terminate the front-end connection and then decrypt requests from clients before sending them to the EC2 instances.

Please review this resource to understand how to associate an ACM SSL/TLS certificate with an Application Load Balancer:
https://aws.amazon.com/premiumsupport/knowledge-center/associate-acm-certificate-alb-nlb/
Incorrect options:
"Create an HTTPS listener on the Application Load Balancer with SSL pass-through" - If you use an HTTPS listener with SSL pass-through, then the EC2 instances would continue to be under heavy CPU load as they would still need to decrypt the secure traffic at the instance level. Hence this option is incorrect.
"Create an HTTP listener on the Application Load Balancer with SSL termination"
"Create an HTTP listener on the Application Load Balancer with SSL pass-through"
You cannot have an HTTP listener for an Application Load Balancer to support SSL termination or SSL pass-through, so both these options are incorrect.
References:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html
https://aws.amazon.com/premiumsupport/knowledge-center/associate-acm-certificate-alb-nlb/

Question 38:

A development team has been using Amazon S3 service as an object store. With Amazon S3 turning strongly consistent, the team wants to understand the impact of this change on its data storage practices.

As a developer associate, can you identify the key characteristics of the strongly consistent data model followed by S3? (Select two)

- A process replaces an existing object and immediately tries to read it. Amazon S3 might return the old data
- If you delete a bucket and immediately list all buckets, the deleted bucket might still appear in the list(Correct)
- A process deletes an existing object and immediately tries to read it. Amazon S3 can return data as the object deletion has not yet propagated completely
- A process deletes an existing object and immediately tries to read it. Amazon S3 will not return any data as the object has been deleted(Correct)
- A process deletes an existing object and immediately lists keys within its bucket. The object could still be visible for few more minutes till the change propagates

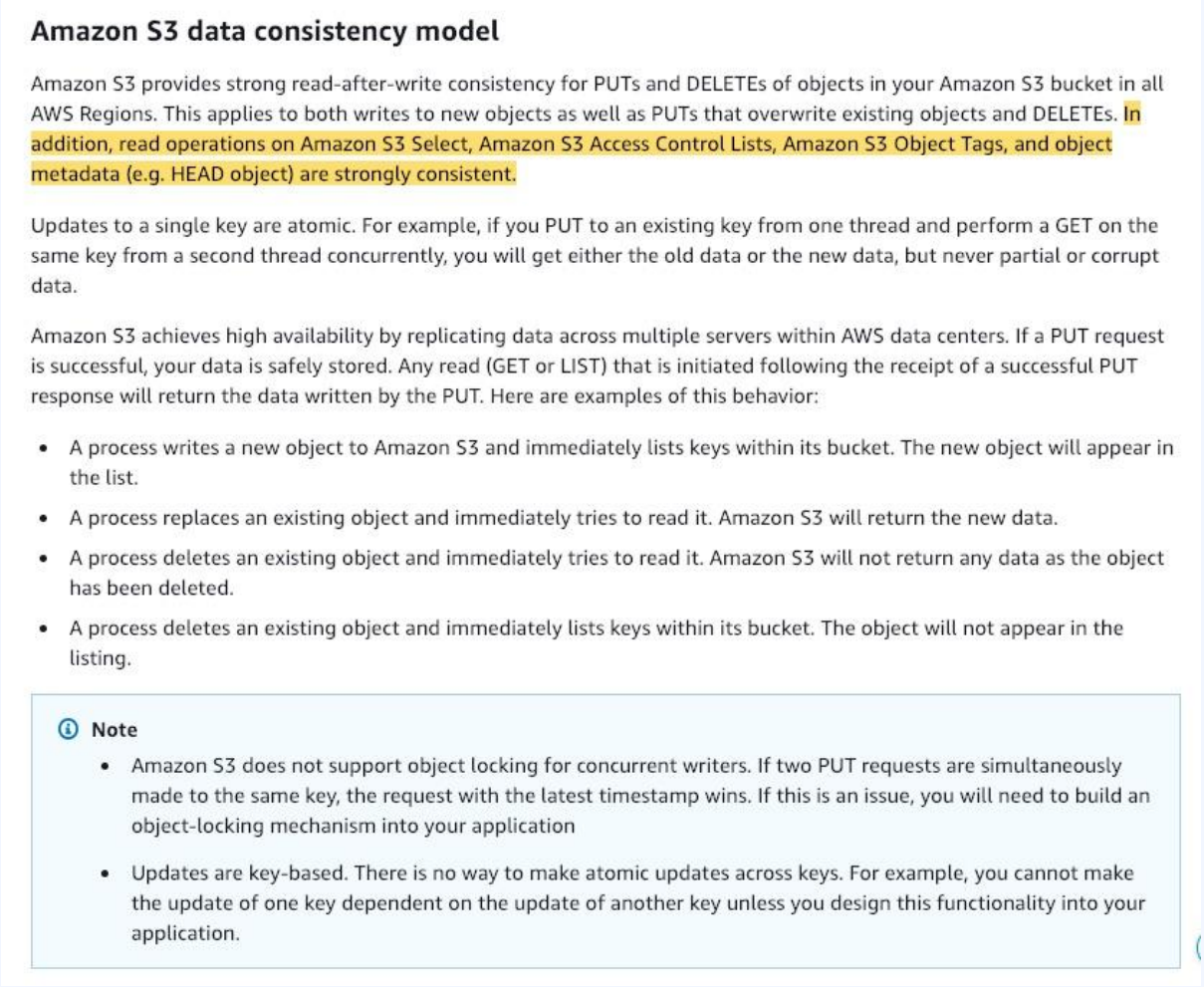
Explanation

Correct options:

If you delete a bucket and immediately list all buckets, the deleted bucket might still appear in the list - Bucket configurations have an eventual consistency model. If you delete a bucket and immediately list all buckets, the deleted bucket might still appear in the list.

A process deletes an existing object and immediately tries to read it. Amazon S3 will not return any data as the object has been deleted - Amazon S3 provides strong read-after-write consistency for PUTs and DELETEs of objects in your Amazon S3 bucket in all AWS Regions. This applies to both writes to new objects as well as PUTs that overwrite existing objects and DELETEs.

Amazon S3 data consistency model:



Amazon S3 data consistency model

Amazon S3 provides strong read-after-write consistency for PUTs and DELETEs of objects in your Amazon S3 bucket in all AWS Regions. This applies to both writes to new objects as well as PUTs that overwrite existing objects and DELETEs. In addition, read operations on Amazon S3 Select, Amazon S3 Access Control Lists, Amazon S3 Object Tags, and object metadata (e.g. HEAD object) are strongly consistent.

Updates to a single key are atomic. For example, if you PUT to an existing key from one thread and perform a GET on the same key from a second thread concurrently, you will get either the old data or the new data, but never partial or corrupt data.

Amazon S3 achieves high availability by replicating data across multiple servers within AWS data centers. If a PUT request is successful, your data is safely stored. Any read (GET or LIST) that is initiated following the receipt of a successful PUT response will return the data written by the PUT. Here are examples of this behavior:

- A process writes a new object to Amazon S3 and immediately lists keys within its bucket. The new object will appear in the list.
- A process replaces an existing object and immediately tries to read it. Amazon S3 will return the new data.
- A process deletes an existing object and immediately tries to read it. Amazon S3 will not return any data as the object has been deleted.
- A process deletes an existing object and immediately lists keys within its bucket. The object will not appear in the listing.

Note

- Amazon S3 does not support object locking for concurrent writers. If two PUT requests are simultaneously made to the same key, the request with the latest timestamp wins. If this is an issue, you will need to build an object-locking mechanism into your application
- Updates are key-based. There is no way to make atomic updates across keys. For example, you cannot make the update of one key dependent on the update of another key unless you design this functionality into your application.

via - https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html#ConsistencyModel

Incorrect options:

A process deletes an existing object and immediately lists keys within its bucket. The object could still be visible for few more minutes till the change propagates

A process deletes an existing object and immediately tries to read it. Amazon S3 can return data as the object deletion has not yet propagated completely -

These two options highlight an eventually consistent behavior. Amazon S3 is now strongly consistent and will not return any data as the object has been deleted. So both these options are incorrect.

A process replaces an existing object and immediately tries to read it. Amazon S3 might return the old data - Amazon S3 will return the new data.

Reference:

https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html#ConsistencyModel

Question 39:
A video encoding application running on an EC2 instance takes about 20 seconds on average to process each raw footage file. The application picks the new job messages from an SQS queue. The development team needs to account for the use-case when the video encoding process takes longer than usual so that the same raw footage is not processed by multiple consumers.
As a Developer Associate, which of the following solutions would you recommend to address this use-case?

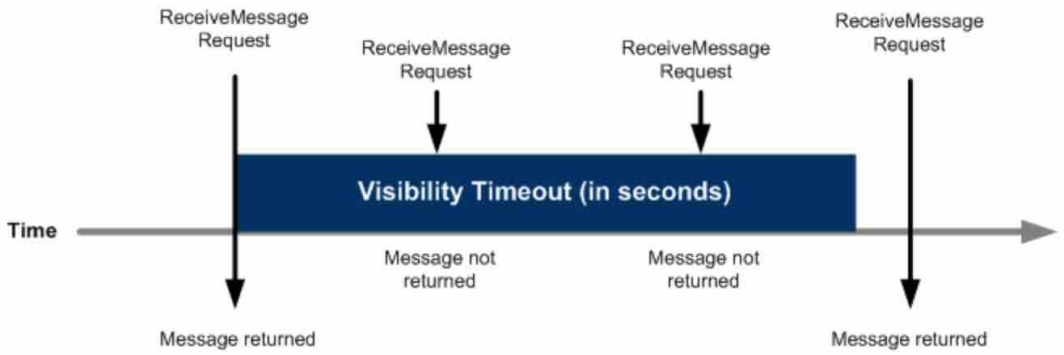
- Use DelaySeconds action to delay a message's visibility timeout
- Use WaitTimeSeconds action to long poll and extend a message's visibility timeout
- Use ChangeMessageVisibility action to extend a message's visibility timeout(Correct)
- Use WaitTimeSeconds action to short poll and extend a message's visibility timeout

Explanation
Correct option:
Use ChangeMessageVisibility action to extend a message's visibility timeout
Amazon SQS uses a visibility timeout to prevent other consumers from receiving and processing the same message. The default visibility timeout for a message is 30 seconds. The minimum is 0 seconds. The maximum is 12 hours.

Amazon SQS visibility timeout

PDF | Kindle | RSS

When a consumer receives and processes a message from a queue, the message remains in the queue. Amazon SQS doesn't automatically delete the message. Because Amazon SQS is a distributed system, there's no guarantee that the consumer actually receives the message (for example, due to a connectivity issue, or due to an issue in the consumer application). Thus, the consumer must delete the message from the queue after receiving and processing it.



Immediately after a message is received, it remains in the queue. To prevent other consumers from processing the message again, Amazon SQS sets a *visibility timeout*, a period of time during which Amazon SQS prevents other consumers from receiving and processing the message. The default visibility timeout for a message is 30 seconds. The minimum is 0 seconds. The maximum is 12 hours. For information about configuring visibility timeout for a queue using the console, see Configuring queue parameters (console).

via - https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-visibility-timeout.html

For example, you have a message with a visibility timeout of 5 minutes. After 3 minutes, you call ChangeMessageVisibility with a timeout of 10 minutes. You can continue to call ChangeMessageVisibility to extend the visibility timeout to the maximum allowed time. If you try to extend the visibility timeout beyond the maximum, your request is rejected. So, for the given use-case, the application can set the initial visibility timeout to 1 minute and then continue to update the ChangeMessageVisibility value if required.

ChangeMessageVisibility

PDF

Changes the visibility timeout of a specified message in a queue to a new value. The default visibility timeout for a message is 30 seconds. The minimum is 0 seconds. The maximum is 12 hours. For more information, see Visibility Timeout in the *Amazon Simple Queue Service Developer Guide*.

For example, you have a message with a visibility timeout of 5 minutes. After 3 minutes, you call `ChangeMessageVisibility` with a timeout of 10 minutes. You can continue to call `ChangeMessageVisibility` to extend the visibility timeout to the maximum allowed time. If you try to extend the visibility timeout beyond the maximum, your request is rejected.

An Amazon SQS message has three basic states:

1. Sent to a queue by a producer.
2. Received from the queue by a consumer.
3. Deleted from the queue.

A message is considered to be *stored* after it is sent to a queue by a producer, but not yet received from the queue by a consumer (that is, between states 1 and 2). There is no limit to the number of stored messages. A message is considered to be *in flight* after it is received from a queue by a consumer, but not yet deleted from the queue (that is, between states 2 and 3). There is a limit to the number of inflight messages.

Limits that apply to inflight messages are unrelated to the *unlimited* number of stored messages.

via - https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-visibility-timeout.html

Incorrect options:

Use DelaySeconds action to delay a message's visibility timeout - Delay queues let you postpone the delivery of new messages to a queue for a number of seconds. 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. You cannot use DelaySeconds to alter the visibility of a message which has been picked for processing.

Use WaitTimeSeconds action to short poll and extend a message's visibility timeout

Use WaitTimeSeconds action to long poll and extend a message's visibility timeout

Amazon SQS provides short polling and long polling to receive messages from a queue. Both these options have been added as distractors as WaitTimeSeconds (via short polling or long polling) cannot be used to influence the message's visibility.

References:

https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-visibility-timeout.html

https://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_ChangeMessageVisibility.html

Question 40:
Your team-mate has configured an Amazon S3 event notification for an S3 bucket that holds sensitive audit data of a firm. As the Team Lead, you are receiving the SNS notifications for every event in this bucket. After validating the event data, you realized that few events are missing.
What could be the reason for this behavior and how to avoid this in the future?

