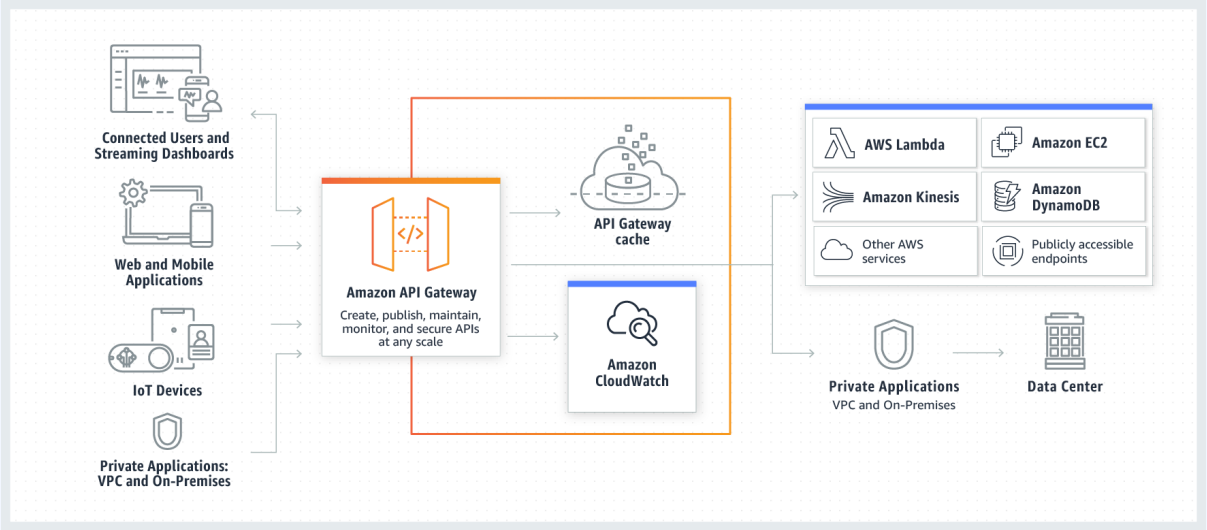
Question 1:
You've just deployed an AWS Lambda function. The lambda function will be invoked via the API Gateway. The API Gateway will need to control access to it.
Which of the following mechanisms is not supported for API Gateway?

- IAM permissions with sigv4
- STS(Correct)
- Cognito User Pools
- Lambda Authorizer

Explanation
Correct option:
Amazon API Gateway is an AWS service for creating, publishing, maintaining, monitoring, and securing REST, HTTP, and WebSocket APIs at any scale. API developers can create APIs that access AWS or other web services, as well as data stored in the AWS Cloud.
How API Gateway Works:

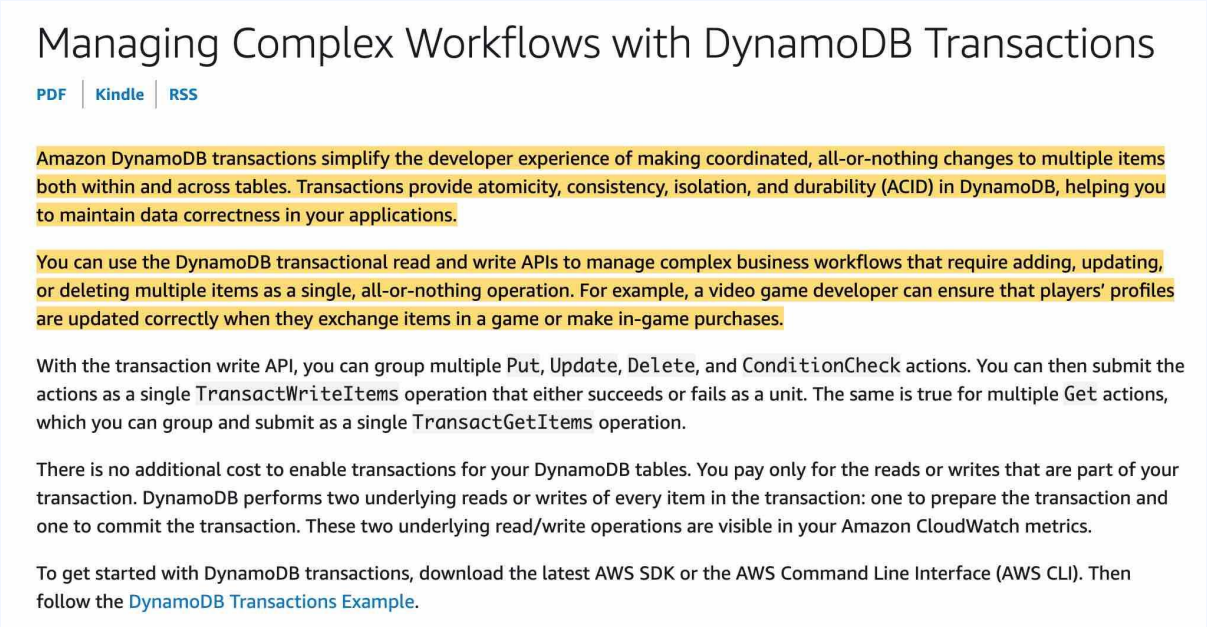


via - https://aws.amazon.com/api-gateway/
STS
The AWS Security Token Service (STS) is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users). However, is not supported at the time with API Gateway.
Incorrect options:
IAM permissions with sigv4 - They can be applied to an entire API or individual methods.
Lambda Authorizer - Control access to REST API methods using bearer token authentication as well as information described by headers, paths, query strings, stage variables, or context variables request parameter.
Cognito User Pools - Use Cognito User Pools to create customizable authentication and authorization solutions for your REST APIs.
Reference:
https://aws.amazon.com/api-gateway/

Question 2:
You are implementing a banking application in which you need to update the Exchanges DynamoDB table and the AccountBalance DynamoDB table at the same time or not at all.
Which DynamoDB feature should you use?

- DynamoDB Transactions
- (Correct)
- DynamoDB TTL
- DynamoDB Indexes
- DynamoDB Streams

Explanation
Correct option:
DynamoDB Transactions
You can use DynamoDB transactions to make coordinated all-or-nothing changes to multiple items both within and across tables. Transactions provide atomicity, consistency, isolation, and durability (ACID) in DynamoDB, helping you to maintain data correctness in your applications.
DynamoDB Transactions Overview:



Managing Complex Workflows with DynamoDB Transactions

PDF | Kindle | RSS

Amazon DynamoDB transactions simplify the developer experience of making coordinated, all-or-nothing changes to multiple items both within and across tables. Transactions provide atomicity, consistency, isolation, and durability (ACID) in DynamoDB, helping you to maintain data correctness in your applications.

You can use the DynamoDB transactional read and write APIs to manage complex business workflows that require adding, updating, or deleting multiple items as a single, all-or-nothing operation. For example, a video game developer can ensure that players' profiles are updated correctly when they exchange items in a game or make in-game purchases.

With the transaction write API, you can group multiple `Put`, `Update`, `Delete`, and `ConditionCheck` actions. You can then submit the actions as a single `TransactWriteItems` operation that either succeeds or fails as a unit. The same is true for multiple `Get` actions, which you can group and submit as a single `TransactGetItems` operation.

There is no additional cost to enable transactions for your DynamoDB tables. You pay only for the reads or writes that are part of your transaction. DynamoDB performs two underlying reads or writes of every item in the transaction: one to prepare the transaction and one to commit the transaction. These two underlying read/write operations are visible in your Amazon CloudWatch metrics.

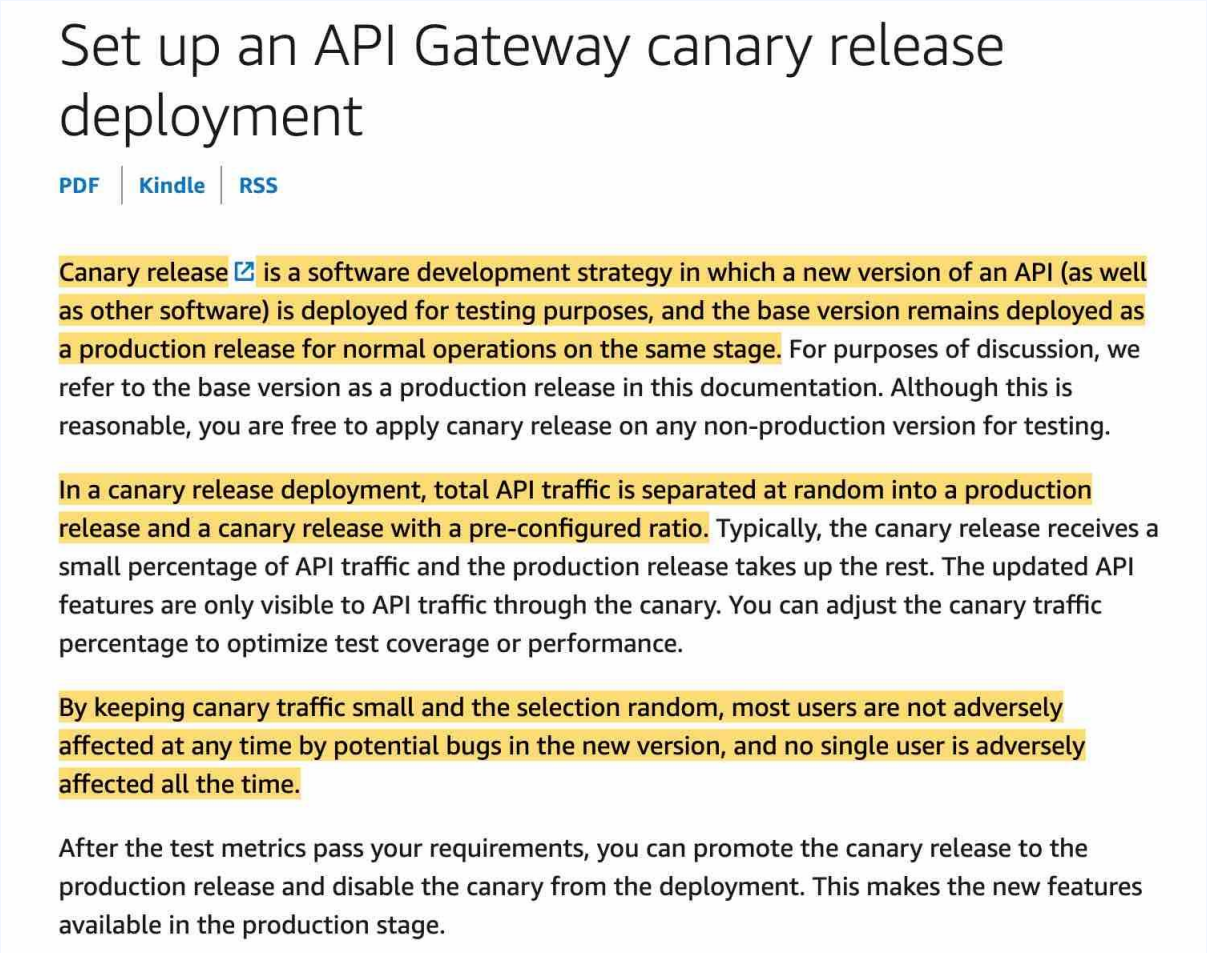
To get started with DynamoDB transactions, download the latest AWS SDK or the AWS Command Line Interface (AWS CLI). Then follow the DynamoDB Transactions Example.