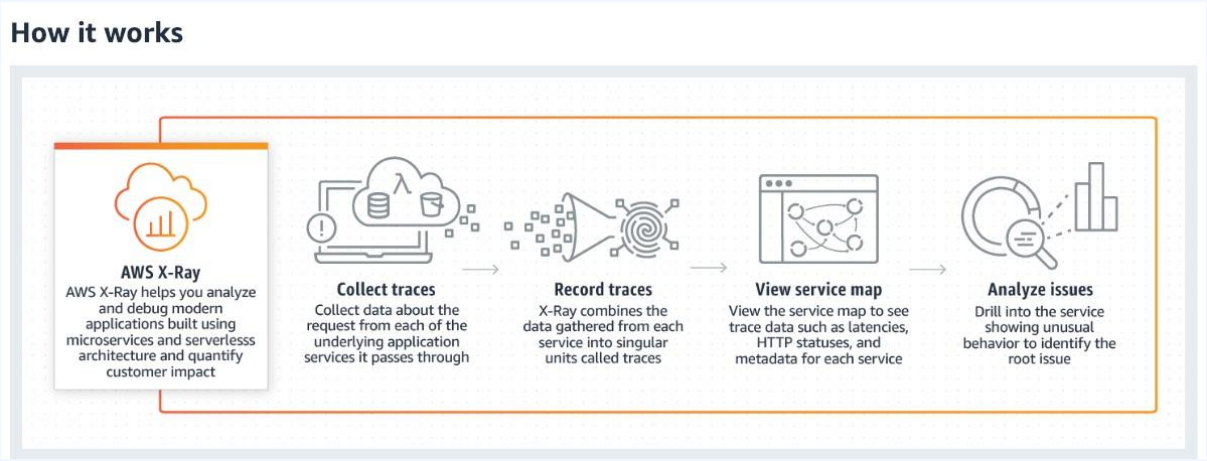
- If two writes are made to a single non-versioned object at the same time, it is possible that only a single event notification will be sent(Correct)
- Versioning is enabled on the S3 bucket and event notifications are getting fired for only one version
- Someone could have created a new notification configuration and that has overridden your existing configuration
- Your notification action is writing to the same bucket that triggers the notification

Explanation
Correct option:
If two writes are made to a single non-versioned object at the same time, it is possible that only a single event notification will be sent - Amazon S3 event notifications are designed to be delivered at least once. Typically, event notifications are delivered in seconds but can sometimes take a minute or longer.
If two writes are made to a single non-versioned object at the same time, it is possible that only a single event notification will be sent. If you want to ensure that an event notification is sent for every successful write, you can enable versioning on your bucket. With versioning, every successful write will create a new version of your object and will also send event notification.
Incorrect options:
Someone could have created a new notification configuration and that has overridden your existing configuration - It is possible that the configuration can be overridden. But, in the current scenario, the team lead is receiving notifications for most of the events, which nullifies the claim that the configuration is overridden.
Versioning is enabled on the S3 bucket and event notifications are getting fired for only one version - This is an incorrect statement. If you want to ensure that an event notification is sent for every successful write, you should enable versioning on your bucket. With versioning, every successful write will create a new version of your object and will also send event notification.
Your notification action is writing to the same bucket that triggers the notification - If your notification ends up writing to the bucket that triggers the notification, this could cause an execution loop. But it will not result in missing events.
Reference:
https://docs.aws.amazon.com/AmazonS3/latest/dev/NotificationHowTo.html

Question 41:
A company uses microservices-based infrastructure to process the API calls from clients, perform request filtering and cache requests using the AWS API Gateway. Users report receiving 501 error code and you have been contacted to find out what is failing.
Which service will you choose to help you troubleshoot?

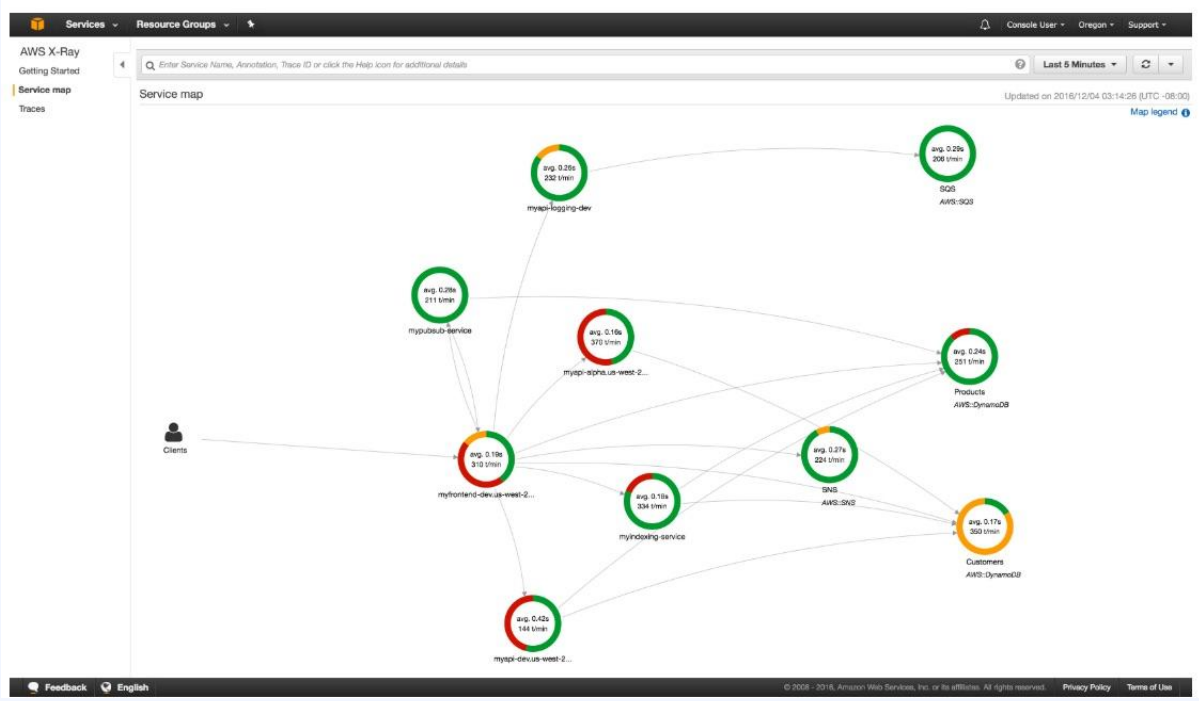
- Use X-Ray service(Correct)
- Use API Gateway service
- Use CloudWatch service
- Use CloudTrail service

Explanation
Correct option:
Use X-Ray service - 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. You can use X-Ray to analyze both applications in development and in production, from simple three-tier applications to complex microservices applications consisting of thousands of services.
X-Ray Overview:



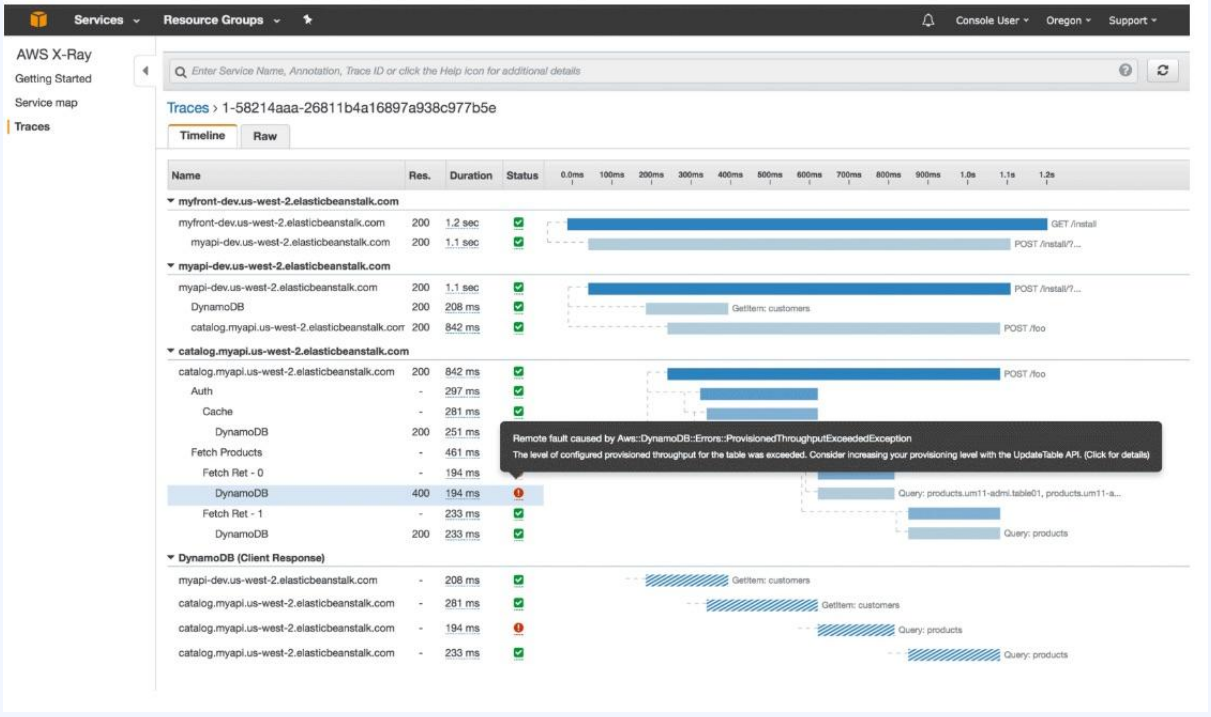
via - https://aws.amazon.com/xray/
AWS X-Ray creates a map of services used by your application with trace data that you can use to drill into specific services or issues. This provides a view of connections between services in your application and aggregated data for each service, including average latency and failure rates. You can create dependency trees, perform cross-availability zone or region call detections, and more.

X-Ray Service maps:



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

X-Ray Traces:



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

Incorrect options:

Use CloudTrail service - With CloudTrail, you can get a history of AWS API calls for your account - including API calls made via the AWS Management Console, AWS SDKs, command-line tools, and higher-level AWS services (such as AWS CloudFormation). This is a very useful service for general

monitoring and tracking. But, it will not give a detailed analysis of the outcome of microservices or drill into specific issues. For the current use case, X-Ray offers a better solution.
Use API Gateway service - Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale. API Gateway will not be able to drill into the flow between different microservices or their issues.
Use CloudWatch service - Amazon CloudWatch is a monitoring and management service that provides data and actionable insights for AWS, hybrid, and on-premises applications and infrastructure resources. CloudWatch can collect numbers and respond to AWS service-related events, but it can't help you debug microservices specific issues on AWS.
References:
https://aws.amazon.com/cloudtrail/
https://aws.amazon.com/api-gateway/
https://aws.amazon.com/cloudwatch/features/

Question 42:
A developer has just completed configuring the Application Load Balancer for the EC2 instances. Just as he started testing his configuration, he realized that he has missed assigning target groups to his ALB.
Which error code should he expect in his debug logs?

- HTTP 504
- HTTP 503(Correct)
- HTTP 500
- HTTP 403

Explanation
Correct option:
HTTP 503 - HTTP 503 indicates 'Service unavailable' error. This error in ALB is an indicator of the target groups for the load balancer having no registered targets.
Incorrect options:
HTTP 500 - HTTP 500 indicates 'Internal server' error. There are several reasons for their error: A client submitted a request without an HTTP protocol, and the load balancer was unable to generate a redirect URL, there was an error executing the web ACL rules.
HTTP 504 - HTTP 504 is 'Gateway timeout' error. Several reasons for this error, to quote a few: The load balancer failed to establish a connection to the target before the connection timeout expired, The load balancer established a connection to the target but the target did not respond before the idle timeout period elapsed.
HTTP 403 - HTTP 403 is 'Forbidden' error. You configured an AWS WAF web access control list (web ACL) to monitor requests to your Application Load Balancer and it blocked a request.
Reference:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html

Question 43:
Your application sends messages to an Amazon Simple Queue Service (SQS) queue frequently, which are then polled by another application that specifies which message to retrieve.
Which of the following options describe the maximum number of messages that can be retrieved at one time?

- 5
- 10(Correct)
- 100
- 20

Explanation
Correct option:
10
After you send messages to a queue, you can receive and delete them. When you request messages from a queue, you can't specify which messages to retrieve. Instead, you specify the maximum number of messages (up to 10) that you want to retrieve.
Incorrect options:
5
20
100
Reference:
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-using-receive-delete-message.html

Question 44:
As a Developer Associate, you are responsible for the data management of the AWS Kinesis streams at your company. The security team has mandated stricter security requirements by leveraging mechanisms available with the Kinesis Data Streams service that won't require code changes on your end.
Which of the following features meet the given requirements? (Select two)

- KMS encryption for data at rest(Correct)
- Encryption in flight with HTTPS endpoint(Correct)
- SSE-C encryption
- Envelope Encryption
- Client-Side Encryption

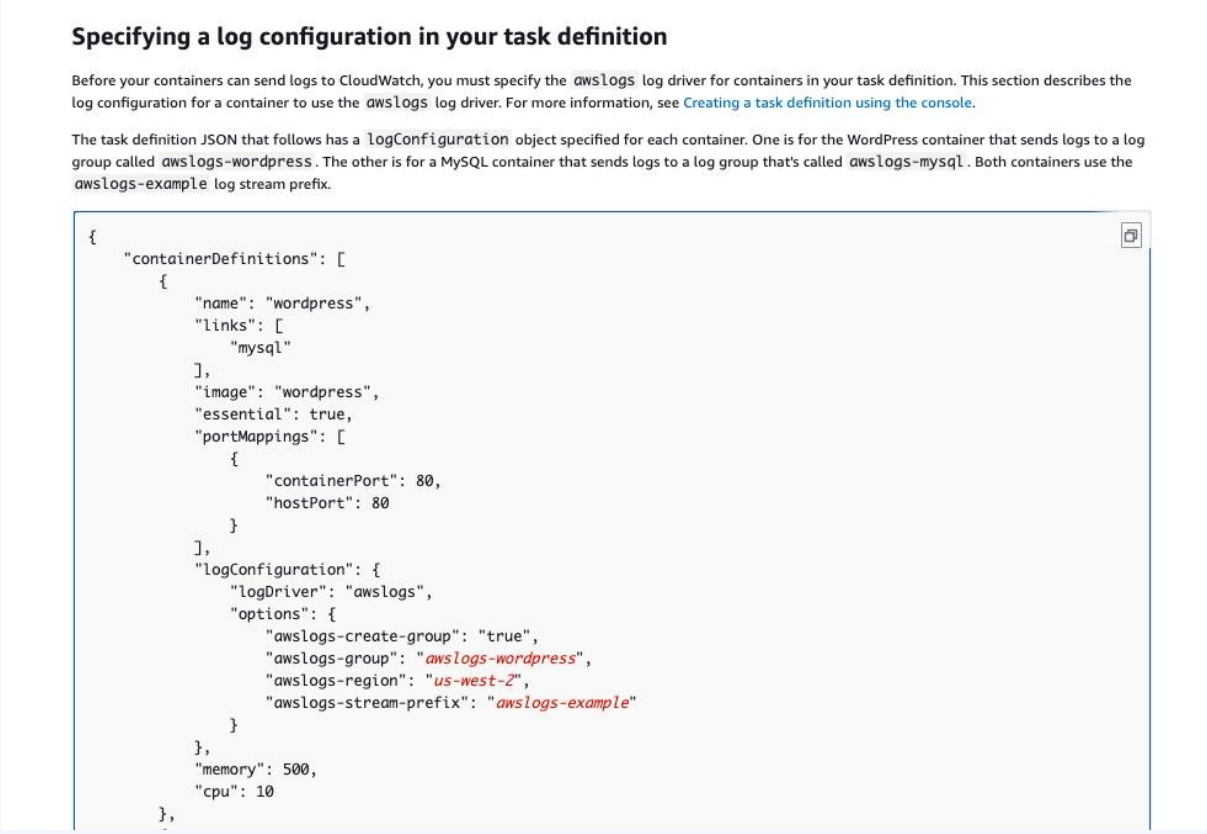
Explanation
Correct options:
KMS encryption for data at rest
Encryption in flight with HTTPS endpoint
Server-side encryption is a feature in Amazon Kinesis Data Streams that automatically encrypts data before it's at rest by using an AWS KMS customer master key (CMK) you specify. Data is encrypted before it's written to the Kinesis stream storage layer and decrypted after it's retrieved from storage. As a result, your data is encrypted at rest within the Kinesis Data Streams service. Also, the HTTPS protocol ensures that data inflight is encrypted as well.
Incorrect options:
SSE-C encryption - SSE-C is functionality in Amazon S3 where S3 encrypts your data, on your behalf, using keys that you provide. This does not apply for the given use-case.
Client-Side Encryption - This involves code changes, so the option is incorrect.
Envelope Encryption - This involves code changes, so the option is incorrect.
Reference:
https://docs.aws.amazon.com/streams/latest/dev/what-is-sse.html

Question 45:
An application runs on Amazon Elastic Container Service (Amazon ECS) on AWS Fargate. The company's audit requirements mandate that logging and storing of application log data must be done centrally on AWS.
How will you configure this requirement?