Reference:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguivia -
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/transactions.html
Incorrect options:
DynamoDB TTL - DynamoDB TTL allows you to expire data based on a timestamp, so this option is not correct.
DynamoDB Streams - DynamoDB Streams gives a changelog of changes that happened to your tables and then may even relay these to a Lambda function for further processing.
DynamoDB Indexes - GSI and LSI are used to allow you to query your tables using different partition/sort keys.
de/transactions.html

Question 3:
You are a developer working on AWS Lambda functions that are triggered by Amazon API Gateway and would like to perform testing on a low volume of traffic for new API versions.
Which of the following features will accomplish this task?

- Mapping Templates
- Canary Deployment(Correct)
- Custom Authorizers
- Stage Variables

Explanation
Correct option:
Canary Deployment
In a canary release deployment, total API traffic is separated at random into a production release and a canary release with a preconfigured ratio. Typically, the canary release receives a small percentage of API traffic and the production release takes up the rest. The updated API features are only visible to API traffic through the canary. You can adjust the canary traffic percentage to optimize test coverage or performance.



Set up an API Gateway canary release deployment

PDF | Kindle | RSS

Canary release is a software development strategy in which a new version of an API (as well as other software) is deployed for testing purposes, and the base version remains deployed as a production release for normal operations on the same stage. For purposes of discussion, we refer to the base version as a production release in this documentation. Although this is reasonable, you are free to apply canary release on any non-production version for testing.

In a canary release deployment, total API traffic is separated at random into a production release and a canary release with a pre-configured ratio. Typically, the canary release receives a small percentage of API traffic and the production release takes up the rest. The updated API features are only visible to API traffic through the canary. You can adjust the canary traffic percentage to optimize test coverage or performance.

By keeping canary traffic small and the selection random, most users are not adversely affected at any time by potential bugs in the new version, and no single user is adversely affected all the time.

After the test metrics pass your requirements, you can promote the canary release to the production release and disable the canary from the deployment. This makes the new features available in the production stage.

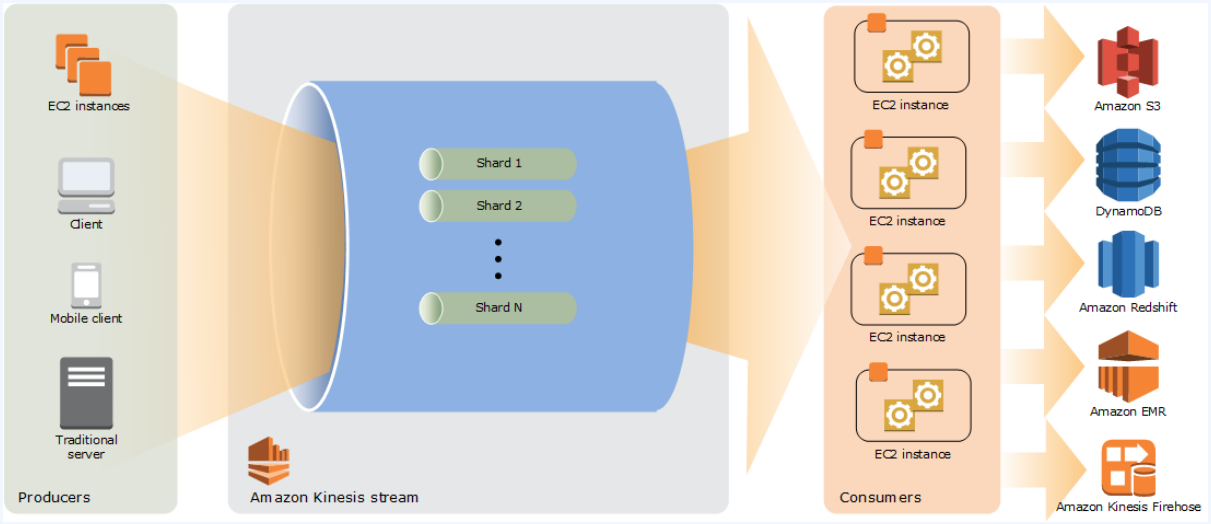
via - https://docs.aws.amazon.com/apigateway/latest/developerguide/canary-release.html
Incorrect options:
Stage Variables - They act like environment variables and can be used in your API setup.

Mapping Templates - Its a script to map the payload from a method request to the corresponding integration request and also maps the integration response to the corresponding method response.
Custom Authorizers - Used for authentication purposes and must return AWS Identity and Access Management (IAM) policies.
Reference:
https://docs.aws.amazon.com/apigateway/latest/developerguide/canary-release.html

Question 4:
One of your Kinesis Stream is experiencing increased traffic due to a sale day. Therefore your Kinesis Administrator has split shards and thus you went from having 6 shards to having 10 shards in your Kinesis Stream. Your consuming application is running a KCL-based application on EC2 instances. What is the maximum number of EC2 instances that can be deployed to process the shards?

- 1
- 10(Correct)
- 6
- 20

Explanation
Correct option:
10
Amazon Kinesis Data Streams enables you to build custom applications that process or analyze streaming data for specialized needs.
A Kinesis data stream is a set of shards. A shard is a uniquely identified sequence of data records in a stream. A stream is composed of one or more shards, each of which provides a fixed unit of capacity.
Kinesis Data Streams Overview:



via - https://docs.aws.amazon.com/streams/latest/dev/key-concepts.html
Each KCL consumer application instance uses "workers" to process data in Kinesis shards. At any given time, each shard of data records is bound to a particular worker via a lease. For the given use-case, an EC2 instance acts as the worker for the KCL application. You can have at most one EC2 instance per shard in Kinesis for the given application. As we have 10 shards, the max number of EC2 instances is 10.

- **KCL consumer application** – an application that is custom-built using KCL and designed to get records from KDS data streams and process them. You can build distributed KCL consumer applications with one or more **consumer application instances** running simultaneously in order to coordinate on failures and dynamically load balance data record processing.
- **Worker** – a high level class that a KCL consumer application instance uses to start processing data. Each KCL consumer application instance has one worker. The worker initializes and oversees various tasks, including syncing shard and lease information, tracking shard assignments, and processing data from the shards. A worker provides KCL with the configuration information for the consumer application, such as the name of the data stream whose data records this KCL consumer application is going to process and the AWS credentials that are needed to access this data stream. The worker also kick starts that specific KCL consumer application instance to deliver data records from the data stream to the record processors.

  > **Important**
  >
  > In KCL 1.x this class is called **Worker**. For more information, see https://github.com/awslabs/amazon-kinesis-client/blob/v1.x/src/main/java/com/amazonaws/services/kinesis/clientlibrary/lib/worker/Worker.java. In KCL 2.x, this class is called **Scheduler**. Scheduler's purpose in KCL 2.x is identical to Worker's purpose in KCL 1.x. For more information about the Scheduler class in KCL 2.x, see https://github.com/awslabs/amazon-kinesis-client/blob/master/amazon-kinesis-client/src/main/java/software/amazon/kinesis/coordinator/Scheduler.java.

- **Lease** – data that defines the binding between a worker and a shard. Distributed KCL consumer applications use leases to partition data record processing across a fleet of workers. At any given time, each shard of data records is bound to a particular worker by a lease identified by the **leaseKey** variable.

  By default, a worker can hold one or more leases (subject to the value of the **maxLeasesForWorker** variable) at the same time.

  > **Important**
  >
  > Every worker will contend to hold all available leases for all available shards in a data stream. But only one worker will successfully hold each lease at any one time.

  For example, if you have a consumer application instance A with worker A that is processing a data stream with 4 shards, worker A can hold leases to shards 1, 2, 3, and 4 at the same time. But if you have two consumer application instances: A and B with worker A and worker B, and these instances are processing a data stream with 4 shards, worker A and worker B cannot both hold the lease to shard 1 at the same time. One worker holds the lease to a particular shard until it is ready to stop processing this shard's data records or until it fails. When one worker stops holding the lease, another worker takes up and holds the lease.

  For more information, see https://github.com/awslabs/amazon-kinesis-client/blob/v1.x/src/main/java/com/amazonaws/services/kinesis/leases/impl/Lease.java for KCL 1.x and https://github.com/awslabs/amazon-kinesis-client/blob/master/amazon-kinesis-client/src/main/java/software/amazon/kinesis/leases/Lease.java for KCL 2.x.
- **Record processor** – this is the logic that defines how your KCL consumer application processes the data that it gets from KDS data streams. At runtime, a KCL consumer application instance instantiates a worker, and this worker instantiates one record processor for every shard to which it holds a lease.

via - https://docs.aws.amazon.com/streams/latest/dev/shared-throughput-kcl-consumers.html

Incorrect options:

1

6

20

These three options contradict the explanation provided earlier. So these are incorrect.

Reference:

https://docs.aws.amazon.com/streams/latest/dev/developing-consumers-with-kcl.html

Question 5:
You would like to run the X-Ray daemon for your Docker containers deployed using AWS Fargate. What do you need to do to ensure the setup will work? (Select two)

- Deploy the X-Ray daemon agent as a daemon set on ECS
- Deploy the X-Ray daemon agent as a sidecar container(Correct)
- Provide the correct IAM instance role to the EC2 instance
- Deploy the X-Ray daemon agent as a process on your EC2 instance
- Provide the correct IAM task role to the X-Ray container(Correct)

Explanation
Correct options:
Deploy the X-Ray daemon agent as a sidecar container
In Amazon ECS, create a Docker image that runs the X-Ray daemon, upload it to a Docker image repository, and then deploy it to your Amazon ECS cluster. You can use port mappings and network mode settings in your task definition file to allow your application to communicate with the daemon container.
As we are using AWS Fargate, we do not have control over the underlying EC2 instance and thus we can't deploy the agent on the EC2 instance or run an X-Ray agent container as a daemon set (only available for ECS classic).

In your task definition, the configuration depends on the networking mode that you use. Bridge networking is the default and can be used in your default VPC. In a bridge network, set the AWS_XRAY_DAEMON_ADDRESS environment variable to tell the X-Ray SDK which container-port to reference and set the host port. For example, you could publish UDP port 2000, and create a link from your application container to the daemon container.

Example Task definition

```
{
  "name": "xray-daemon",
  "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/xray-daemon",
  "cpu": 32,
  "memoryReservation": 256,
  "portMappings" : [
      {
          "hostPort": 0,
          "containerPort": 2000,
          "protocol": "udp"
      }
   ],
},
{
  "name": "scorekeep-api",
  "image": "123456789012.dkr.ecr.us-east-2.amazonaws.com/scorekeep-api",
  "cpu": 192,
  "memoryReservation": 512,
  "environment": [
      { "name" : "AWS_REGION", "value" : "us-east-2" },
      { "name" : "NOTIFICATION_TOPIC", "value" : "arn:aws:sns:us-east-2:123456789012:scorekeep-notifications" },
      { "name" : "AWS_XRAY_DAEMON_ADDRESS", "value" : "xray-daemon:2000" }
  ],
  "portMappings" : [
      {
          "hostPort": 5000,
          "containerPort": 5000
      }
  ],
  "links": [
    "xray-daemon"
  ]
}
```

If you run your cluster in the private subnet of a VPC, you can use the awsvpc network mode to attach an elastic network interface (ENI) to your containers. This enables you to avoid using links. Omit the host port in the port mappings, the link, and the AWS_XRAY_DAEMON_ADDRESS environment variable.

via - https://docs.aws.amazon.com/xray/latest/devguide/xray-daemon-ecs.html

Provide the correct IAM task role to the X-Ray container

For Fargate, we can only provide IAM roles to tasks, which is also the best security practice should we use EC2 instances.

Incorrect options:

Deploy the X-Ray daemon agent as a daemon set on ECS - As explained above, since we are using AWS Fargate, we do not have control over the underlying EC2 instance and thus we can't run an X-Ray agent container as a daemon set.

Deploy the X-Ray daemon agent as a process on your EC2 instance

Provide the correct IAM instance role to the EC2 instance

As we are using AWS Fargate, we do not have control over the underlying EC2 instance, so both these options are incorrect.

Reference:

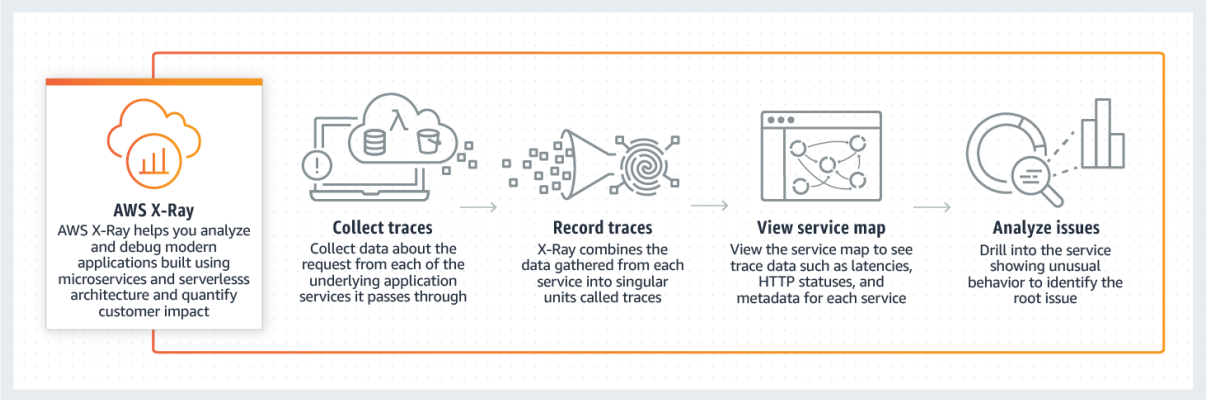
https://docs.aws.amazon.com/xray/latest/devguide/xray-daemon-ecs.html

Question 6:
You have been collecting AWS X-Ray traces across multiple applications and you would now like to index your XRay traces to search and filter through them efficiently.
What should you use in your instrumentation?

- Annotations(Correct)
- Metadata
- Sampling
- Segments

Explanation

Correct option:
Annotations
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 collect data across AWS Accounts. The X-Ray agent can assume a role to publish data into an account different from the one in which it is running. This enables you to publish data from various components of your application into a central account.
How X-Ray Works:



via - https://aws.amazon.com/xray/
Annotations are simple key-value pairs that are indexed for use with filter expressions. Use annotations to record data that you want to use to group traces in the console, or when calling the GetTraceSummaries API.
X-Ray indexes up to 50 annotations per trace.
Incorrect options:
Metadata - Metadata are key-value pairs with values of any type, including objects and lists, but that is not indexed. Use metadata to record data you want to store in the trace but don't need to use for searching traces.
Segments - The computing resources running your application logic send data about their work as segments. A segment provides the resource's name, details about the request, and details about the work done.

Sampling - To ensure efficient tracing and provide a representative sample of the requests that your application serves, the X-Ray SDK applies a sampling algorithm to determine which requests get traced. By default, the X-Ray SDK records the first request each second, and five percent of any additional requests.
Reference:
https://docs.aws.amazon.com/xray/latest/devguide/xray-concepts.html

Question 7:
A popular mobile app retrieves data from an AWS DynamoDB table that was provisioned with read-capacity units (RCU's) that are evenly shared across four partitions. One of those partitions is receiving more traffic than the other partitions, causing hot partition issues.
What technology will allow you to reduce the read traffic on your AWS DynamoDB table with minimal effort?

- ElastiCache
- DynamoDB Streams
- More partitions
- DynamoDB DAX(Correct)

Explanation

Correct option:
DynamoDB DAX
Amazon DynamoDB Accelerator (DAX) is a fully managed, highly available, in-memory cache for DynamoDB that delivers up to a 10x performance improvement: from milliseconds to microseconds: even at millions of requests per second.
Incorrect options:
DynamoDB Streams - A stream record contains information about a data modification to a single item in a DynamoDB table. This is not the correct option for the given use-case.
ElastiCache - ElastiCache can cache the results from anything but you will need to modify your code to check the cache before querying the main query store. As the given use-case mandates minimal effort, so this option is not correct.
More partitions - This option has been added as a distractor as DynamoDB handles that for you automatically.
Reference:
https://aws.amazon.com/dynamodb/dax/

Question 8:
Your company is shifting towards Elastic Container Service (ECS) to deploy applications. The process should be automated using the AWS CLI to create a service where at least ten instances of a task definition are kept running under the default cluster.
Which of the following commands should be executed?

- aws ecr create-service --service-name ecs-simple-service --task-definition ecs-demo --desired-count 10
- docker-compose create ecs-simple-service
- aws ecs create-service --service-name ecs-simple-service --task-definition ecs-demo --desired-count 10(Correct)
- aws ecs run-task --cluster default --task-definition ecs-demo

Explanation

Correct option:
aws ecs create-service --service-name ecs-simple-service --task-definition ecs-demo --desired-count 10
To create a new service you would use this command which creates a service in your default region called ecs-simple-service. The service uses the ecs-demo task definition and it maintains 10 instantiations of that task.
Incorrect options:
aws ecr create-service --service-name ecs-simple-service --task-definition ecs-demo --desired-count 10 - This command is referencing a different service called Amazon Elastic Container Registry (ECR) which's is a fully-managed Docker container registry
docker-compose create ecs-simple-service - This is a docker command to create containers for a service.
aws ecs run-task --cluster default --task-definition ecs-demo - This is a valid command but used for starting a new task using a specified task definition.
Reference:
https://docs.aws.amazon.com/cli/latest/reference/ecs/create-service.html

Question 9:
A data analytics company ingests a large number of messages and stores them in an RDS database using Lambda. Because of the increased payload size, it is taking more than 15 minutes to process each message.
As a Developer Associate, which of the following options would you recommend to process each message in the MOST scalable way?