- Use the awslogs log driver to configure the containers in your tasks to send log information to CloudWatch Logs. Add the required logConfiguration parameters to your task definition(Correct)
- Use the awslogs log driver to send log information to CloudWatch Logs. To turn on the awslogs log driver, your Amazon ECS container instances require at least version 1.9.0 of the container agent
- Download and install the unified CloudWatch agent on the ECS instances to collect internal system-level metrics and application logs from the instances. The logs collected by the unified CloudWatch agent are processed and stored in Amazon CloudWatch logs and can be queried for report generation
- Amazon ECS metric data is automatically sent to CloudWatch in 1-minute periods. Amazon ECS service using the Fargate launch type has CloudWatch CPU and memory utilization metrics that can be enabled from the ECS console

Explanation
Correct option:
Use the awslogs log driver to configure the containers in your tasks to send log information to CloudWatch Logs. Add the required logConfiguration parameters to your task definition
Using the awslogs log driver you can configure the containers in your tasks to send log information to CloudWatch Logs. If you're using the Fargate launch type for your tasks, you need to add the required logConfiguration parameters to your task definition to turn on the awslogs log driver.
Before your containers can send logs to CloudWatch, you must specify the awslogs log driver for containers in your task definition. The example task definition JSON that follows has a logConfiguration object specified for each container. One is for the WordPress container that sends logs to a log group called awslogs-wordpress. The other is for a MySQL container that sends logs to a log group that's called awslogs-mysql. Both containers use the awslogs-example log stream. prefix.

Specifying a log configuration in your task definition:



Specifying a log configuration in your task definition

Before your containers can send logs to CloudWatch, you must specify the `awslogs` log driver for containers in your task definition. This section describes the log configuration for a container to use the `awslogs` log driver. For more information, see Creating a task definition using the console.

The task definition JSON that follows has a `logConfiguration` object specified for each container. One is for the WordPress container that sends logs to a log group called `awslogs-wordpress`. The other is for a MySQL container that sends logs to a log group that's called `awslogs-mysql`. Both containers use the `awslogs-example` log stream prefix.

```
{
    "containerDefinitions": [
        {
            "name": "wordpress",
            "links": [
                "mysql"
            ],
            "image": "wordpress",
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-create-group": "true",
                    "awslogs-group": "awslogs-wordpress",
                    "awslogs-region": "us-west-2",
                    "awslogs-stream-prefix": "awslogs-example"
                }
            },
            "memory": 500,
            "cpu": 10
        },
```

via - https://docs.aws.amazon.com/AmazonECS/latest/developerguide/using_awslogs.html#specify-log-config

Incorrect options:

Use the awslogs log driver to send log information to CloudWatch Logs. To turn on the awslogs log driver, your Amazon ECS container instances require at least version 1.9.0 of the container agent - This statement is incorrect. If you're using the EC2 launch type (and not Fargate) for your tasks and want to turn on the awslogs log driver, your Amazon ECS container instances require at least version 1.9.0 of the container agent.

Amazon ECS metric data is automatically sent to CloudWatch in 1-minute periods. Amazon ECS service using the Fargate launch type has CloudWatch CPU and memory utilization metrics that can be enabled from the ECS console - Indeed, Amazon ECS metric data is automatically sent to CloudWatch in 1-minute periods. Also, any Amazon ECS service using the Fargate launch type has CloudWatch CPU and memory utilization metrics automatically, so you don't need to take any manual steps. But, these are system logs and not the application logs as is needed in the current use case.

Download and install the unified CloudWatch agent on the ECS instances to collect internal system-level metrics and application logs from the instances. The logs collected by the unified CloudWatch agent are processed and stored in Amazon CloudWatch logs and can be queried for report generation - This statement is incorrect and given only as a distractor. ECS Fargate is serverless and hence the scope of downloading and installing software on the instance does not arise.

References:

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/using_awslogs.html
https://docs.aws.amazon.com/AmazonECS/latest/userguide/cloudwatch-metrics.html#available_cloudwatch_metrics

Question 46:

The development team at an IT company uses CloudFormation to manage its AWS infrastructure. The team has created a network stack containing a VPC with subnets and a web application stack with EC2 instances and an RDS instance. The team wants to reference the VPC created in the network stack into its web application stack.

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

- Create a cross-stack reference and use the Outputs output field to flag the value of VPC from the network stack. Then use Ref intrinsic function to reference the value of VPC into the web application stack
- Create a cross-stack reference and use the Export output field to flag the value of VPC from the network stack. Then use Fn::ImportValue intrinsic function to import the value of VPC into the web application stack(Correct)
- Create a cross-stack reference and use the Export output field to flag the value of VPC from the network stack. Then use Ref intrinsic function to reference the value of VPC into the web application stack
- Create a cross-stack reference and use the Outputs output field to flag the value of VPC from the network stack. Then use Fn::ImportValue intrinsic function to import the value of VPC into the web application stack

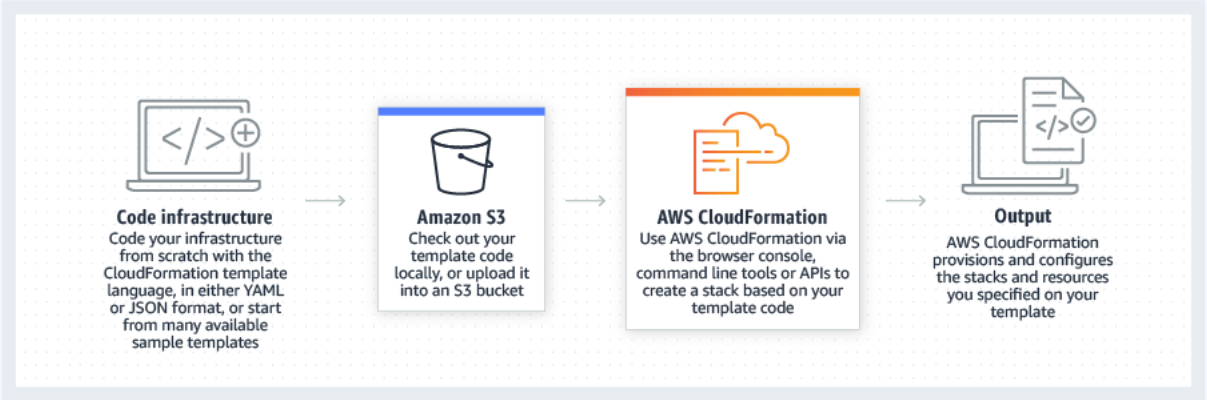
Explanation

Correct option:

Create a cross-stack reference and use the Export output field to flag the value of VPC from the network stack. Then use Fn::ImportValue intrinsic function to import the value of VPC into the web application stack

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.

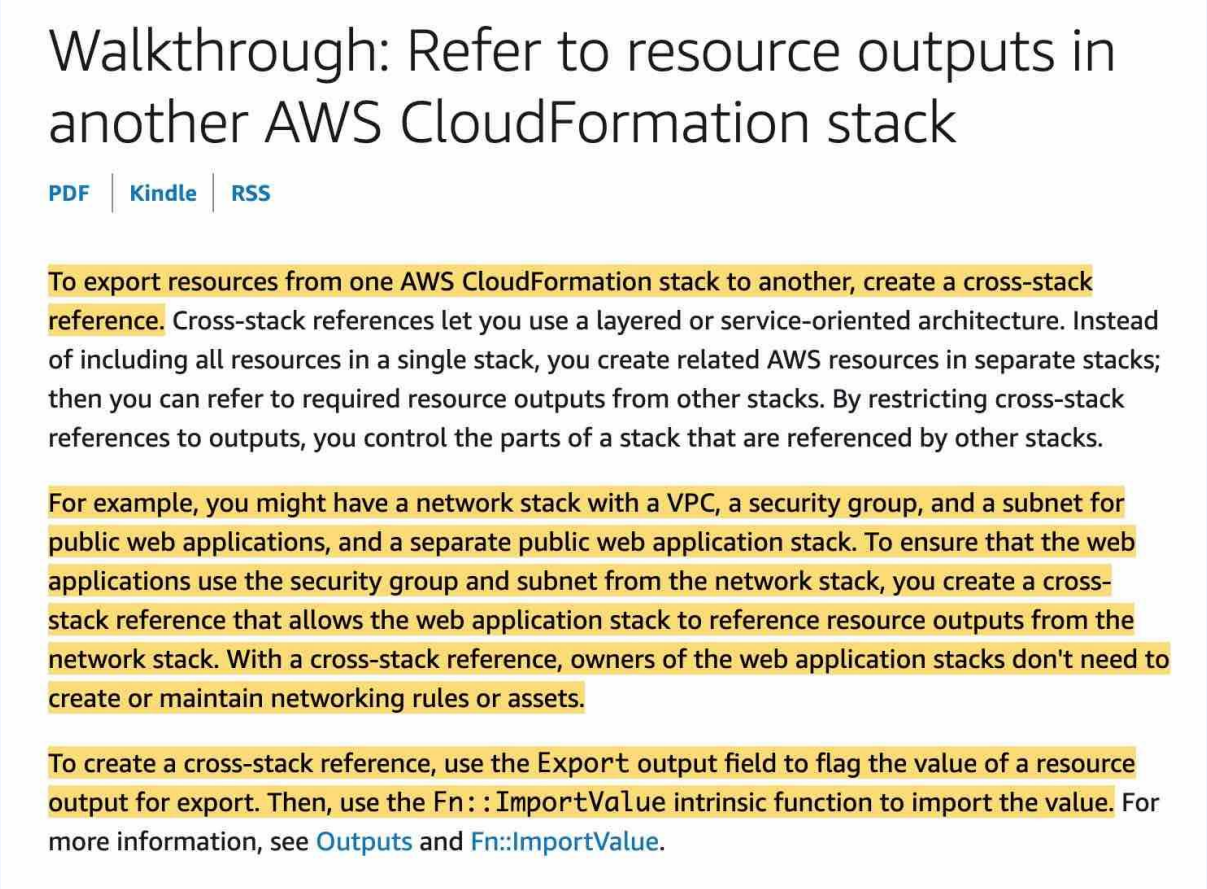
How CloudFormation Works:



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

You can create a cross-stack reference to export resources from one AWS CloudFormation stack to another. For example, you might have a network stack with a VPC and subnets and a separate public web application stack. To use the security group and subnet from the network stack, you can create a cross-stack reference that allows the web application stack to reference resource outputs from the

network stack. With a cross-stack reference, owners of the web application stacks don't need to create or maintain networking rules or assets.
To create a cross-stack reference, use the Export output field to flag the value of a resource output for export. Then, use the Fn::ImportValue intrinsic function to import the value.
You cannot use the Ref intrinsic function to import the value.



Walkthrough: Refer to resource outputs in another AWS CloudFormation stack

PDF | Kindle | RSS

To export resources from one AWS CloudFormation stack to another, create a cross-stack reference. Cross-stack references let you use a layered or service-oriented architecture. Instead of including all resources in a single stack, you create related AWS resources in separate stacks; then you can refer to required resource outputs from other stacks. By restricting cross-stack references to outputs, you control the parts of a stack that are referenced by other stacks.

For example, you might have a network stack with a VPC, a security group, and a subnet for public web applications, and a separate public web application stack. To ensure that the web applications use the security group and subnet from the network stack, you create a cross-stack reference that allows the web application stack to reference resource outputs from the network stack. With a cross-stack reference, owners of the web application stacks don't need to create or maintain networking rules or assets.

To create a cross-stack reference, use the `Export` output field to flag the value of a resource output for export. Then, use the `Fn::ImportValue` intrinsic function to import the value. For more information, see Outputs and Fn::ImportValue.

via - https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/walkthrough-crossstackref.html

Incorrect options:
Create a cross-stack reference and use the Outputs output field to flag the value of VPC from the network stack. Then use Fn::ImportValue intrinsic function to import the value of VPC into the web application stack
Create a cross-stack reference and use the Outputs output field to flag the value of VPC from the network stack. Then use Ref intrinsic function to reference the value of VPC into the web application stack
Create a cross-stack reference and use the Export output field to flag the value of VPC from the network stack. Then use Ref intrinsic function to reference the value of VPC into the web application stack
These three options contradict the details provided in the explanation above, so these options are not correct.
References:
https://aws.amazon.com/cloudformation/

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/walkthrough-crossstackref.html

Question 47:
A developer in your company has configured a build using AWS CodeBuild. The build fails and the developer needs to quickly troubleshoot the issue to see which commands or settings located in the BuildSpec file are causing an issue.
Which approach will help them accomplish this?