- Provision an EC2 instance to poll the messages from an SQS queue
- Use DynamoDB instead of RDS as database
- Contact AWS Support to increase the Lambda timeout to 60 minutes
- Provision EC2 instances in an Auto Scaling group to poll the messages from an SQS queue(Correct)

Explanation

Correct option:
Provision EC2 instances in an Auto Scaling group to poll the messages from an SQS queue
As each message takes more than 15 minutes to process, therefore the development team cannot use Lambda for message processing. To build a scalable solution, the EC2 instances must be provisioning via an Auto Scaling group to handle variations in the message processing workload.

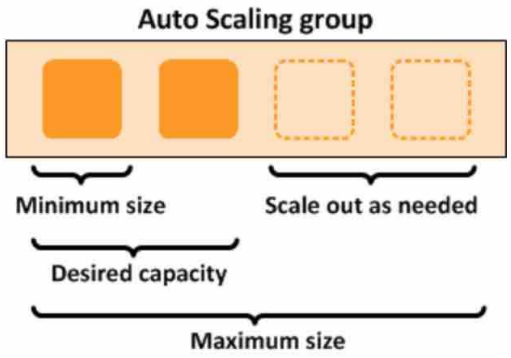
Amazon EC2 Auto Scaling Overview:

What Is Amazon EC2 Auto Scaling?

PDF | Kindle | RSS

Amazon EC2 Auto Scaling helps you ensure that you have the correct number of Amazon EC2 instances available to handle the load for your application. You create collections of EC2 instances, called *Auto Scaling groups*. You can specify the minimum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes below this size. You can specify the maximum number of instances in each Auto Scaling group, and Amazon EC2 Auto Scaling ensures that your group never goes above this size. If you specify the desired capacity, either when you create the group or at any time thereafter, Amazon EC2 Auto Scaling ensures that your group has this many instances. If you specify scaling policies, then Amazon EC2 Auto Scaling can launch or terminate instances as demand on your application increases or decreases.

For example, the following Auto Scaling group has a minimum size of one instance, a desired capacity of two instances, and a maximum size of four instances. The scaling policies that you define adjust the number of instances, within your minimum and maximum number of instances, based on the criteria that you specify.



via - https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html

Incorrect options:

Provision an EC2 instance to poll the messages from an SQS queue - Just using a single EC2 instance may not be sufficient to handle a sudden spike in the number of incoming messages.

Contact AWS Support to increase the Lambda timeout to 60 minutes - AWS Support cannot increase the Lambda timeout upper limit.

Use DynamoDB instead of RDS as database - This option has been added as a distractor, as changing the database would have no impact on the Lambda timeout while processing the message.

Reference:

https://docs.aws.amazon.com/autoscaling/ec2/userguide/what-is-amazon-ec2-auto-scaling.html

Question 10:

Your client wants to deploy a service on EC2 instances, and as EC2 instances are added into an ASG, each EC2 instance should be running 3 different Docker Containers simultaneously.

What Elastic Beanstalk platform should they choose?

- Custom platform
- Third-party platform
- Docker single-container platform
- Docker multi-container platform(Correct)

Explanation

Correct option:
Docker multi-container platform
Docker is a container platform that allows you to define your software stack and store it in an image that can be downloaded from a remote repository. Use the Multicontainer Docker platform if you need to run multiple containers on each instance. The Multicontainer Docker platform does not include a proxy server. Elastic Beanstalk uses Amazon Elastic Container Service (Amazon ECS) to coordinate container deployments to multi-container Docker environments.
Incorrect options:
Docker single-container platform - Docker is a container platform that allows you to define your software stack and store it in an image that can be downloaded from a remote repository. Use the Single Container Docker platform if you only need to run a single Docker container on each instance in your environment. The single container platform includes an Nginx proxy server.
Custom Platform - Elastic Beanstalk supports custom platforms. A custom platform provides more advanced customization than a custom image in several ways. A custom platform lets you develop an entirely new platform from scratch, customizing the operating system, additional software, and scripts that Elastic Beanstalk runs on platform instances. This flexibility enables you to build a platform for an application that uses a language or other infrastructure software, for which Elastic Beanstalk doesn't provide a managed platform. Compare that to custom images, where you modify an Amazon Machine Image (AMI) for use with an existing Elastic Beanstalk platform, and Elastic Beanstalk still provides the platform scripts and controls the platform's software stack. Besides, with custom platforms, you use an automated, scripted way to create and maintain your customization, whereas with custom images you make the changes manually over a running instance.
Third Party Platform - This is a made-up option.
Reference:
https://docs.aws.amazon.com/elasticbeanstalk/latest/platforms/platforms-supported.html#platforms-supported.mcdocker
Question 11:
A company ingests real-time data into its on-premises data center and subsequently a daily data feed is compressed into a single file and uploaded on Amazon S3 for backup. The typical compressed file size is around 2 GB.
Which of the following is the fastest way to upload the daily compressed file into S3?

- FTP the compressed file into an EC2 instance that runs in the same region as the S3 bucket. Then transfer the file from the EC2 instance into the S3 bucket
- Upload the compressed file in a single operation
- Upload the compressed file using multipart upload
- Upload the compressed file using multipart upload with S3 transfer acceleration
- (Correct)

Explanation

Correct option:

Upload the compressed file using multipart upload with S3 transfer acceleration

Amazon S3 Transfer Acceleration enables fast, easy, and secure transfers of files over long distances between your client and an S3 bucket. Transfer Acceleration takes advantage of Amazon CloudFront's globally distributed edge locations. As the data arrives at an edge location, data is routed to Amazon S3 over an optimized network path.

Multipart upload allows you to upload a single object as a set of parts. Each part is a contiguous portion of the object's data. You can upload these object parts independently and in any order. If transmission of any part fails, you can retransmit that part without affecting other parts. After all parts of your object are uploaded, Amazon S3 assembles these parts and creates the object. If you're uploading large objects over a stable high-bandwidth network, use multipart uploading to maximize the use of your available bandwidth by uploading object parts in parallel for multi-threaded performance. If you're uploading over a spotty network, use multipart uploading to increase resiliency to network errors by avoiding upload restarts.

Incorrect options:

Upload the compressed file in a single operation - In general, when your object size reaches 100 MB, you should consider using multipart uploads instead of uploading the object in a single operation. Multipart upload provides improved throughput - you can upload parts in parallel to improve throughput. Therefore, this option is not correct.

Upload the compressed file using multipart upload - Although using multipart upload would certainly speed up the process, combining with S3 transfer acceleration would further improve the transfer speed. Therefore just using multipart upload is not the correct option.

FTP the compressed file into an EC2 instance that runs in the same region as the S3 bucket. Then transfer the file from the EC2 instance into the S3 bucket - This is a roundabout process of getting the file into S3 and added as a distractor. Although it is technically feasible to follow this process, it would involve a lot of scripting and certainly would not be the fastest way to get the file into S3.

References:

https://docs.aws.amazon.com/AmazonS3/latest/dev/transfer-acceleration.html

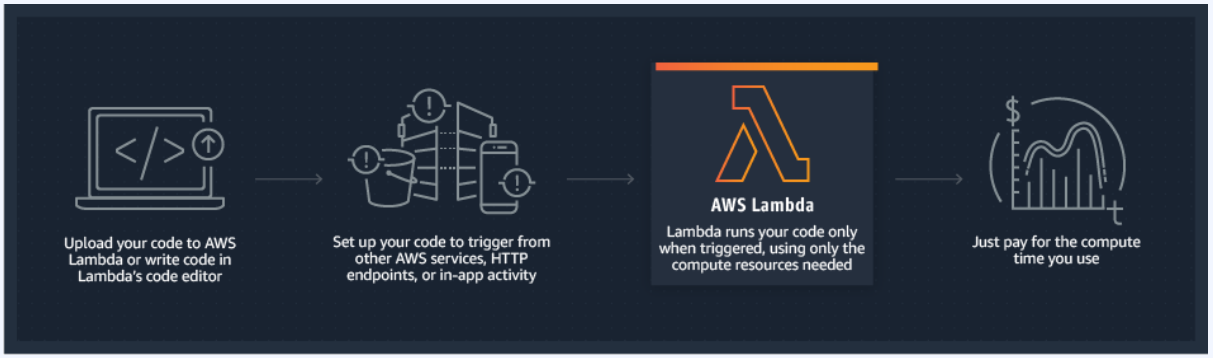
https://docs.aws.amazon.com/AmazonS3/latest/dev/uploadobjusingmpu.html

Question 12:
Your Lambda function must use the Node.js drivers to connect to your RDS PostgreSQL database in your VPC.
How do you bundle your Lambda function to add the dependencies?

- Put the function and the dependencies in one folder and zip them together(Correct)
- Upload the code through the AWS console and upload the dependencies as a zip
- Zip the function as-is with a package.json file so that AWS Lambda can resolve the dependencies for you
- Zip the function and the dependencies separately and upload them in AWS Lambda as two parts

Explanation

Correct option:
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/
Put the function and the dependencies in one folder and zip them together
A deployment package is a ZIP archive that contains your function code and dependencies. You need to create a deployment package if you use the Lambda API to manage functions, or if you need to include libraries and dependencies other than the AWS SDK. You can upload the package directly to Lambda, or you can use an Amazon S3 bucket, and then upload it to Lambda. If the deployment package is larger than 50 MB, you must use Amazon S3. This is the standard way of packaging Lambda functions.
Incorrect options:
Zip the function as-is with a package.json file so that AWS Lambda can resolve the dependencies for you
Upload the code through the AWS console and upload the dependencies as a zip
Zip the function and the dependencies separately and upload them in AWS Lambda as two parts
These three options are incorrect as there's only one way of deploying a Lambda function, which is to provide the zip file with all dependencies that it'll need.
Reference:
https://docs.aws.amazon.com/lambda/latest/dg/nodejs-create-deployment-pkg.html

Question 13:
You are working with a t2.small instance bastion host that has the AWS CLI installed to help manage all the AWS services installed on it. You would like to know the security group and the instance id of the current instance.
Which of the following will help you fetch the needed information?

- Query the user data at http://169.254.169.254/latest/user-data
- Query the metadata at http://169.254.169.254/latest/meta-data(Correct)
- Create an IAM role and attach it to your EC2 instance that helps you perform a 'describe' API call
- Query the user data at http://254.169.254.169/latest/meta-data

Explanation

Correct option:
Query the metadata at http://169.254.169.254/latest/meta-data - Because your instance metadata is available from your running instance, you do not need to use the Amazon EC2 console or the AWS CLI. This can be helpful when you're writing scripts to run from your instance. For example, you can access the local IP address of your instance from instance metadata to manage a connection to an external application. To view all categories of instance metadata from within a running instance, use the following URI - http://169.254.169.254/latest/meta-data/. The IP address 169.254.169.254 is a link-local address and is valid only from the instance. All instance metadata is returned as text (HTTP content type text/plain).
Incorrect options:
Create an IAM role and attach it to your EC2 instance that helps you perform a 'describe' API call - The AWS CLI has a describe-instances API call needs instance ID as an input. So, this will not work for the current use case wherein we do not know the instance ID.
Query the user data at http://169.254.169.254/latest/user-data - This address retrieves the user data that you specified when launching your instance.
Query the user data at http://254.169.254.169/latest/meta-data - The IP address specified is wrong.
References:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instancedata-data-retrieval.html
https://awscli.amazonaws.com/v2/documentation/api/latest/reference/ec2/describe-instances.html

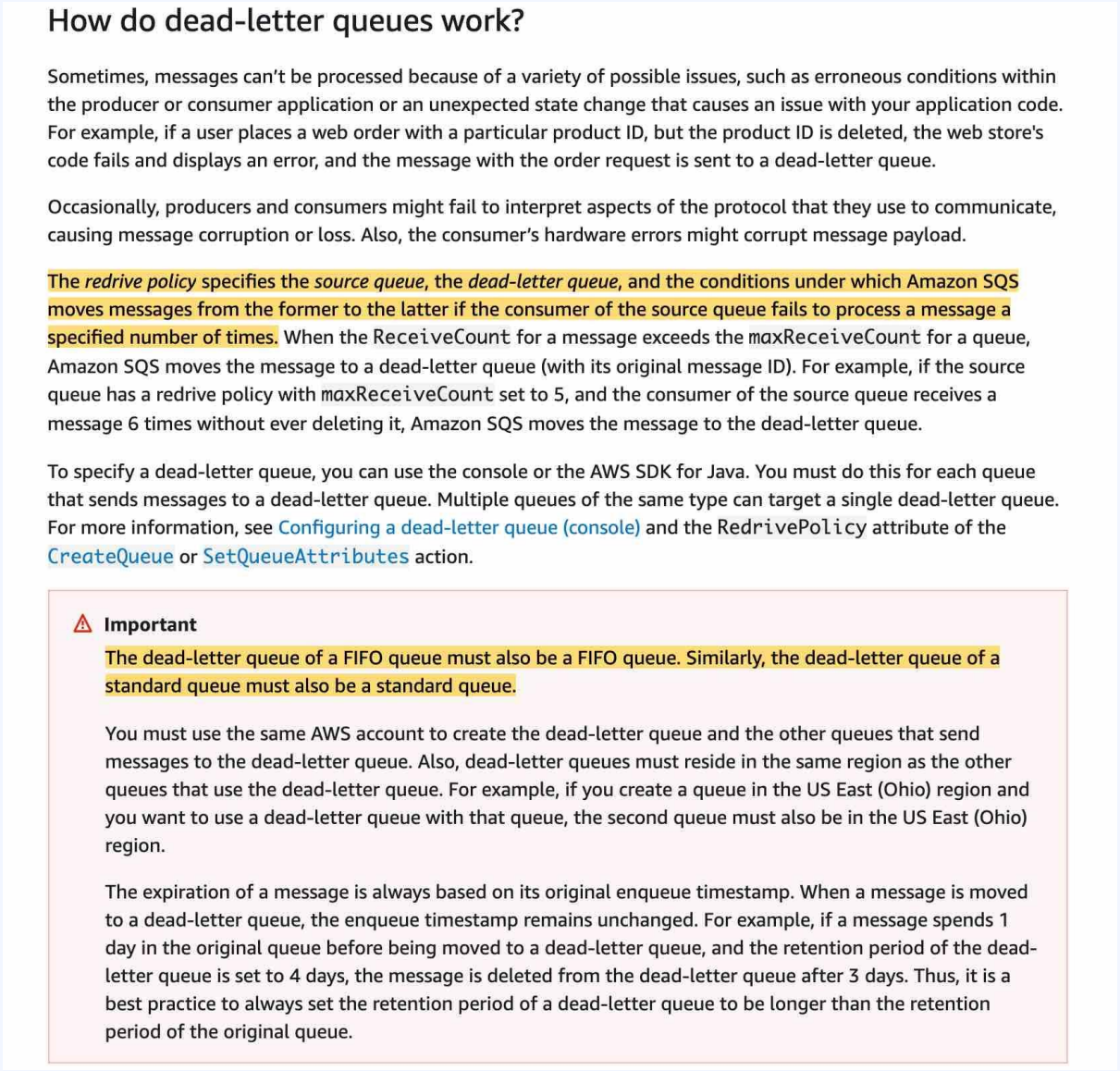
Question 14:
Applications running on EC2 instances process messages from an SQS queue but sometimes they experience errors due to messages not being processed.
To isolate the messages, which option will help with further debugging?

- Implement a Dead Letter Queue(Correct)
- Reduce the VisibilityTimeout
- Increase the VisibilityTimeout
- Use DeleteMessage

Explanation

Correct option:
Implement a Dead Letter Queue
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.

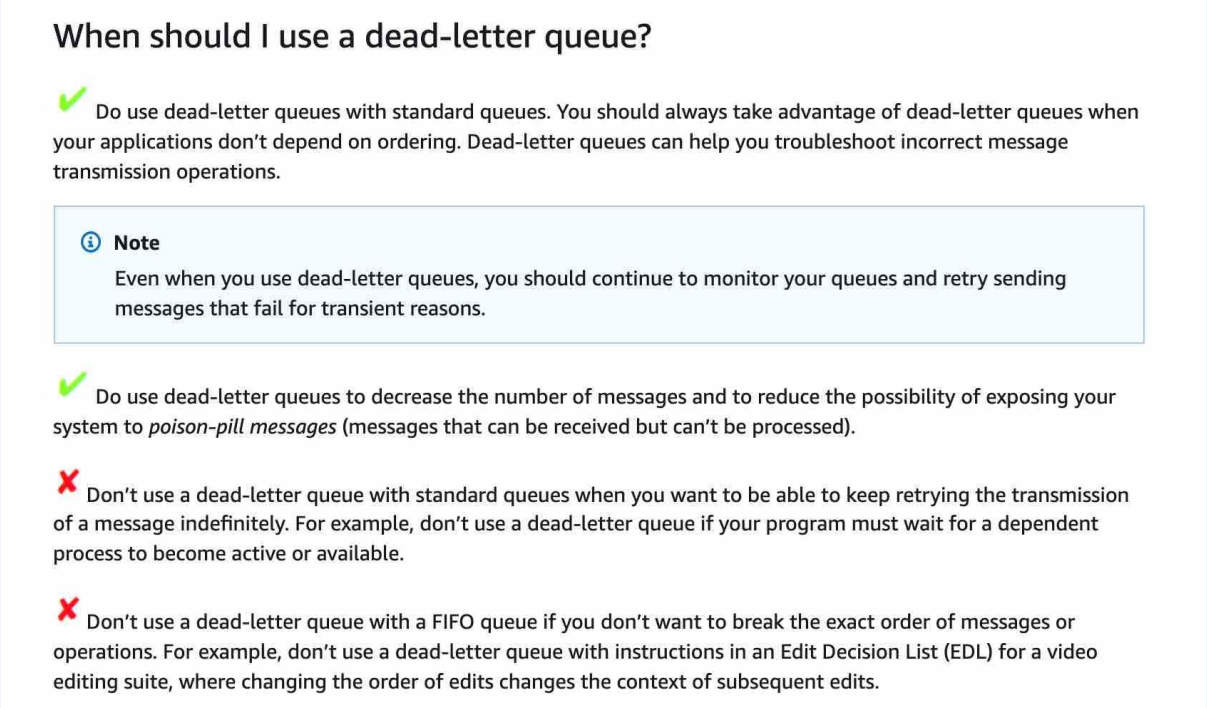
The *redrive policy* specifies the *source queue*, the *dead-letter queue*, and the conditions under which Amazon SQS moves messages from the former to the latter if the consumer of the source queue fails to process a message a specified number of times. When the `ReceiveCount` for a message exceeds 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

Amazon SQS supports dead-letter queues, which other queues (source queues) can target for messages that cannot 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.

Incorrect:

Use DeleteMessage - This API call deletes the message in the queue but does not help you find the issue.

Reduce the VisibilityTimeout - 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. If you reduce the VisibilityTimeout, more consumers will get the failed message

Increase the VisibilityTimeout - It won't help because you don't need more time but rather an isolated place to debug.