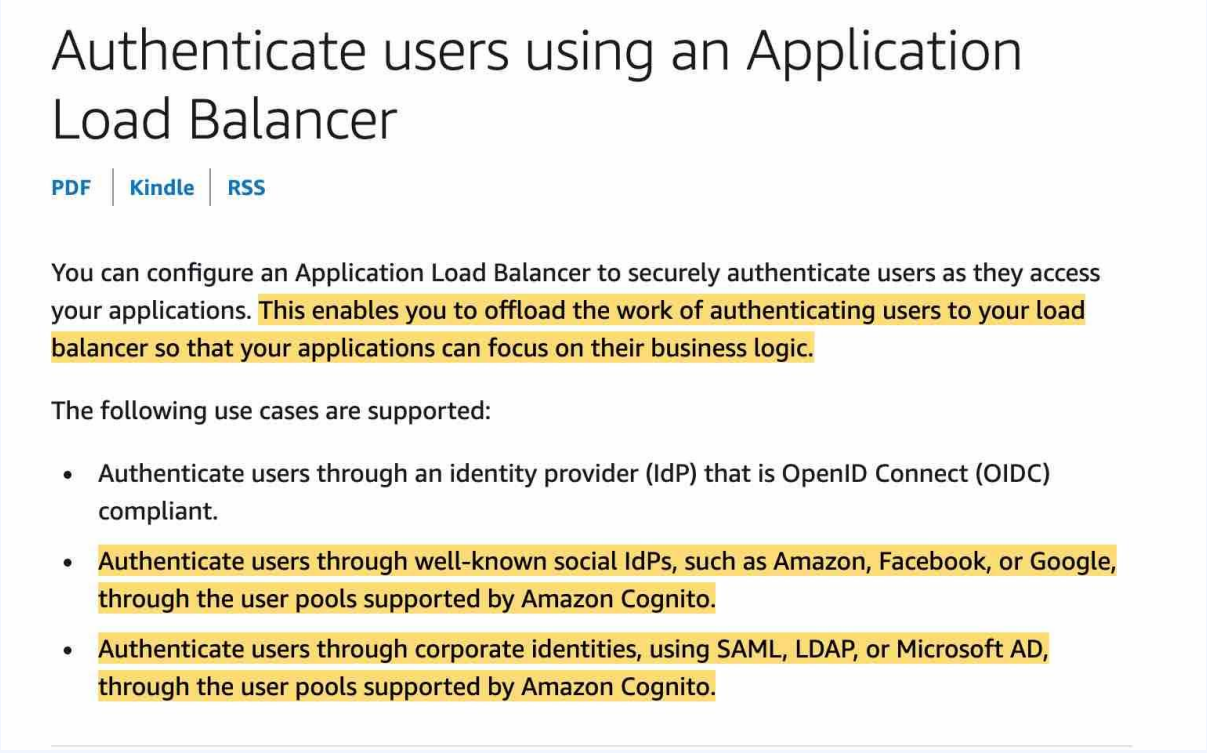
- Freeze the CodeBuild during its next execution
- Enable detailed monitoring
- Run AWS CodeBuild locally using CodeBuild Agent(Correct)
- SSH into the CodeBuild Docker container

Explanation
Correct option:
Run AWS CodeBuild locally using CodeBuild Agent
AWS CodeBuild is a fully managed build service. There are no servers to provision and scale, or software to install, configure, and operate.
With the Local Build support for AWS CodeBuild, you just specify the location of your source code, choose your build settings, and CodeBuild runs build scripts for compiling, testing, and packaging your code. You can use the AWS CodeBuild agent to test and debug builds on a local machine.
By building an application on a local machine you can:
Test the integrity and contents of a buildspec file locally.
Test and build an application locally before committing.
Identify and fix errors quickly from your local development environment.
Incorrect options:
SSH into the CodeBuild Docker container - It is not possible to SSH into the CodeBuild Docker container, that's why you should test and fix errors locally.
Freeze the CodeBuild during its next execution - You cannot freeze the CodeBuild process but you can stop it. Please see more details on -
https://docs.aws.amazon.com/codebuild/latest/userguide/stop-build.html
Enable detailed monitoring - Detailed monitoring is available for EC2 instances. You do not enable detailed monitoring but you can specify output logs to be captured via CloudTrail.
AWS CodeBuild is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in CodeBuild. CloudTrail captures all API calls for CodeBuild as events, including calls from the CodeBuild console and from code calls to the CodeBuild APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an S3 bucket, including events for CodeBuild.
References:
https://docs.aws.amazon.com/codebuild/latest/userguide/troubleshooting.html
https://aws.amazon.com/blogs/devops/announcing-local-build-support-for-aws-codebuild/
https://docs.aws.amazon.com/codebuild/latest/userguide/use-codebuild-agent.html

Question 48:
A photo-sharing application manages its EC2 server fleet running behind an Application Load Balancer and the traffic is fronted by a CloudFront distribution. The development team wants to decouple the user authentication process for the application so that the application servers can just focus on the business logic.
As a Developer Associate, which of the following solutions would you recommend to address this use-case with minimal development effort?

- Use Cognito Authentication via Cognito User Pools for your Application Load Balancer(Correct)
- Use Cognito Authentication via Cognito Identity Pools for your Application Load Balancer
- Use Cognito Authentication via Cognito Identity Pools for your CloudFront distribution
- Use Cognito Authentication via Cognito User Pools for your CloudFront distribution

Explanation
Correct option:
Use Cognito Authentication via Cognito User Pools for your Application Load Balancer
Application Load Balancer can be used to securely authenticate users for accessing your applications. This enables you to offload the work of authenticating users to your load balancer so that your applications can focus on their business logic. You can use Cognito User Pools to authenticate users through well-known social IdPs, such as Amazon, Facebook, or Google, through the user pools supported by Amazon Cognito or through corporate identities, using SAML, LDAP, or Microsoft AD, through the user pools supported by Amazon Cognito. You configure user authentication by creating an authenticate action for one or more listener rules. The authenticate-cognito and authenticate-oidc action types are supported only with HTTPS listeners.



Authenticate users using an Application Load Balancer

PDF | Kindle | RSS

You can configure an Application Load Balancer to securely authenticate users as they access your applications. This enables you to offload the work of authenticating users to your load balancer so that your applications can focus on their business logic.

The following use cases are supported:

- Authenticate users through an identity provider (IdP) that is OpenID Connect (OIDC) compliant.
- Authenticate users through well-known social IdPs, such as Amazon, Facebook, or Google, through the user pools supported by Amazon Cognito.
- Authenticate users through corporate identities, using SAML, LDAP, or Microsoft AD, through the user pools supported by Amazon Cognito.

via -

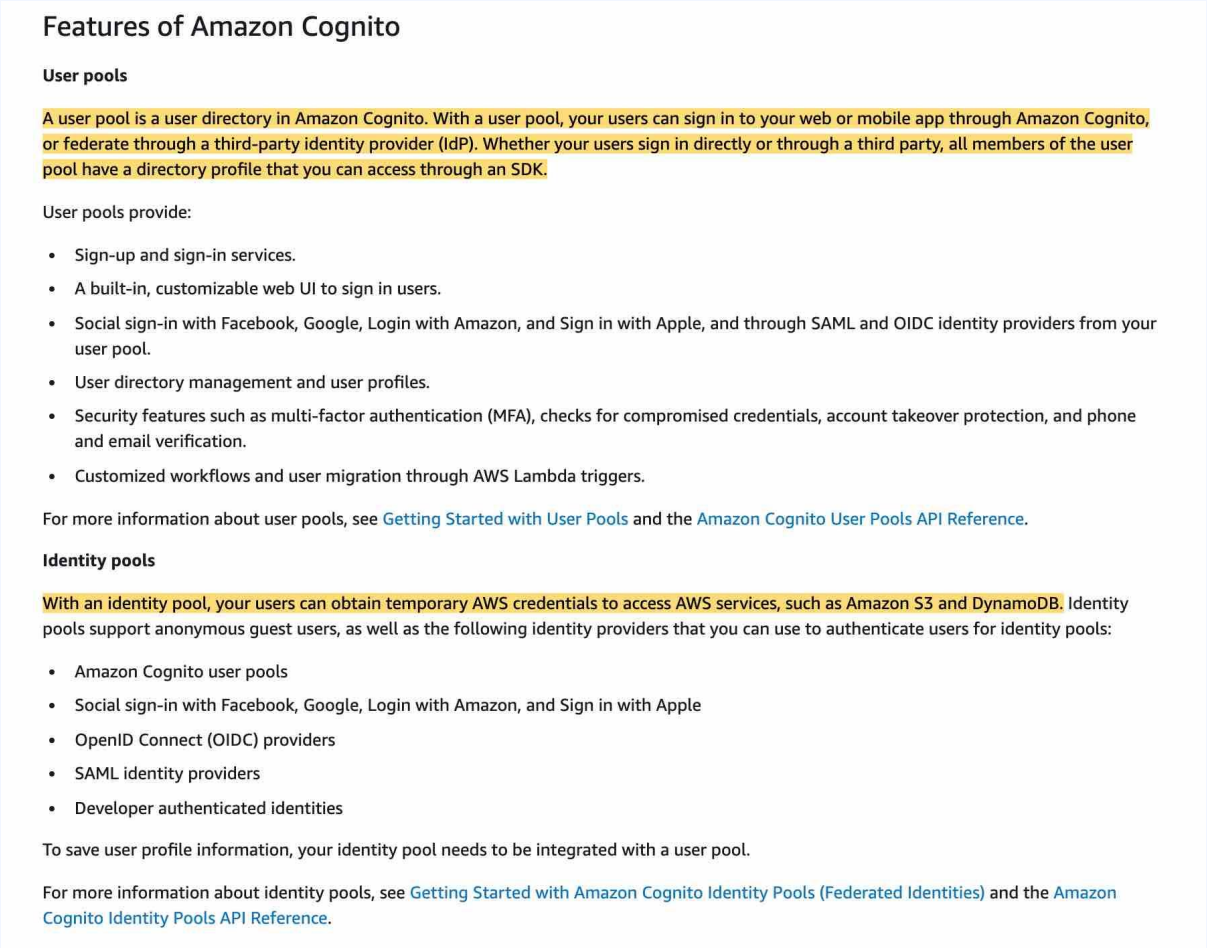
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/listener-authenticate-users.html
Please make sure that you adhere to the following configurations while using CloudFront distribution in front of your Application Load Balancer:

Prepare to use Amazon CloudFront

Enable the following settings if you are using a CloudFront distribution in front of your Application Load Balancer:

- Forward request headers (all) — Ensures that CloudFront does not cache responses for authenticated requests. This prevents them from being served from the cache after the authentication session expires. Alternatively, to reduce this risk while caching is enabled, owners of a CloudFront distribution can set the time-to-live (TTL) value to expire before the authentication cookie expires.
- Query string forwarding and caching (all) — Ensures that the load balancer has access to the query string parameters required to authenticate the user with the IdP.
- Cookie forwarding (all) — Ensures that CloudFront forwards all authentication cookies to the load balancer.

via - https://docs.aws.amazon.com/elasticloadbalancing/latest/application/listener-authenticate-users.html
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

Incorrect options:

Use Cognito Authentication via Cognito Identity Pools for your Application Load Balancer - There is no such thing as using Cognito Authentication via Cognito Identity Pools for managing user authentication for the application. Application-specific user authentication can be provided via Cognito User 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.

Use Cognito Authentication via Cognito User Pools for your CloudFront distribution - You cannot directly integrate Cognito User Pools with CloudFront distribution as you have to create a separate Lambda@Edge function to accomplish the authentication via Cognito User Pools. This involves additional development effort, so this option is not the best fit for the given use-case.

Use Cognito Authentication via Cognito Identity Pools for your CloudFront distribution - You cannot use Cognito Identity Pools for managing user authentication, so this option is not correct.

References:

https://docs.aws.amazon.com/elasticloadbalancing/latest/application/listener-authenticate-users.html

https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-identity-pools.html

https://aws.amazon.com/blogs/networking-and-content-delivery/authorizationedge-using-cookies-protect-your-amazon-cloudfront-content-from-being-downloaded-by-unauthenticated-users/

Question 49:
A company wants to automate the creation of ECS clusters using CloudFormation. The process has worked for a while, but after creating task definitions and assigning roles, the development team discovers that the tasks for containers are not using the permissions assigned to them.
Which ECS config must be set in /etc/ecs/ecs.config to allow ECS tasks to use IAM roles?

- ECS_ENABLE_TASK_IAM_ROLE(Correct)
- ECS_ENGINE_AUTH_DATA
- ECS_AVAILABLE_LOGGING_DRIVERS
- ECS_CLUSTER

Explanation
Correct option:
ECS_ENABLE_TASK_IAM_ROLE
This configuration item is used to enable IAM roles for tasks for containers with the bridge and default network modes.
Incorrect options:
ECS_ENGINE_AUTH_DATA - This refers to the authentication data within a Docker configuration file, so this is not the correct option.
ECS_AVAILABLE_LOGGING_DRIVERS - The Amazon ECS container agent running on a container instance must register the logging drivers available on that instance with this variable. This configuration item refers to the logging driver.
ECS_CLUSTER - This refers to the ECS cluster that the ECS agent should check into. This is passed to the container instance at launch through Amazon EC2 user data.
Reference:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-agent-config.html

Question 50:
A developer while working on Amazon EC2 instances, realized that an instance was not needed and had shut it down. But another instance of the same type automatically got launched in the account. Which of the following options can attribute the given sequence of actions?

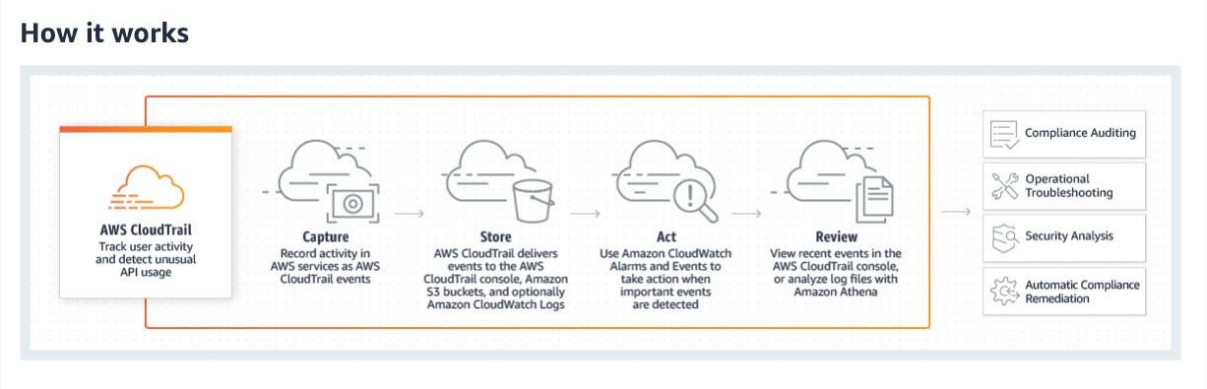
- Instance might be part of Auto Scaling Group and hence re-launched similar instance(Correct)
- The instance could have been a part of Application Load Balancer and hence was automatically started
- The instance could have been a part of Network Load Balancer and hence was automatically started
- The user did not have the right permissions to shutdown the instance. User needs root permissions to terminate an instance

Explanation
Correct option:
Instance might be part of Auto Scaling Group and hence re-launched similar instance - Auto Scaling groups can be configured to launch an instance to replace an instance that is undergoing maintenance. This could have been the reason why an instance of the same type got launched automatically. The size of an Auto Scaling group depends on the number of instances that you set as the desired capacity. If you wish to terminate an instance that is part of Auto Scaling Group, the configuration of the group should be changed to a reduced number of instances, so the automatic launch of instances does not happen when an unwanted instance is terminated.
Incorrect options:
The user did not have the right permissions to shutdown the instance. User needs root permissions to terminate an instance - This is an incorrect statement. If the user does not have enough permissions, then the action itself is unavailable for him. A user does not need root permissions to terminate an EC2 instance.
The instance could have been a part of the Application Load Balancer and hence was automatically started - Application Load Balancer is used to balance the incoming traffic requests equally among the available EC2 instances so keep the performance and availability at its best. ALBs are configured with Auto Scaling Groups, but this is not specified in the use-case. In the absence of Auto Scaling Group, ALB cannot launch instances by itself.
The instance could have been a part of Network Load Balancer and hence was automatically started - As explained above for ALB, a Network Load Balancer is not capable of launching instances by itself if it's not configured with an Auto Scaling Group.
References:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/AutoScalingGroup.html
https://docs.aws.amazon.com/autoscaling/ec2/userguide/detach-instance-asg.html

Question 51:
An AWS CodePipeline was configured to be triggered by Amazon CloudWatch Events. Recently the pipeline failed and upon investigation, the Team Lead noticed that the source was changed from AWS CodeCommit to Amazon Simple Storage Service (S3). The Team Lead has requested you to find the user who had made the changes.
Which service will help you solve this?