Reference:

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

Question 15:
A company has hosted its restaurant review website on an Amazon EC2 instance. The website supports multiple languages and the preferred language is added as a query string parameter to the request. The directory structure and file names for all versions of the website are identical. The website responds with the chosen language's webpage when accessed directly. However, when the same webpage is accessed through the configured CloudFront distribution, it defaults to a single language.
How will you fix this issue?

- Create a new cache policy for the CloudFront distribution and set the cache behavior to Cache based on selected request headers. Use Whitelist Headers as the caching criteria
- Choose the Customize option for the Object Caching setting and reduce the Default TTL value so that CloudFront forwards requests to your origin more frequently
- Create a new cache policy for the CloudFront distribution and set the cache behavior to None to improve caching performance. Update the CloudFront distribution to use the new cache policy
- Create a new cache policy for the CloudFront distribution and set the cache behavior to Query string forwarding and caching. In the Query string whitelist field include the language string. Update the CloudFront distribution to use the new cache policy(Correct)

Explanation

Correct option:
Create a new cache policy for the CloudFront distribution and set the cache behavior to Query string forwarding and caching. In the Query string whitelist field include the language string. Update the CloudFront distribution to use the new cache policy
CloudFront can cache different versions of your content based on the values of query string parameters. Forward all, cache based on whitelist option should be chosen if your origin server returns different versions of your objects based on one or more query string parameters. Then specify the parameters that you want CloudFront to use as a basis for caching in the Query string whitelist field.
Incorrect options:
Create a new cache policy for the CloudFront distribution and set the cache behavior to None to improve caching performance. Update the CloudFront distribution to use the new cache policy - None improves Caching. Choose this option if your origin returns the same version of an object regardless of the values of query string parameters. This increases the likelihood that CloudFront can serve a request from the cache, which improves performance and reduces the load on your origin.
Create a new cache policy for the CloudFront distribution and set the cache behavior to Cache based on selected request headers. Use Whitelist Headers as the caching criteria - Cache based on selected request headers is not a valid option since the use case mentions using query string parameters.
Choose the Customize option for the Object Caching setting and reduce the Default TTL value so that CloudFront forwards requests to your origin more frequently - Default TTL specifies the default amount of time, in seconds, that you want objects to stay in CloudFront caches before CloudFront forwards another request to your origin to determine whether the object has been updated. This option is irrelevant for the current use case since the response returned defaulting to the same language is not a TTL issue.
References:

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/distribution-web-values-specify.html#DownloadDistValuesQueryString
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/distribution-web-values-specify.html#DownloadDistValuesQueryStringWhiteList

Question 16:
A media company wants to migrate a video editing service to Amazon EC2 while following security best practices. The videos are sourced and read from a non-public S3 bucket.
As a Developer Associate, which of the following solutions would you recommend for the given use-case?

- Set up an IAM user with read-only permissions for the S3 bucket. Configure AWS credentials for this user via AWS CLI on the EC2 instance
- Set up an IAM user with read-only permissions for the S3 bucket. Configure the IAM user credentials in the user data of the EC2 instance
- Set up an S3 service role with read-only permissions for the S3 bucket and attach the role to the EC2 instance profile
- Set up an EC2 service role with read-only permissions for the S3 bucket and attach the role to the EC2 instance profile(Correct)

Explanation

Correct option:
Set up an EC2 service role with read-only permissions for the S3 bucket and attach the role to the EC2 instance profile
As an AWS security best practice, you should not create an IAM user and pass the user's credentials to the application or embed the credentials in the application. Instead, create an IAM role that you attach to the EC2 instance to give temporary security credentials to applications running on the instance. When an application uses these credentials in AWS, it can perform all of the operations that are allowed by the policies attached to the role.
So for the given use-case, you should create an IAM role with read-only permissions for the S3 bucket and apply it to the EC2 instance profile.

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

Create an IAM role in the following situations:

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

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

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

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

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

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

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

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

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

Incorrect options:

Set up an IAM user with read-only permissions for the S3 bucket. Configure AWS credentials for this user via AWS CLI on the EC2 instance

Set up an IAM user with read-only permissions for the S3 bucket. Configure the IAM user credentials in the user data of the EC2 instance

As mentioned in the explanation above, it is dangerous to pass an IAM user's credentials to the application or embed the credentials in the application or even configure these credentials in the user data of the EC2 instance. So both these options are incorrect.

Set up an S3 service role with read-only permissions for the S3 bucket and attach the role to the EC2 instance profile - As the application is running on EC2 instances, therefore you need to set up an EC2 service role, not an S3 service role.

Reference:

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

Question 17:
You are running a public DNS service on an EC2 instance where the DNS name is pointing to the IP address of the instance. You wish to upgrade your DNS service but would like to do it without any downtime.
Which of the following options will help you accomplish this?

- Elastic IP(Correct)
- Provide a static private IP
- Create a Load Balancer and an auto scaling group
- Use Route 53

Explanation

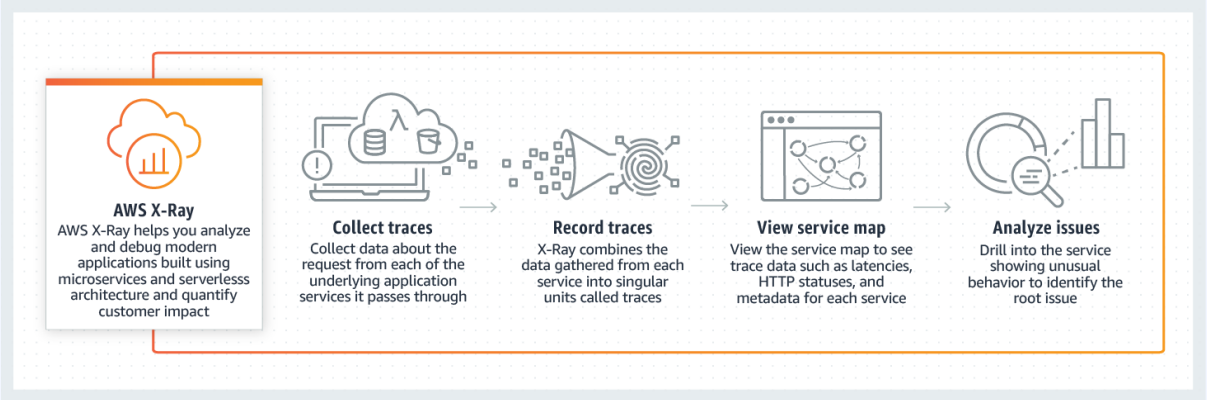
Correct option:
Route 53 is a DNS managed by AWS, but nothing prevents you from running your own DNS (it's just a software) on an EC2 instance. The trick of this question is that it's about EC2, running some software that needs a fixed IP, and not about Route 53 at all.
Elastic IP
DNS services are identified by a public IP, so you need to use Elastic IP.
Incorrect options:
Create a Load Balancer and an auto-scaling group - Load balancers do not provide an IP, instead they provide a DNS name, so this option is ruled out.
Provide a static private IP - If you provide a private IP it will not be accessible from the internet, so this option is incorrect.
Use Route 53 - Route 53 is a DNS service from AWS but the use-case talks about offering a DNS service using an EC2 instance, so this option is incorrect.
Reference:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html#using-instance-addressing-eips-associating-different

Question 18:
Your company likes to operate multiple AWS accounts so that teams have their environments. Services deployed across these accounts interact with one another, and now there's a requirement to implement X-Ray traces across all your applications deployed on EC2 instances and AWS accounts. As such, you would like to have a unified account to view all the traces. What should you in your X-Ray daemon set up to make this work? (Select two)

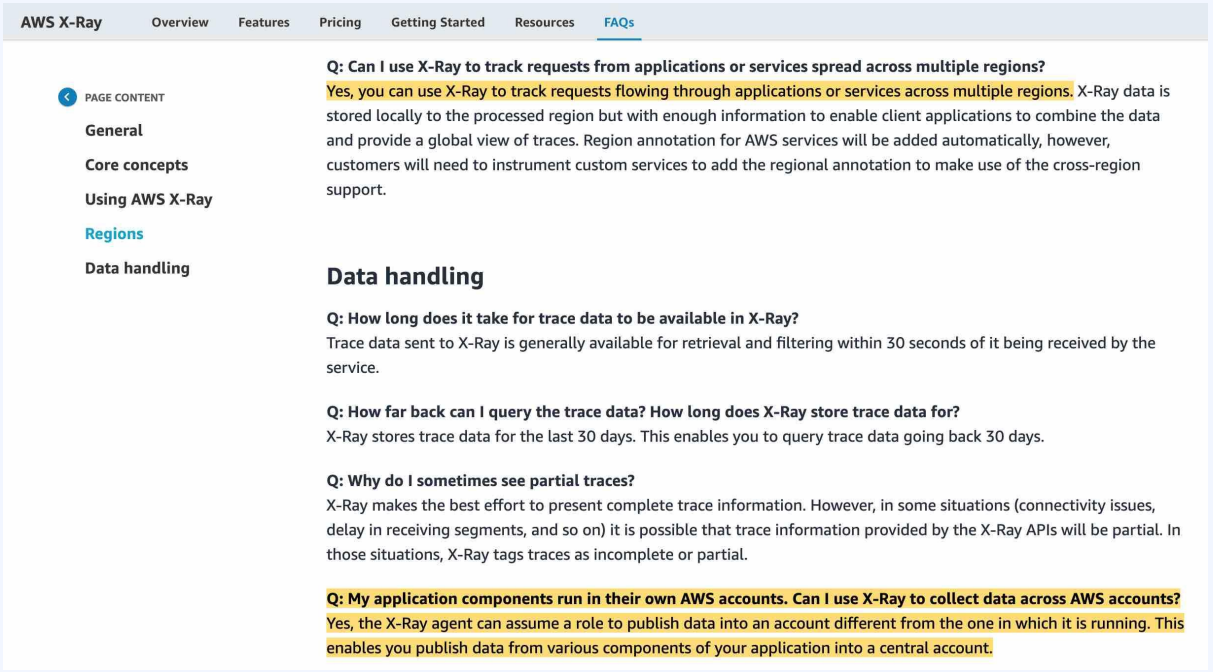
- Configure the X-Ray daemon to use access and secret keys
- Create a role in the target unified account and allow roles in each sub-account to assume the role.(Correct)
- Create a user in the target unified account and generate access and secret keys
- Enable Cross Account collection in the X-Ray console
- Configure the X-Ray daemon to use an IAM instance role(Correct)

Explanation

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



via - https://aws.amazon.com/xray/
Create a role in the target unified account and allow roles in each sub-account to assume the role
Configure the X-Ray daemon to use an IAM instance role
The X-Ray agent can assume a role to publish data into an account different from the one in which it is running. This enables you to publish data from various components of your application into a central account.
X-Ray can also track requests flowing through applications or services across multiple AWS Regions.