- Amazon CloudWatch
- AWS CloudTrail(Correct)
- AWS X-Ray
- Amazon Inspector

Explanation
Correct option:
AWS 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.
AWS CloudTrail increases visibility into your user and resource activity by recording AWS Management Console actions and API calls. You can identify which users and accounts called AWS, the source IP address from which the calls were made, and when the calls occurred.
How CloudTrail works:



via - https://aws.amazon.com/cloudtrail/
Incorrect options:
Amazon CloudWatch - Amazon CloudWatch is a monitoring and management service that provides data and actionable insights for AWS, hybrid, and on-premises applications and infrastructure resources. CloudWatch can collect numbers and respond to AWS service-related events, but it does not help in user activity logging.
AWS 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 is a very important tool in troubleshooting but is not useful in logging user activity.
Amazon Inspector - Amazon Inspector is an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. Amazon Inspector security assessments help you check for unintended network accessibility of your Amazon EC2 instances and for vulnerabilities on those EC2 instances. This does not log User activity at the account level.
References:
https://aws.amazon.com/xray/
https://aws.amazon.com/inspector/
https://aws.amazon.com/cloudwatch/

Question 52:
A startup manages its Cloud resources with Elastic Beanstalk. The environment consists of few Amazon EC2 instances, an Auto Scaling Group (ASG), and an Elastic Load Balancer. Even after the Load Balancer marked an EC2 instance as unhealthy, the ASG has not replaced it with a healthy instance.
As a Developer, suggest the necessary configurations to automate the replacement of unhealthy instance.

- The health check type of your instance's Auto Scaling group, must be changed from EC2 to ELB by using a configuration file(Correct)
- Health check parameters were configured for checking the instance health alone. The instance failed because of application failure which was not configured as a parameter for health check status
- The ping path field of the Load Balancer is configured incorrectly
- Auto Scaling group doesn't automatically replace the unhealthy instances marked by the load balancer. They have to be manually replaced from AWS Console

Explanation
Correct option:
The health check type of your instance's Auto Scaling group, must be changed from EC2 to ELB by using a configuration file - By default, the health check configuration of your Auto Scaling group is set as an EC2 type that performs a status check of EC2 instances. To automate the replacement of unhealthy EC2 instances, you must change the health check type of your instance's Auto Scaling group from EC2 to ELB by using a configuration file.
Incorrect options:
Health check parameters were configured for checking the instance health alone. The instance failed because of application failure which was not configured as a parameter for health check status - This is an incorrect statement. Status checks, by definition, cover only an EC2 instance's health, and not the health of your application, server, or any Docker containers running on the instance.
Auto Scaling group doesn't automatically replace the unhealthy instances marked by the load balancer. They have to be manually replaced from AWS Console - Incorrect statement. As discussed above, if the health check type of ASG is changed from EC2 to ELB, Auto Scaling will be able to replace the unhealthy instance.
The ping path field of the Load Balancer is configured incorrectly - Ping path is a health check configuration field of Elastic Load Balancer. If the ping path is configured wrong, ELB will not be able to reach the instance and hence will consider the instance unhealthy. However, this would then apply to all instances, not just once instance. So it does not address the issue given in the use-case.
References:
https://aws.amazon.com/premiumsupport/knowledge-center/elastic-beanstalk-instance-automation/
https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-healthchecks.html

Question 53:
Your organization has developers that merge code changes regularly to an AWS CodeCommit repository. Your pipeline has AWS CodeCommit as the source and you would like to configure a rule that reacts to changes in CodeCommit.
Which of the following options do you choose for this type of integration?

- Use Lambda Event Rules
- Use CloudWatch Event Rules(Correct)
- Use CloudTrail Event rules with Amazon Simple Email Service (SES)
- Use Lambda function with Amazon Simple Notification Service (SNS)

Explanation
Correct option:
Use CloudWatch Event Rules
Amazon CloudWatch Events is a web service that monitors your AWS resources and the applications you run on AWS. You can use Amazon CloudWatch Events to detect and react to changes in the state of a pipeline, stage, or action. Then, based on rules you create, CloudWatch Events invokes one or more target actions when a pipeline, stage, or action enters the state you specify in a rule. Examples of Amazon CloudWatch Events rules and targets:
A rule that sends a notification when the instance state changes, where an EC2 instance is the event source, and Amazon SNS is the event target.
A rule that sends a notification when the build phase changes, where a CodeBuild configuration is the event source, and Amazon SNS is the event target.
A rule that detects pipeline changes and invokes an AWS Lambda function.
Incorrect options:
Use CloudTrail Event rules with Amazon Simple Email Service (SES) - This is an incorrect statement. There is no such thing as CloudTrail Event Rule.
Use Lambda function with Amazon Simple Notification Service (SNS) - Lambda functions can be triggered by the use of CloudWatch Event Rules as discussed above. AWS CodePipeline does not trigger Lambda functions directly.
Use Lambda Event Rules - This is an incorrect statement. There is no such thing as Lambda Event Rule.
Reference:
https://docs.aws.amazon.com/codepipeline/latest/userguide/detect-state-changes-cloudwatch-events.html

Question 54:
A media application uses Amazon CloudFront distribution to distribute static content configured on an Amazon S3 bucket. The application is used across different countries and various AWS Regions. Some regions have been experiencing latency when there is a cache miss on CloudFront.
Which of the following configuration changes will you suggest to decrease latency and improve user performance?

- Redirect requests on cache misses to the S3 bucket nearest to the user country. Create a Lambda@Edge function to redirect requests based on the value of the CloudFront-Viewer-Country header. Associate the Lambda@Edge function with the distribution's viewer request event
- Redirect requests on cache misses to the S3 bucket nearest to the user country. Create a Lambda@Edge function to redirect requests based on the value of the CloudFront-Viewer-Country header. Associate the Lambda@Edge function with the distribution's origin request event
- Redirect requests on cache misses to the S3 bucket nearest to the user country. Create a CloudFront function to redirect requests based on the value of the CloudFront-Viewer-Country header. Associate the CloudFront function with the distribution's viewer request event(Correct)
- Redirect requests on cache misses to the S3 bucket nearest to the user country. Create a CloudFront function to redirect requests based on the value of the CloudFront-Viewer-Country header. Associate the CloudFront function with the distribution's origin request event

Explanation
Correct option:
Redirect requests on cache misses to the Amazon S3 bucket nearest to the user country. Create a CloudFront function to redirect requests based on the value of the CloudFront-Viewer-Country header. Associate the CloudFront function with the distribution's viewer request event
With CloudFront Functions in Amazon CloudFront, you can write lightweight functions in JavaScript for high-scale, latency-sensitive CDN customizations. Your functions can manipulate the requests and responses that flow through CloudFront, perform basic authentication and authorization, generate HTTP responses at the edge, and more.
When you associate a CloudFront function with a CloudFront distribution, CloudFront intercepts requests and responses at CloudFront edge locations and passes them to your function. You can invoke CloudFront functions when the following events occur: 1. When CloudFront receives a request from a viewer (viewer request): The function executes when CloudFront receives a request from a viewer before it checks to see whether the requested object is in the CloudFront cache.
Before CloudFront returns the response to the viewer (viewer response): The function executes before returning the requested file to the viewer. Note that the function executes regardless of whether the file is already in the CloudFront cache.
We use the value of the CloudFront-Viewer-Country header to update the S3 bucket domain name to a bucket in a Region that is closer to the viewer. This can be useful in several ways: 1. It reduces latencies when the Region specified is nearer to the viewer's country. 2. It provides data sovereignty

by making sure that data is served from an origin that's in the same country that the request came from.
The example below shows how a Lambda handler is used to change response based on user country. A similar CloudFront function can be defined to direct user traffic to the nearest S3 bucket.
Example for redirecting viewer requests to a country-specific URL:

```python
# This is an origin request function

def lambda_handler(event, context):
    request = event['Records'][0]['cf']['request']
    headers = request['headers']

    '''
    Based on the value of the CloudFront-Viewer-Country header, generate an
    HTTP status code 302 (Redirect) response, and return a country-specific
    URL in the Location header.
    NOTE: 1. You must configure your distribution to cache based on the
            CloudFront-Viewer-Country header. For more information, see
            https://docs.aws.amazon.com/console/cloudfront/cache-on-selected-headers
          2. CloudFront adds the CloudFront-Viewer-Country header after the viewer
            request event. To use this example, you must create a trigger for the
            origin request event.
    '''

    url = 'https://example.com/'
    viewerCountry = headers.get('cloudfront-viewer-country')
    if viewerCountry:
        countryCode = viewerCountry[0]['value']
        if countryCode == 'TW':
            url = 'https://tw.example.com/'
        elif countryCode == 'US':
            url = 'https://us.example.com/'

    response = {
        'status': '302',
        'statusDescription': 'Found',
        'headers': {
            'location': [{
                'key': 'Location',
                'value': url
            }]
        }
    }

    return response
```

via - https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/lambda-examples.html#lambda-examples-content-based-S3-origin-request-trigger

Choosing between CloudFront Functions and Lambda@Edge:

CloudFront Functions is ideal for lightweight, short-running functions for use cases like the following:

- **Cache key normalization** – You can transform HTTP request attributes (headers, query strings, cookies, and even the URL path) to create an optimal cache key, which can improve your cache hit ratio.
- **Header manipulation** – You can insert, modify, or delete HTTP headers in the request or response. For example, you can add a `True-Client-IP` header to every request.
- **URL redirects or rewrites** – You can redirect viewers to other pages based on information in the request, or rewrite all requests from one path to another.
- **Request authorization** – You can validate hashed authorization tokens, such as JSON web tokens (JWT), by inspecting authorization headers or other request metadata.

To get started with CloudFront Functions, see Customizing at the edge with CloudFront Functions.

Lambda@Edge is a good fit for the following scenarios:

- Functions that take several milliseconds or more to complete.
- Functions that require adjustable CPU or memory.
- Functions that depend on third-party libraries (including the AWS SDK, for integration with other AWS services).
- Functions that require network access to use external services for processing.
- Functions that require file system access or access to the body of HTTP requests.

via - https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/edge-functions.html

Incorrect options:

Redirect requests on cache misses to the S3 bucket nearest to the user country. Create a Lambda@Edge function to redirect requests based on the value of the CloudFront-Viewer-Country header. Associate the Lambda@Edge function with the distribution's origin request event

Redirect requests on cache misses to the S3 bucket nearest to the user country. Create a Lambda@Edge function to redirect requests based on the value of the CloudFront-Viewer-Country header. Associate the Lambda@Edge function with the distribution's viewer request event - As discussed above, CloudFront Function is a better choice than Lambda@Edge function for this use case. Also, the Viewer request event is better suited for this requirement than the Origin request event to trigger the function.

Viewer request event: The function executes when CloudFront receives a request from a viewer before it checks to see whether the requested object is in the CloudFront cache.

Origin request event: The function executes only when CloudFront forwards a request to your origin. When the requested object is in the CloudFront cache, the function doesn't execute.

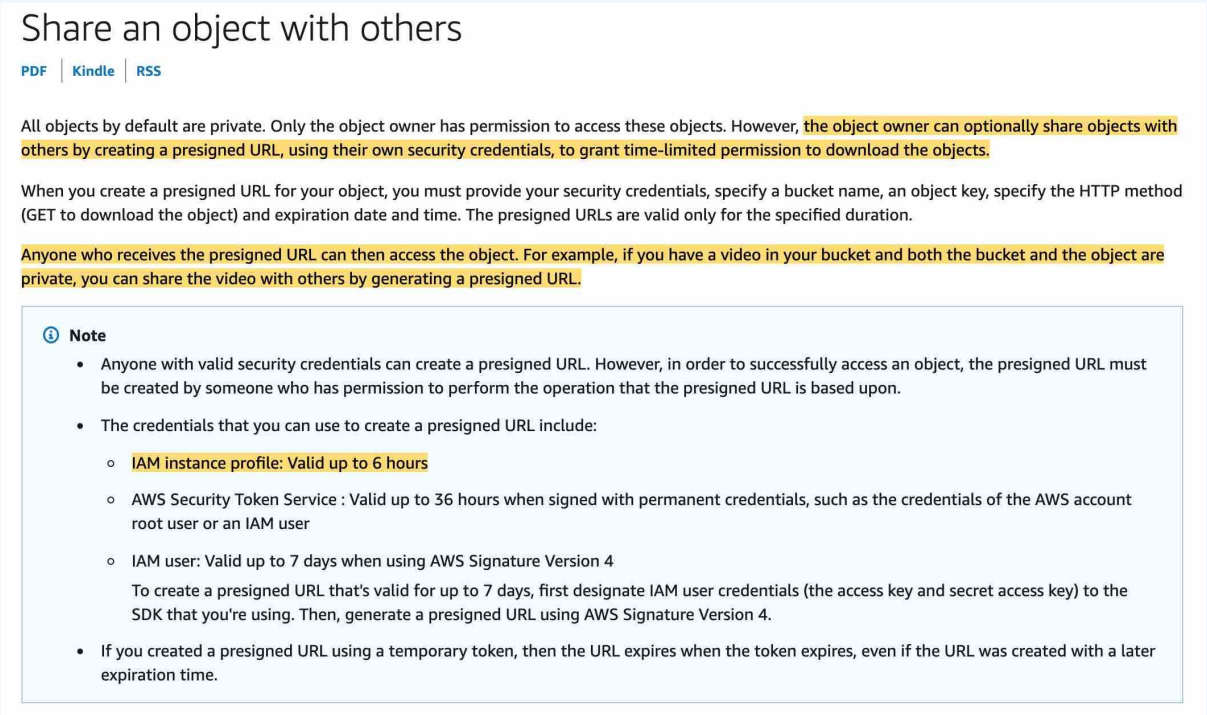
Redirect requests on cache misses to the S3 bucket nearest to the user country. Create a CloudFront function to redirect requests based on the value of the CloudFront-Viewer-Country header. Associate the CloudFront function with the distribution's origin request event - This option is incorrect. You can invoke CloudFront functions for only two events: When CloudFront receives a request from a viewer

(viewer request) and Before CloudFront returns the response to the viewer (viewer response). Origin request and Origin response events are not supported for the CloudFront function.
References:
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/lambda-examples.html#lambda-examples-redirecting-examples
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/edge-functions.html
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/lambda-cloudfront-trigger-events.html

Question 55:
You're a developer maintaining a web application written in .NET. The application makes references to public objects in a public S3 accessible bucket using a public URL. While doing a code review your colleague advises that the approach is not a best practice because some of the objects contain private data. After the administrator makes the S3 bucket private you can no longer access the S3 objects but you would like to create an application that will enable people to access some objects as needed with a time policy constraint.
Which of the following options will give access to the objects?

- Using pre-signed URL(Correct)
- Using IAM policy
- Using Routing Policy
- Using bucket policy

Explanation
Correct option:
"Using pre-signed URL"
All objects by default are private, with object owner having permission to access the objects. However, the object owner can optionally share objects with others by creating a pre-signed URL, using their own security credentials, to grant time-limited permission to download the objects. When you create a pre-signed URL for your object, you must provide your security credentials, specify a bucket name, an object key, specify the HTTP method (GET to download the object) and expiration date and time. The pre-signed URLs are valid only for the specified duration.
Please see this note for more details:



Share an object with others

PDF | Kindle | RSS

All objects by default are private. Only the object owner has permission to access these objects. However, the object owner can optionally share objects with others by creating a presigned URL, using their own security credentials, to grant time-limited permission to download the objects.

When you create a presigned URL for your object, you must provide your security credentials, specify a bucket name, an object key, specify the HTTP method (GET to download the object) and expiration date and time. The presigned URLs are valid only for the specified duration.

Anyone who receives the presigned URL can then access the object. For example, if you have a video in your bucket and both the bucket and the object are private, you can share the video with others by generating a presigned URL.

Note

- Anyone with valid security credentials can create a presigned URL. However, in order to successfully access an object, the presigned URL must be created by someone who has permission to perform the operation that the presigned URL is based upon.
- The credentials that you can use to create a presigned URL include:
  - IAM instance profile: Valid up to 6 hours
  - AWS Security Token Service : Valid up to 36 hours when signed with permanent credentials, such as the credentials of the AWS account root user or an IAM user
  - IAM user: Valid up to 7 days when using AWS Signature Version 4

    To create a presigned URL that's valid for up to 7 days, first designate IAM user credentials (the access key and secret access key) to the SDK that you're using. Then, generate a presigned URL using AWS Signature Version 4.
- If you created a presigned URL using a temporary token, then the URL expires when the token expires, even if the URL was created with a later expiration time.

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

"Using bucket policy" - You can use this policy to limit users from a source IP address however for time-based constraints you are better off using a pre-signed URL.
"Using Routing Policy" - This concept applies to DNS in Route 53, so this option is ruled out.
"Using IAM policy" - You can use IAM policy to grant access toa specific bucket however for time-based constraints you are better off using a pre-signed URL.
Reference:
https://docs.aws.amazon.com/AmazonS3/latest/dev/ShareObjectPreSignedURL.html

Question 56:

A company has configured an Auto Scaling group with health checks. The configuration is set to the desired capacity value of 3 and maximum capacity value of 3. The EC2 instances of your Auto Scaling group are configured to scale when CPU utilization is at 60 percent and is now running at 80 percent utilization.

Which of the following will take place?

- System will keep running as is(Correct)
- The desired capacity will go up to 4 and the maximum capacity will stay at 3
- System will trigger CloudWatch alarms to AWS support
- The desired capacity will go up to 4 and the maximum capacity will also go up to 4

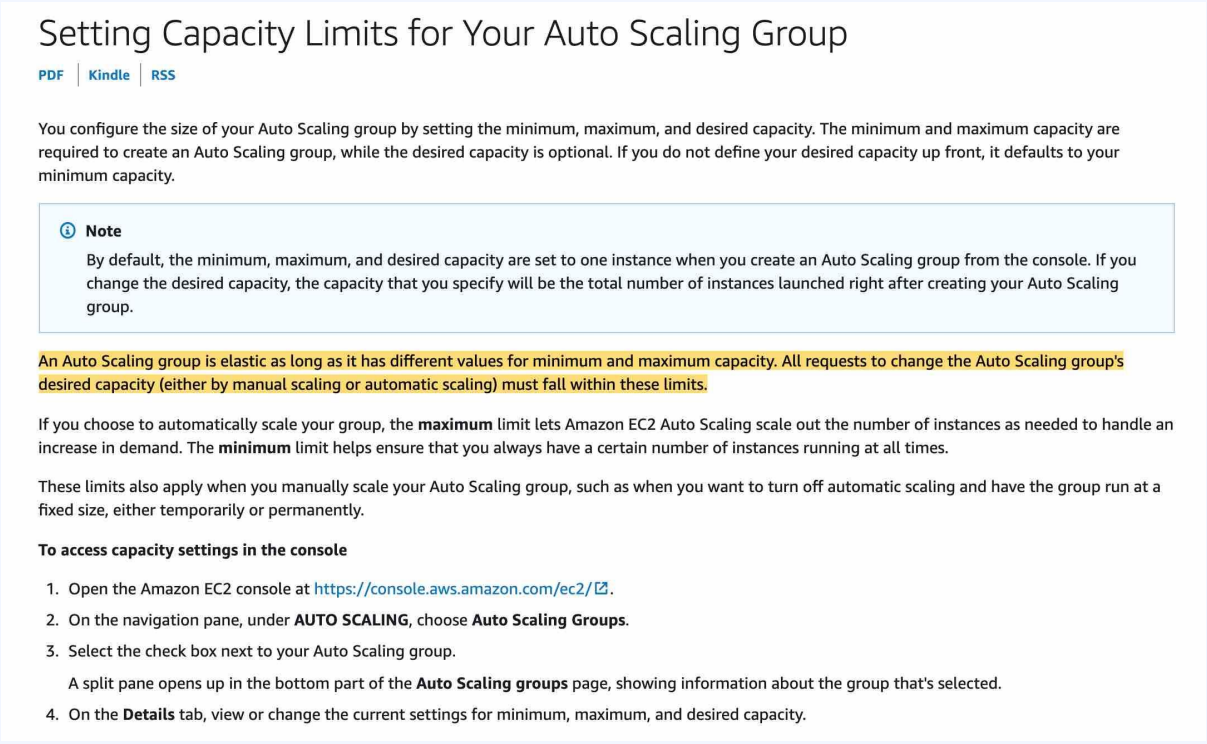
Explanation

Correct option:

System will keep running as is

You are already running at max capacity. After you have created your Auto Scaling group, the Auto Scaling group starts by launching enough EC2 instances to meet its minimum capacity (or its desired capacity, if specified). If there are no other scaling conditions attached to the Auto Scaling group, the Auto Scaling group maintains this number of running instances even if an instance becomes unhealthy.

Setting Capacity Limits for Your Auto Scaling Group:



Setting Capacity Limits for Your Auto Scaling Group

PDF | Kindle | RSS

You configure the size of your Auto Scaling group by setting the minimum, maximum, and desired capacity. The minimum and maximum capacity are required to create an Auto Scaling group, while the desired capacity is optional. If you do not define your desired capacity up front, it defaults to your minimum capacity.

Note

By default, the minimum, maximum, and desired capacity are set to one instance when you create an Auto Scaling group from the console. If you change the desired capacity, the capacity that you specify will be the total number of instances launched right after creating your Auto Scaling group.

An Auto Scaling group is elastic as long as it has different values for minimum and maximum capacity. All requests to change the Auto Scaling group's desired capacity (either by manual scaling or automatic scaling) must fall within these limits.

If you choose to automatically scale your group, the **maximum** limit lets Amazon EC2 Auto Scaling scale out the number of instances as needed to handle an increase in demand. The **minimum** limit helps ensure that you always have a certain number of instances running at all times.

These limits also apply when you manually scale your Auto Scaling group, such as when you want to turn off automatic scaling and have the group run at a fixed size, either temporarily or permanently.

To access capacity settings in the console

1. Open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. On the navigation pane, under **AUTO SCALING**, choose **Auto Scaling Groups**.
3. Select the check box next to your Auto Scaling group.

   A split pane opens up in the bottom part of the **Auto Scaling groups** page, showing information about the group that's selected.
4. On the **Details** tab, view or change the current settings for minimum, maximum, and desired capacity.

via - https://docs.aws.amazon.com/autoscaling/ec2/userguide/asg-capacity-limits.html

Incorrect options:

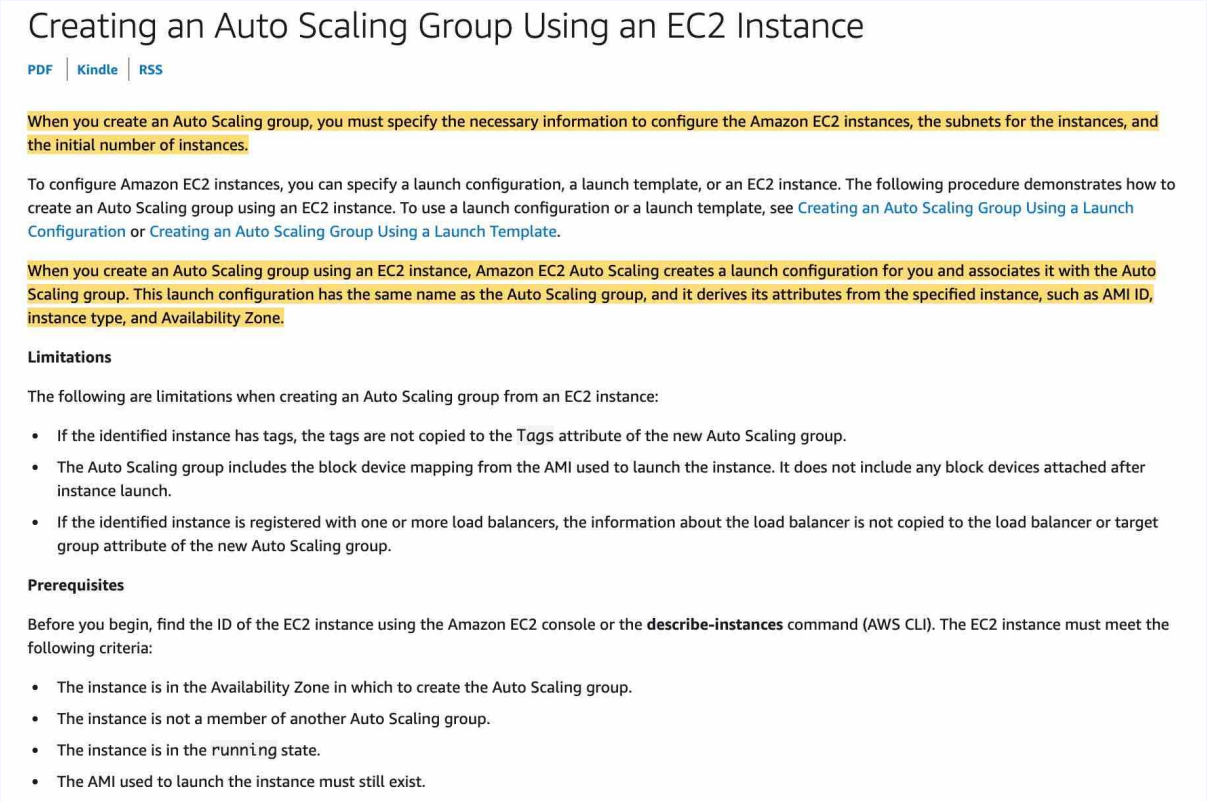
The desired capacity will go up to 4 and the maximum capacity will stay at 3 - The desired capacity cannot go over the maximum capacity.

The desired capacity will go up to 4 and the maximum capacity will also go up to 4 - The maximum capacity cannot change on its own just because the desired capacity has been set to a higher value. You will have to make those changes to the maximum capacity manually.
System will trigger CloudWatch alarms to AWS support - This option has been added as a distractor. You already have alarms configured based on rules but AWS support will not intervene for you.
Reference:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/asg-capacity-limits.html

Question 57:
A new recruit is trying to understand the nuances of EC2 Auto Scaling. As an AWS Certified Developer Associate, you have been asked to mentor the new recruit.
Can you identify and explain the correct statements about Auto Scaling to the new recruit? (Select two).

- Every time you create an Auto Scaling group from an existing instance, it creates a new AMI (Amazon Machine Image)
- Amazon EC2 Auto Scaling works with both Application Load Balancers and Network Load Balancers(Correct)
- You cannot use Amazon EC2 Auto Scaling for health checks (to replace unhealthy instances) if you are not using Elastic Load Balancing (ELB)
- EC2 Auto Scaling groups are regional constructs. They span across Availability Zones and AWS regions
- Amazon EC2 Auto Scaling cannot add a volume to an existing instance if the existing volume is approaching capacity(Correct)

Explanation
Correct options:
Amazon EC2 Auto Scaling is a fully managed service designed to launch or terminate Amazon EC2 instances automatically to help ensure you have the correct number of Amazon EC2 instances available to handle the load for your application.
Amazon EC2 Auto Scaling cannot add a volume to an existing instance if the existing volume is approaching capacity - A volume is attached to a new instance when it is added. Amazon EC2 Auto Scaling doesn't automatically add a volume when the existing one is approaching capacity. You can use the EC2 API to add a volume to an existing instance.
Amazon EC2 Auto Scaling works with both Application Load Balancers and Network Load Balancers - Amazon EC2 Auto Scaling works with Application Load Balancers and Network Load Balancers including their health check feature.
Incorrect options:
EC2 Auto Scaling groups are regional constructs. They span across Availability Zones and AWS regions - This is an incorrect statement. EC2 Auto Scaling groups are regional constructs. They can span Availability Zones, but not AWS regions.
Every time you create an Auto Scaling group from an existing instance, it creates a new AMI (Amazon Machine Image) - This is an incorrect statement. When you create an Auto Scaling group from an existing instance, it does not create a new AMI.



Creating an Auto Scaling Group Using an EC2 Instance

PDF | Kindle | RSS

When you create an Auto Scaling group, you must specify the necessary information to configure the Amazon EC2 instances, the subnets for the instances, and the initial number of instances.

To configure Amazon EC2 instances, you can specify a launch configuration, a launch template, or an EC2 instance. The following procedure demonstrates how to create an Auto Scaling group using an EC2 instance. To use a launch configuration or a launch template, see Creating an Auto Scaling Group Using a Launch Configuration or Creating an Auto Scaling Group Using a Launch Template.

When you create an Auto Scaling group using an EC2 instance, Amazon EC2 Auto Scaling creates a launch configuration for you and associates it with the Auto Scaling group. This launch configuration has the same name as the Auto Scaling group, and it derives its attributes from the specified instance, such as AMI ID, instance type, and Availability Zone.

Limitations

The following are limitations when creating an Auto Scaling group from an EC2 instance:

- If the identified instance has tags, the tags are not copied to the `Tags` attribute of the new Auto Scaling group.
- The Auto Scaling group includes the block device mapping from the AMI used to launch the instance. It does not include any block devices attached after instance launch.
- If the identified instance is registered with one or more load balancers, the information about the load balancer is not copied to the load balancer or target group attribute of the new Auto Scaling group.