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

You can create the necessary configurations for cross-account access via this reference documentation - https://docs.aws.amazon.com/xray/latest/devguide/xray-daemon-configuration.html

Incorrect options:

Create a user in the target unified account and generate access and secret keys

Configure the X-Ray daemon to use access and secret keys

These two options combined together would work but wouldn't be a best-practice security-wise. Therefore these are not correct.

Enable Cross Account collection in the X-Ray console - This is a made-up option and has been added as a distractor.

References:

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

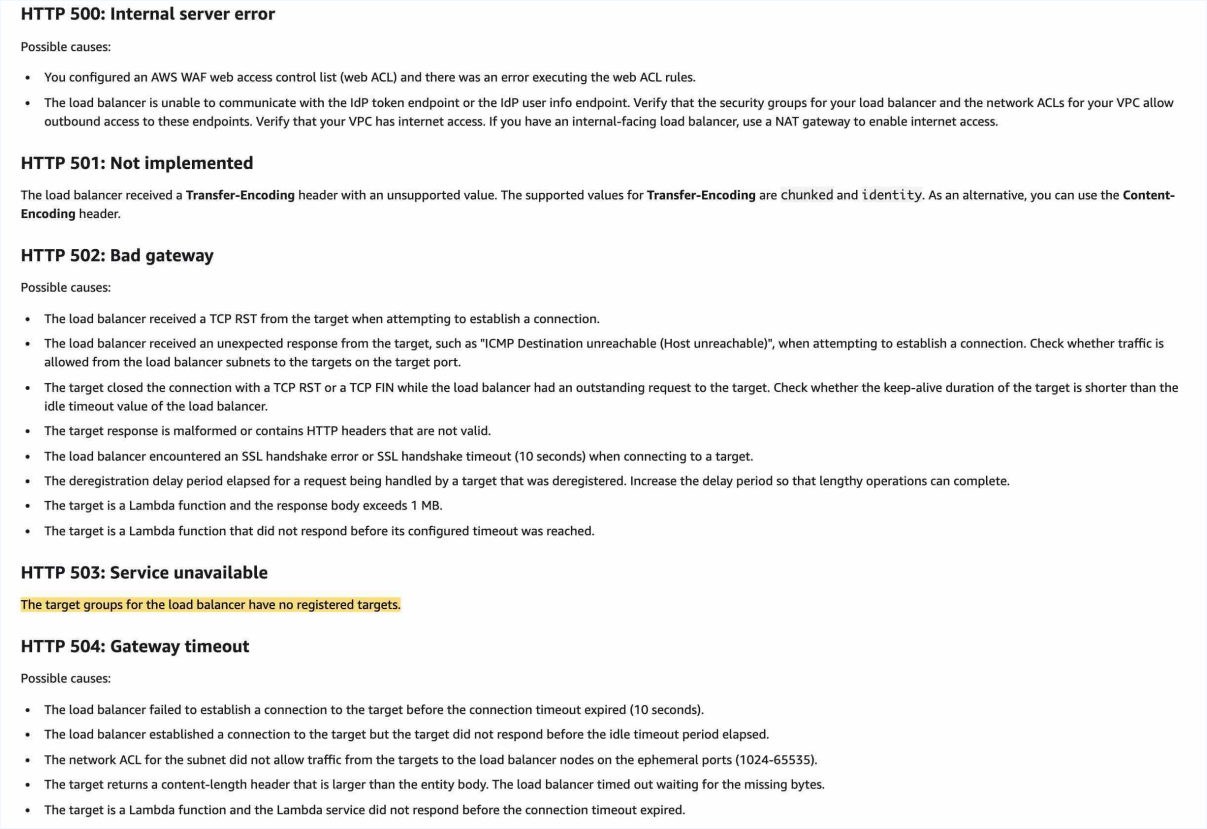
https://docs.aws.amazon.com/xray/latest/devguide/xray-daemon-configuration.html

Question 19:
A developer has created a new Application Load Balancer but has not registered any targets with the target groups.
Which of the following errors would be generated by the Load Balancer?

- HTTP 500: Internal server error
- HTTP 502: Bad gateway
- HTTP 503: Service unavailable(Correct)
- HTTP 504: Gateway timeout

Explanation

Correct option:
HTTP 503: Service unavailable
The Load Balancer generates the HTTP 503: Service unavailable error when the target groups for the load balancer have no registered targets.
Incorrect options:
HTTP 500: Internal server error
HTTP 502: Bad gateway
HTTP 504: Gateway timeout
Here is a summary of the possible causes for these error types:



HTTP 500: Internal server error

Possible causes:

- You configured an AWS WAF web access control list (web ACL) and there was an error executing the web ACL rules.
- The load balancer is unable to communicate with the IdP token endpoint or the IdP user info endpoint. Verify that the security groups for your load balancer and the network ACLs for your VPC allow outbound access to these endpoints. Verify that your VPC has internet access. If you have an internal-facing load balancer, use a NAT gateway to enable internet access.

HTTP 501: Not implemented

The load balancer received a **Transfer-Encoding** header with an unsupported value. The supported values for **Transfer-Encoding** are `chunked` and `identity`. As an alternative, you can use the **Content-Encoding** header.

HTTP 502: Bad gateway

Possible causes:

- The load balancer received a TCP RST from the target when attempting to establish a connection.
- The load balancer received an unexpected response from the target, such as "ICMP Destination unreachable (Host unreachable)", when attempting to establish a connection. Check whether traffic is allowed from the load balancer subnets to the targets on the target port.
- The target closed the connection with a TCP RST or a TCP FIN while the load balancer had an outstanding request to the target. Check whether the keep-alive duration of the target is shorter than the idle timeout value of the load balancer.
- The target response is malformed or contains HTTP headers that are not valid.
- The load balancer encountered an SSL handshake error or SSL handshake timeout (10 seconds) when connecting to a target.
- The deregistration delay period elapsed for a request being handled by a target that was deregistered. Increase the delay period so that lengthy operations can complete.
- The target is a Lambda function and the response body exceeds 1 MB.
- The target is a Lambda function that did not respond before its configured timeout was reached.

HTTP 503: Service unavailable

The target groups for the load balancer have no registered targets.

HTTP 504: Gateway timeout

Possible causes:

- The load balancer failed to establish a connection to the target before the connection timeout expired (10 seconds).
- The load balancer established a connection to the target but the target did not respond before the idle timeout period elapsed.
- The network ACL for the subnet did not allow traffic from the targets to the load balancer nodes on the ephemeral ports (1024-65535).
- The target returns a content-length header that is larger than the entity body. The load balancer timed out waiting for the missing bytes.
- The target is a Lambda function and the Lambda service did not respond before the connection timeout expired.

via - https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-troubleshooting.html
Reference:

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

Question 20:
You would like to have a one-stop dashboard for all the CI/CD needs of one of your projects. You don't need heavy control of the individual configuration of each component in your CI/CD, but need to be able to get a holistic view of your projects.
Which service do you recommend?

- CodePipeline
- CodeBuild
- CodeStar(Correct)
- CodeDeploy

Explanation

Correct option:
CodeStar
AWS CodeStar enables you to quickly develop, build, and deploy applications on AWS. AWS CodeStar provides a unified user interface, enabling you to easily manage your software development activities in one place. With AWS CodeStar, you can set up your entire continuous delivery toolchain in minutes, allowing you to start releasing code faster. AWS CodeStar makes it easy for your whole team to work together securely, allowing you to easily manage access and add owners, contributors, and viewers to your projects. Each AWS CodeStar project comes with a project management dashboard, including an integrated issue tracking capability powered by Atlassian JIRA Software. With the AWS CodeStar project dashboard, you can easily track progress across your entire software development process, from your backlog of work items to teams' recent code deployments.
Incorrect options:
CodeBuild
CodeDeploy
CodePipeline
All these options are individual services encompassed by CodeStar when you deploy a project. They have to be used individually and don't provide a unified "project" view.
Reference:
https://aws.amazon.com/codestar/

Question 21:
You are deploying Lambda functions that operate on your S3 buckets to read files and extract key metadata. The Lambda functions are managed using SAM.
Which Policy should you insert in your serverless model template to give buckets read access?