Prerequisites

Before you begin, find the ID of the EC2 instance using the Amazon EC2 console or the **describe-instances** command (AWS CLI). The EC2 instance must meet the following criteria:

- The instance is in the Availability Zone in which to create the Auto Scaling group.
- The instance is not a member of another Auto Scaling group.
- The instance is in the `running` state.
- The AMI used to launch the instance must still exist.

via - https://docs.aws.amazon.com/autoscaling/ec2/userguide/create-asg-from-instance.html
You cannot use Amazon EC2 Auto Scaling for health checks (to replace unhealthy instances) if you are not using Elastic Load Balancing (ELB) - This is an incorrect statement. You don't have to use ELB to use Auto Scaling. You can use the EC2 health check to identify and replace unhealthy instances.
References:
https://aws.amazon.com/ec2/autoscaling/faqs/
https://docs.aws.amazon.com/autoscaling/ec2/userguide/create-asg-from-instance.html

Question 58:
A developer at a university is encrypting a large XML payload transferred over the network using AWS KMS and wants to test the application before going to production.
What is the maximum data size supported by AWS KMS?

- 10MB
- 4KB(Correct)
- 16KB
- 1MB

Explanation
Correct option:
4 KB
You can encrypt up to 4 kilobytes (4096 bytes) of arbitrary data such as an RSA key, a database password, or other sensitive information.
While AWS KMS does support sending data up to 4 KB to be encrypted directly, envelope encryption can offer significant performance benefits. When you encrypt data directly with AWS KMS it must be transferred over the network. Envelope encryption reduces the network load since only the request and delivery of the much smaller data key go over the network. The data key is used locally in your application or encrypting AWS service, avoiding the need to send the entire block of data to AWS KMS and suffer network latency.
Incorrect options:
1MB - For anything over 4 KB, you may want to look at envelope encryption
10MB - For anything over 4 KB, you may want to look at envelope encryption
16KB - For anything over 4 KB, you may want to look at envelope encryption
Reference:
https://aws.amazon.com/kms/faqs/

Question 59:
An e-commerce application writes log files into Amazon S3. The application also reads these log files in parallel on a near real-time basis. The development team wants to address any data discrepancies that might arise when the application overwrites an existing log file and then tries to read that specific log file.
Which of the following options BEST describes the capabilities of Amazon S3 relevant to this scenario?

- A process replaces an existing object and immediately tries to read it. Amazon S3 always returns the latest version of the object(Correct)
- A process replaces an existing object and immediately tries to read it. Until the change is fully propagated, Amazon S3 might return the previous data
- A process replaces an existing object and immediately tries to read it. Until the change is fully propagated, Amazon S3 might return the new data
- A process replaces an existing object and immediately tries to read it. Until the change is fully propagated, Amazon S3 does not return any data

Explanation
Correct option:
A process replaces an existing object and immediately tries to read it. Amazon S3 always returns the latest version of the object
Amazon S3 delivers strong read-after-write consistency automatically, without changes to performance or availability, without sacrificing regional isolation for applications, and at no additional cost.
After a successful write of a new object or an overwrite of an existing object, any subsequent read request immediately receives the latest version of the object. S3 also provides strong consistency for list operations, so after a write, you can immediately perform a listing of the objects in a bucket with any changes reflected.
Strong read-after-write consistency helps when you need to immediately read an object after a write. For example, strong read-after-write consistency when you often read and list immediately after writing objects.
To summarize, all S3 GET, PUT, and LIST operations, as well as operations that change object tags, ACLs, or metadata, are strongly consistent. What you write is what you will read, and the results of a LIST will be an accurate reflection of what's in the bucket.
Incorrect options:
A process replaces an existing object and immediately tries to read it. Until the change is fully propagated, Amazon S3 might return the previous data
A process replaces an existing object and immediately tries to read it. Until the change is fully propagated, Amazon S3 does not return any data
A process replaces an existing object and immediately tries to read it. Until the change is fully propagated, Amazon S3 might return the new data
These three options contradict the earlier details provided in the explanation.
References:
https://docs.aws.amazon.com/AmazonS3/latest/dev/Introduction.html#ConsistencyModel

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

Question 60:
An investment firm wants to continuously generate time-series analytics of the stocks being purchased by its customers. The firm wants to build a live leaderboard with near-real-time analytics for these in-demand stocks.
Which of the following represents a fully managed solution with the least cost to address this use-case?

- Use Kinesis Data Streams to ingest data and Amazon Kinesis Client Library to the application logic to generate leaderboard scores and time-series analytics
- Use Kinesis Firehose to ingest data and Kinesis Data Analytics to generate leaderboard scores and time-series analytics(Correct)
- Use Kinesis Firehose to ingest data and Amazon Athena to generate leaderboard scores and time-series analytics
- Use Kinesis Data Streams to ingest data and Kinesis Data Analytics to generate leaderboard scores and time-series analytics

Explanation
Correct option:
Use Kinesis Firehose to ingest data and Kinesis Data Analytics to generate leaderboard scores and time-series analytics
Amazon Kinesis Data Firehose is the easiest way to reliably load streaming data into data lakes, data stores, and analytics services. It can capture, transform, and deliver streaming data to Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, generic HTTP endpoints, and service providers like Datadog, New Relic, MongoDB, and Splunk. It is a fully managed service that automatically scales to match the throughput of your data and requires no ongoing administration. It can also batch, compress, transform, and encrypt your data streams before loading, minimizing the amount of storage used and increasing security.
Amazon Kinesis Data Analytics is the easiest way to transform and analyze streaming data in real-time with Apache Flink. Apache Flink is an open source framework and engine for processing data streams. Amazon Kinesis Data Analytics reduces the complexity of building, managing, and integrating Apache Flink applications with other AWS services.
Amazon Kinesis Data Analytics provides built-in functions to filter, aggregate, and transform streaming data for advanced analytics. It processes streaming data with sub-second latencies, enabling you to analyze and respond to incoming data and events in real-time.
Amazon Kinesis Data Analytics is serverless; there are no servers to manage. It runs your streaming applications without requiring you to provision or manage any infrastructure. Amazon Kinesis Data Analytics automatically scales the infrastructure up and down as required to process incoming data.
Incorrect options:
Use Kinesis Data Streams to ingest data and Kinesis Data Analytics to generate leaderboard scores and time-series analytics - Amazon Kinesis Data Streams (KDS) is a massively scalable and durable real-time data streaming service. KDS can continuously capture gigabytes of data per second from hundreds of thousands of sources such as website clickstreams, database event streams, financial transactions, social media feeds, IT logs, and location-tracking events. The data collected is available

in milliseconds to enable real-time analytics use cases such as real-time dashboards, real-time anomaly detection, dynamic pricing, and more.
Although Kinesis Data Streams supports on-demand provisioning of shards, however, the data ingestion cost along with the per hour shards cost would be more than the corresponding cost incurred while using Firehose. The use-case clearly states that the company wants a fully managed solution with the least cost, so Kinesis Firehose is a better solution.
Use Kinesis Data Streams to ingest data and Amazon Kinesis Client Library to the application logic to generate leaderboard scores and time-series analytics - The Amazon Kinesis Client Library (KCL) is a pre-built library that helps you build consumer applications for reading and processing data from an Amazon Kinesis data stream. The KCL handles complex issues such as adapting to changes in data stream volume, load balancing streaming data, coordinating distributed services, and processing data with fault-tolerance. The KCL enables you to focus on business logic while building applications.
If you want a fully managed solution and you want to use SQL to process the data from your data stream, you should use Kinesis Data Analytics. Use KCL if you need to build a custom processing solution whose requirements are not met by Kinesis Data Analytics, and you can manage the resulting consumer application.
Use Kinesis Firehose to ingest data and Amazon Athena to generate leaderboard scores and time-series analytics - Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run. Athena is used for running analytics on S3 based data. For running analytics on real-time streaming data, Kinesis Data Analytics is the right fit.
References:
https://docs.aws.amazon.com/firehose/latest/dev/data-analysis.html
https://aws.amazon.com/kinesis/data-analytics/faqs/
https://aws.amazon.com/kinesis/data-firehose/pricing/
https://aws.amazon.com/kinesis/data-streams/pricing/

Question 61:
You are a developer working at a cloud company that embraces serverless. You have performed your initial deployment and would like to work towards adding API Gateway stages and associate them with existing deployments. Your stages will include prod, test, and dev and will need to match a Lambda function variant that can be updated over time.
Which of the following features must you add to achieve this? (select two)

- Stage Variables(Correct)
- Lambda Aliases(Correct)
- Lambda X-Ray integration
- Lambda Versions
- Mapping Templates

Explanation
Correct options:
Stage Variables
Stage variables are name-value pairs that you can define as configuration attributes associated with a deployment stage of an API. They act like environment variables and can be used in your API setup and mapping templates. With deployment stages in API Gateway, you can manage multiple release stages for each API, such as alpha, beta, and production. Using stage variables you can configure an API deployment stage to interact with different backend endpoints.
For example, your API can pass a GET request as an HTTP proxy to the backend web host (for example, http://example.com). In this case, the backend web host is configured in a stage variable so that when developers call your production endpoint, API Gateway calls example.com. When you call your beta endpoint, API Gateway uses the value configured in the stage variable for the beta stage and calls a different web host (for example, beta.example.com).
Lambda Aliases
A Lambda alias is like a pointer to a specific Lambda function version. Users can access the function version using the alias ARN.
Lambda Aliases allow you to create a "mutable" Lambda version that points to whatever version you want in the backend. This allows you to have a "dev", "test", prod" Lambda alias that can remain stable over time.
Incorrect options:
Lambda Versions - Versions are immutable and cannot be updated over time. So this option is not correct.
Lambda X-Ray integration - This is good for tracing and debugging requests so it can be looked at as a good option for troubleshooting issues in the future. This is not the right fit for the given use-case.
Mapping Templates - Mapping template overrides provides you with the flexibility to perform many-to-one parameter mappings; override parameters after standard API Gateway mappings have been applied; conditionally map parameters based on body content or other parameter values; programmatically create new parameters on the fly, and override status codes returned by your integration endpoint. This is not the right fit for the given use-case.
References:
https://docs.aws.amazon.com/apigateway/latest/developerguide/stage-variables.html

https://docs.aws.amazon.com/lambda/latest/dg/configuration-aliases.html
https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-override-request-response-parameters.html

Question 62:
The development team at a health-care company is planning to migrate to AWS Cloud from the on-premises data center. The team is evaluating Amazon RDS as the database tier for its flagship application.
Which of the following would you identify as correct for RDS Multi-AZ? (Select two)

- To enhance read scalability, a Multi-AZ standby instance can be used to serve read requests
- Updates to your DB Instance are asynchronously replicated across the Availability Zone to the standby in order to keep both in sync
- RDS applies OS updates by performing maintenance on the standby, then promoting the standby to primary and finally performing maintenance on the old primary, which becomes the new standby(Correct)
- Amazon RDS automatically initiates a failover to the standby, in case primary database fails for any reason(Correct)
- For automated backups, I/O activity is suspended on your primary DB since backups are not taken from standby DB

Explanation
Correct options:
RDS applies OS updates by performing maintenance on the standby, then promoting the standby to primary, and finally performing maintenance on the old primary, which becomes the new standby
Running a DB instance as a Multi-AZ deployment can further reduce the impact of a maintenance event because Amazon RDS applies operating system updates by following these steps:
Perform maintenance on the standby.
Promote the standby to primary.
Perform maintenance on the old primary, which becomes the new standby.
When you modify the database engine for your DB instance in a Multi-AZ deployment, then Amazon RDS upgrades both the primary and secondary DB instances at the same time. In this case, the database engine for the entire Multi-AZ deployment is shut down during the upgrade.
Amazon RDS automatically initiates a failover to the standby, in case the primary database fails for any reason - You also benefit from enhanced database availability when running your DB instance as a Multi-AZ deployment. If an Availability Zone failure or DB instance failure occurs, your availability impact is limited to the time automatic failover takes to complete.
Another implied benefit of running your DB instance as a Multi-AZ deployment is that DB instance failover is automatic and requires no administration. In an Amazon RDS context, this means you are not required to monitor DB instance events and initiate manual DB instance recovery in the event of an Availability Zone failure or DB instance failure.
Incorrect options:
For automated backups, I/O activity is suspended on your primary DB since backups are not taken from standby DB - The availability benefits of Multi-AZ also extend to planned maintenance. For example, with automated backups, I/O activity is no longer suspended on your primary during your preferred backup window, since backups are taken from the standby.
To enhance read scalability, a Multi-AZ standby instance can be used to serve read requests - A Multi-AZ standby cannot serve read requests. Multi-AZ deployments are designed to provide

enhanced database availability and durability, rather than read scaling benefits. As such, the feature uses synchronous replication between primary and standby. AWS implementation makes sure the primary and the standby are constantly in sync, but precludes using the standby for read or write operations.
Updates to your DB Instance are asynchronously replicated across the Availability Zone to the standby in order to keep both in sync - When you create your DB instance to run as a Multi-AZ deployment, Amazon RDS automatically provisions and maintains a synchronous “standby” replica in a different Availability Zone. Updates to your DB Instance are synchronously replicated across the Availability Zone to the standby in order to keep both in sync and protect your latest database updates against DB instance failure.
Reference:
https://aws.amazon.com/rds/faqs/

Question 63:
A multi-national company maintains separate AWS accounts for different verticals in their organization. The project manager of a team wants to migrate the Elastic Beanstalk environment from Team A's AWS account into Team B's AWS account. As a Developer, you have been roped in to help him in this process.
Which of the following will you suggest?

- It is not possible to migrate Elastic Beanstalk environment from one AWS account to the other
- Create a saved configuration in Team A's account and download it to your local machine. Make the account-specific parameter changes and upload to the S3 bucket in Team B's account. From Elastic Beanstalk console, create an application from 'Saved Configurations'(Correct)
- Create a saved configuration in Team A's account and configure it to Export. Now, log into Team B's account and choose the Import option. Here, you need to specify the name of the saved configuration and allow the system to create the new application. This takes a little time based on the Regions the two accounts belong to
- Create an export configuration from the Elastic Beanstalk console from Team A's account. This configuration has to be shared with the IAM Role of Team B's account. The import option of Team B's account will show the saved configuration, that can be used to create a new Beanstalk application

Explanation
Correct option:
Create a saved configuration in Team A's account and download it to your local machine. Make the account-specific parameter changes and upload to the S3 bucket in Team B's account. From Elastic Beanstalk console, create an application from 'Saved Configurations - You must use saved configurations to migrate an Elastic Beanstalk environment between AWS accounts. You can save your environment's configuration as an object in Amazon Simple Storage Service (Amazon S3) that can be applied to other environments during environment creation, or applied to a running environment. Saved configurations are YAML formatted templates that define an environment's platform version, tier, configuration option settings, and tags.
Download the saved configuration to your local machine. Change your account-specific parameters in the downloaded configuration file, and then save the changes. For example, change the key pair name, subnet ID, or application name (such as application-b-name). Upload the saved configuration from your local machine to an S3 bucket in Team B's account. From this account, create a new Beanstalk application by choosing 'Saved Configurations' from the navigation panel.
Incorrect options:
Create a saved configuration in Team A's account and configure it to Export. Now, log into Team B's account and choose the Import option. Here, you need to specify the name of the saved configuration and allow the system to create the new application. This takes a little time based on the Regions the two accounts belong to - There is no direct Export and Import option for migrating Elastic Beanstalk configurations.

It is not possible to migrate Elastic Beanstalk environment from one AWS account to the other - This is an incorrect statement.
Create an export configuration from the Elastic Beanstalk console from Team A's account. This configuration has to be shared with the IAM Role of Team B's account. The import option of the Team B's account will show the saved configuration, that can be used to create a new Beanstalk application - This contradicts the explanation provided earlier.
References:
https://aws.amazon.com/premiumsupport/knowledge-center/elastic-beanstalk-migration-accounts/
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/environment-configuration-savedconfig.html

Question 64:

An analytics company is using Kinesis Data Streams (KDS) to process automobile health-status data from the taxis managed by a taxi ride-hailing service. Multiple consumer applications are using the incoming data streams and the engineers have noticed a performance lag for the data delivery speed between producers and consumers of the data streams.

As a Developer Associate, which of the following options would you suggest for improving the performance for the given use-case?

- Swap out Kinesis Data Streams with SQS FIFO queues
- Swap out Kinesis Data Streams with Kinesis Data Firehose
- Use Enhanced Fanout feature of Kinesis Data Streams(Correct)
- Swap out Kinesis Data Streams with SQS Standard queues

Explanation

Correct option:

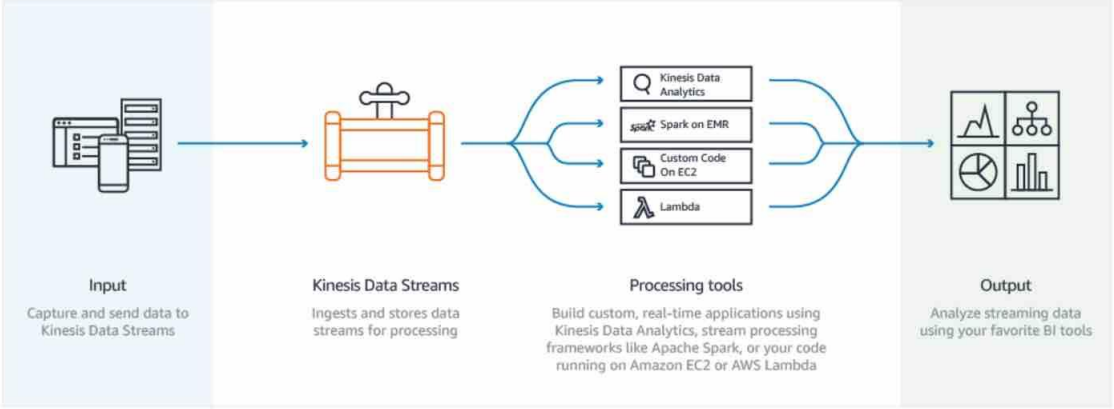
Use Enhanced Fanout feature of Kinesis Data Streams

Amazon Kinesis Data Streams (KDS) is a massively scalable and durable real-time data streaming service. KDS can continuously capture gigabytes of data per second from hundreds of thousands of sources such as website clickstreams, database event streams, financial transactions, social media feeds, IT logs, and location-tracking events.

By default, the 2MB/second/shard output is shared between all of the applications consuming data from the stream. You should use enhanced fan-out if you have multiple consumers retrieving data from a stream in parallel. With enhanced fan-out developers can register stream consumers to use enhanced fan-out and receive their own 2MB/second pipe of read throughput per shard, and this throughput automatically scales with the number of shards in a stream.

Kinesis Data Streams Fanout

Kinesis actually refers to a family of streaming services: Kinesis Video Streams, Kinesis Data Firehose, Kinesis Data Analytics, and the topic of today's blog post, Kinesis Data Streams (KDS). Kinesis Data Streams allows developers to easily and continuously collect, process, and analyze streaming data in real-time with a fully-managed and massively scalable service. KDS can capture gigabytes of data per second from hundreds of thousands of sources – everything from website clickstreams and social media feeds to financial transactions and location-tracking events.



Kinesis Data Streams are scaled using the concept of a **shard**. One shard provides an ingest capacity of 1MB/second or 1000 records/second and an output capacity of 2MB/second. It's not uncommon for customers to have thousands or tens of thousands of shards supporting 10s of GB/sec of ingest and egress. Before the enhanced fan-out capability, that 2MB/second/shard output was shared between all of the applications consuming data from the stream. With enhanced fan-out developers can register stream consumers to use enhanced fan-out and receive their own 2MB/second pipe of read throughput per shard, and this throughput automatically scales with the number of shards in a stream. Prior to the launch of Enhanced Fan-out customers would frequently fan-out their data out to multiple streams to support their desired read throughput for their downstream applications. That sounds like undifferentiated heavy lifting to us, and that's something we decided our customers shouldn't need to worry about. Customers pay for enhanced fan-out based on the amount of data retrieved from the stream using enhanced fan-out and the number of consumers registered per-shard. You can find additional info on the pricing page.

via - https://aws.amazon.com/blogs/aws/kds-enhanced-fanout/

Incorrect options:

Swap out Kinesis Data Streams with Kinesis Data Firehose - Amazon Kinesis Data Firehose is the easiest way to reliably load streaming data into data lakes, data stores, and analytics tools. It is a fully managed service that automatically scales to match the throughput of your data and requires no ongoing administration. It can also batch, compress, transform, and encrypt the data before loading it, minimizing the amount of storage used at the destination and increasing security. Kinesis Data Firehose can only write to S3, Redshift, Elasticsearch or Splunk. You can't have applications consuming data streams from Kinesis Data Firehose, that's the job of Kinesis Data Streams. Therefore this option is not correct.

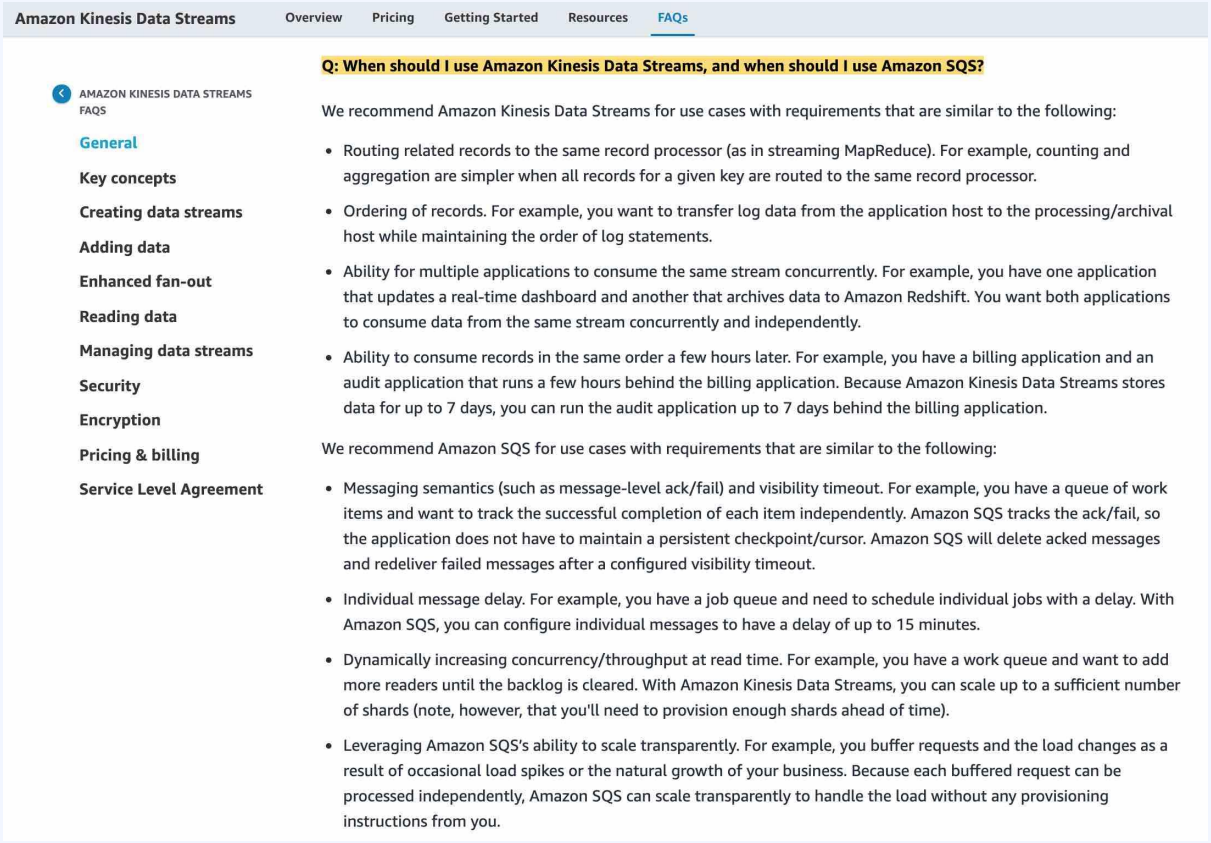
Swap out Kinesis Data Streams with SQS Standard queues

Swap out Kinesis Data Streams with SQS FIFO queues

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. As multiple applications are consuming the same stream concurrently, both SQS Standard and SQS FIFO are not the right fit for the given use-case.

Exam Alert:

Please understand the differences between the capabilities of Kinesis Data Streams vs SQS, as you may be asked scenario-based questions on this topic in the exam.



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

References:

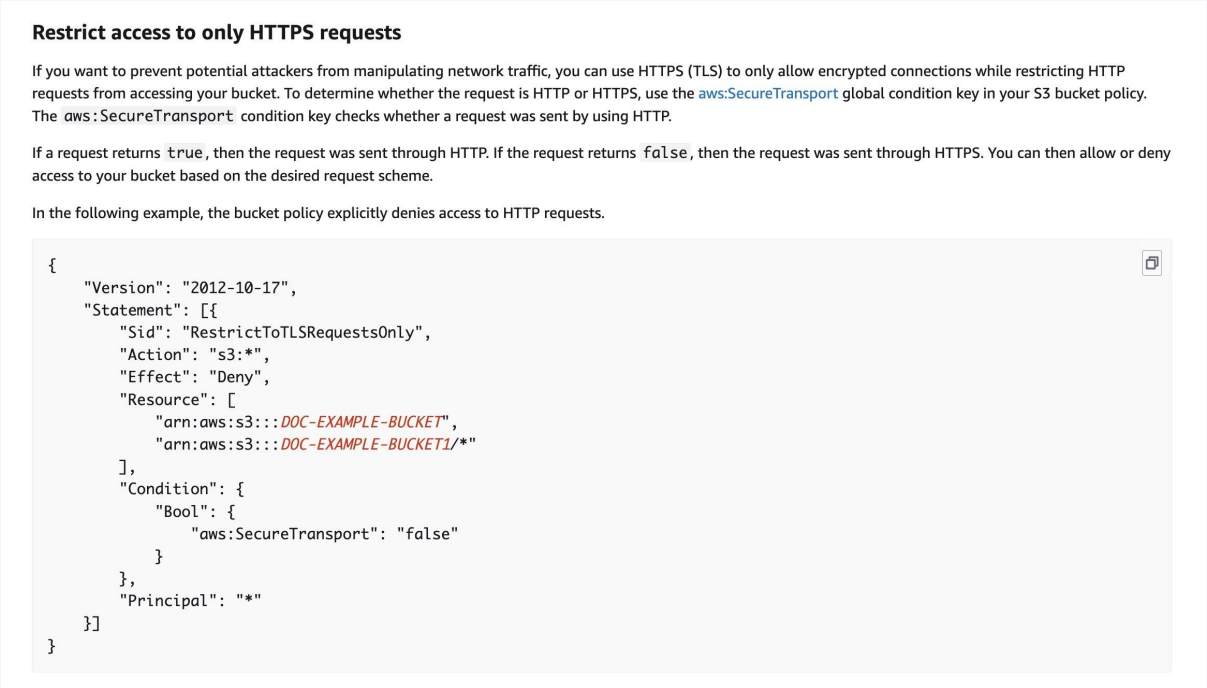
https://aws.amazon.com/blogs/aws/kds-enhanced-fanout/

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

Question 65:
A company has sensitive data stored in an Amazon S3 bucket that is encrypted using AWS Key Management Service (AWS KMS). A developer wants to enforce encryption in transit for all users who have been granted permission to use the S3 GetObject operation across multiple AWS accounts. Which of the following represents the best solution for this use case?

- Configure a resource-based policy on the KMS key to allow access when a request has the condition "aws:SecureTransport": "false"
- Configure a resource-based policy on the S3 bucket to allow access when a request has the condition "aws:SecureTransport": "false"
- Configure a resource-based policy on the S3 bucket to deny access when a request has the condition "aws:SecureTransport": "false"(Correct)
- Configure a resource-based policy on the KMS key to deny access when a request has the condition "aws:SecureTransport": "false"

Explanation
Correct option:
Configure a resource-based policy on the S3 bucket to deny access when a request has the condition "aws:SecureTransport": "false"
If you want to prevent potential attackers from manipulating network traffic, you can use HTTPS (TLS) to only allow encrypted connections while restricting HTTP requests from accessing your bucket. To determine whether the request is HTTP or HTTPS, use the aws:SecureTransport global condition key in your S3 bucket policy. The aws:SecureTransport condition key checks whether a request was sent by using HTTP.



Restrict access to only HTTPS requests

If you want to prevent potential attackers from manipulating network traffic, you can use HTTPS (TLS) to only allow encrypted connections while restricting HTTP requests from accessing your bucket. To determine whether the request is HTTP or HTTPS, use the aws:SecureTransport global condition key in your S3 bucket policy. The `aws:SecureTransport` condition key checks whether a request was sent by using HTTP.

If a request returns `true`, then the request was sent through HTTP. If the request returns `false`, then the request was sent through HTTPS. You can then allow or deny access to your bucket based on the desired request scheme.

In the following example, the bucket policy explicitly denies access to HTTP requests.

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Sid": "RestrictToTLSRequestsOnly",
        "Action": "s3:*",
        "Effect": "Deny",
        "Resource": [
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET",
            "arn:aws:s3:::DOC-EXAMPLE-BUCKET1/*"
        ],
        "Condition": {
            "Bool": {
                "aws:SecureTransport": "false"
            }
        },
        "Principal": "*"
    }]
}
```

via - https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html
Incorrect options:
Configure a resource-based policy on the S3 bucket to allow access when a request has the condition "aws:SecureTransport": "false" - This option contradicts the explanation provided above.

Configure a resource-based policy on the KMS key to allow access when a request has the condition "aws:SecureTransport": "false"
Configure a resource-based policy on the KMS key to deny access when a request has the condition "aws:SecureTransport": "false"
Since the use case is about granting permission to use the S3 GetObject operations with encryption in transit, you cannot use a Resource-based policy for KMS. So, both these options are incorrect.
References:
https://aws.amazon.com/blogs/security/how-to-use-bucket-policies-and-apply-defense-in-depth-to-help-secure-your-amazon-s3-data/
https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html