- SQSPollerPolicy
- S3ReadPolicy
- (Correct)
- LambdaInvokePolicy
- S3CrudPolicy

Explanation

Correct option:
S3ReadPolicy
The AWS Serverless Application Model (AWS SAM) is an open-source framework that you can use to build serverless applications on AWS.
A serverless application is a combination of Lambda functions, event sources, and other resources that work together to perform tasks. Note that a serverless application is more than just a Lambda function—it can include additional resources such as APIs, databases, and event source mappings.
AWS SAM allows you to choose from a list of policy templates to scope the permissions of your Lambda functions to the resources that are used by your application.
AWS SAM applications in the AWS Serverless Application Repository that use policy templates don't require any special customer acknowledgments to deploy the application from the AWS Serverless Application Repository.
S3ReadPolicy => Gives read-only permission to objects in an Amazon S3 bucket.
S3CrudPolicy => Gives create, read, update, and delete permission to objects in an Amazon S3 bucket.
SQSPollerPolicy => Permits to poll an Amazon SQS Queue.
LambdaInvokePolicy => Permits to invoke a Lambda function, alias, or version.
Incorrect options:
SQSPollerPolicy
S3CrudPolicy
LambdaInvokePolicy
These three options contradict the explanation provided earlier. So these are incorrect.
Reference:
https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-policy-templates.html

Question 22:
You would like to paginate the results of an S3 List to show 100 results per page to your users and minimize the number of API calls that you will use.
Which CLI options should you use? (Select two)

- --page-size
- --next-token
- --max-items(Correct)
- --limit
- --starting-token(Correct)

Explanation

Correct options:
--max-items
--starting-token
For commands that can return a large list of items, the AWS Command Line Interface (AWS CLI) has three options to control the number of items included in the output when the AWS CLI calls a service's API to populate the list.
--page-size
--max-items
--starting-token
By default, the AWS CLI uses a page size of 1000 and retrieves all available items. For example, if you run aws s3api list-objects on an Amazon S3 bucket that contains 3,500 objects, the AWS CLI makes four calls to Amazon S3, handling the service-specific pagination logic for you in the background and returning all 3,500 objects in the final output.
Here's an example: aws s3api list-objects --bucket my-bucket --max-items 100 --starting-token eyJNYXJrZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiAxfQ==
Incorrect options:
"--page-size" - You can use the --page-size option to specify that the AWS CLI requests a smaller number of items from each call to the AWS service. The CLI still retrieves the full list but performs a larger number of service API calls in the background and retrieves a smaller number of items with each call.
"--next-token" - This is a made-up option and has been added as a distractor.
"--limit" - This is a made-up option and has been added as a distractor.
Reference:
https://docs.aws.amazon.com/cli/latest/userguide/cli-usage-pagination.html

Question 23:
Which environment variable can be used by AWS X-Ray SDK to ensure that the daemon is correctly discovered on ECS?

- AWS_XRAY_DEBUG_MODE
- AWS_XRAY_CONTEXT_MISSING
- AWS_XRAY_DAEMON_ADDRESS(Correct)
- AWS_XRAY_TRACING_NAME

Explanation

Correct option:
AWS_XRAY_DAEMON_ADDRESS
Set the host and port of the X-Ray daemon listener. By default, the SDK uses 127.0.0.1:2000 for both trace data (UDP) and sampling (TCP). Use this variable if you have configured the daemon to listen on a different port or if it is running on a different host.
Incorrect options:
AWS_XRAY_TRACING_NAME - This sets a service name that the SDK uses for segments.
AWS_XRAY_CONTEXT_MISSING - This should be set to LOG_ERROR to avoid throwing exceptions when your instrumented code attempts to record data when no segment is open.
AWS_XRAY_DEBUG_MODE - This should be set to TRUE to configure the SDK to output logs to the console, instead of configuring a logger.
Reference:
https://docs.aws.amazon.com/xray/latest/devguide/xray-sdk-nodejs-configuration.html

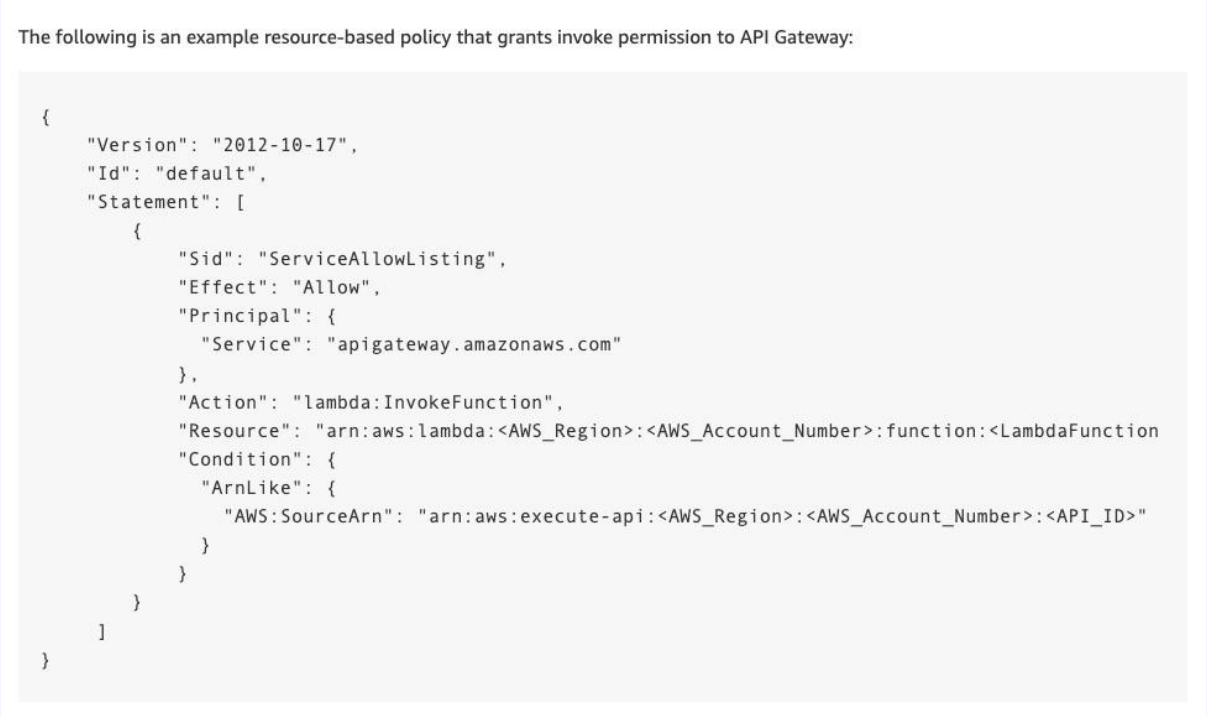
Question 24:
A developer has defined a Lambda integration in Amazon API Gateway using a stage variable. However, when the developer invokes the API method, it consistently returns an "Internal server error" and a 500 status code.
What steps should the developer take to fix the issue?

- API calls can't exceed the maximum allowed API request rate per account and per Region. Implement error retries and exponential backoffs to fix the error
- If you create a stage variable to call a Lambda function through your API, you must add the required permissions. Update your Lambda function's resource-based AWS Identity and Access Management (IAM) policy so that it grants invoke permission to the API Gateway(Correct)
- If you create a stage variable to call a function through your API, you must add the required permissions. Create an IAM role that your Lambda function can assume when invoking the respective AWS resources
- When setting the Lambda function as the value of a stage variable, use the function's ARN and not the function alias for setting up the value

Explanation

Correct option:
If you create a stage variable to call a Lambda function through your API, you must add the required permissions. Update your Lambda function's resource-based AWS Identity and Access Management (IAM) policy so that it grants invoke permission to the API Gateway
If your Lambda function's resource-based policy doesn't include permissions for your API to invoke the function, API Gateway returns an Internal server error message.
If you create a stage variable to call a function through your API, you must add the required permissions by doing one of the following:
Update your Lambda function's resource-based AWS Identity and Access Management (IAM) policy so that it grants invoke permission to API Gateway.
OR
Create an IAM role that API Gateway can assume to invoke your Lambda function.
If you build an API Gateway API with standard Lambda integration using the API Gateway console, the console adds the required permissions automatically.

Example resource-based policy that grants invoke permission to API Gateway:


The following is an example resource-based policy that grants invoke permission to API Gateway:

```
{
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "ServiceAllowListing",
            "Effect": "Allow",
            "Principal": {
              "Service": "apigateway.amazonaws.com"
            },
            "Action": "lambda:InvokeFunction",
            "Resource": "arn:aws:lambda:<AWS_Region>:<AWS_Account_Number>:function:<LambdaFunction
            "Condition": {
              "ArnLike": {
                "AWS:SourceArn": "arn:aws:execute-api:<AWS_Region>:<AWS_Account_Number>:<API_ID>"
              }
            }
        }
     ]
}
```

via - https://aws.amazon.com/premiumsupport/knowledge-center/api-gateway-lambda-stage-variable-500/

Incorrect options:

API calls can't exceed the maximum allowed API request rate per account and per Region. Implement error retries and exponential backoffs to fix the error - When API calls exceed the maximum allowed API request rate per account and per Region, the error received is a "Rate Exceeded" error and not an "Internal server error". Hence, this option is irrelevant to the given use case.

If you create a stage variable to call a function through your API, you must add the required permissions. Create an IAM role that your Lambda function can assume when invoking the respective AWS resources - This option is incorrect. However, creating an IAM role that API Gateway can assume to invoke the Lambda function is another solution to fix the given problem.

When setting the Lambda function as the value of a stage variable, use the function's ARN and not the function alias for setting up the value - This statement is incorrect. When setting a Lambda function as the value of a stage variable, use the function's local name, possibly including its alias or version specification. Do not use the function's ARN. The API Gateway console assumes the stage variable value for a Lambda function as the unqualified function name and expands the given stage variable into an ARN.

References:

https://docs.aws.amazon.com/lambda/latest/dg/services-apigateway.html

https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-set-stage-variables-aws-console.html

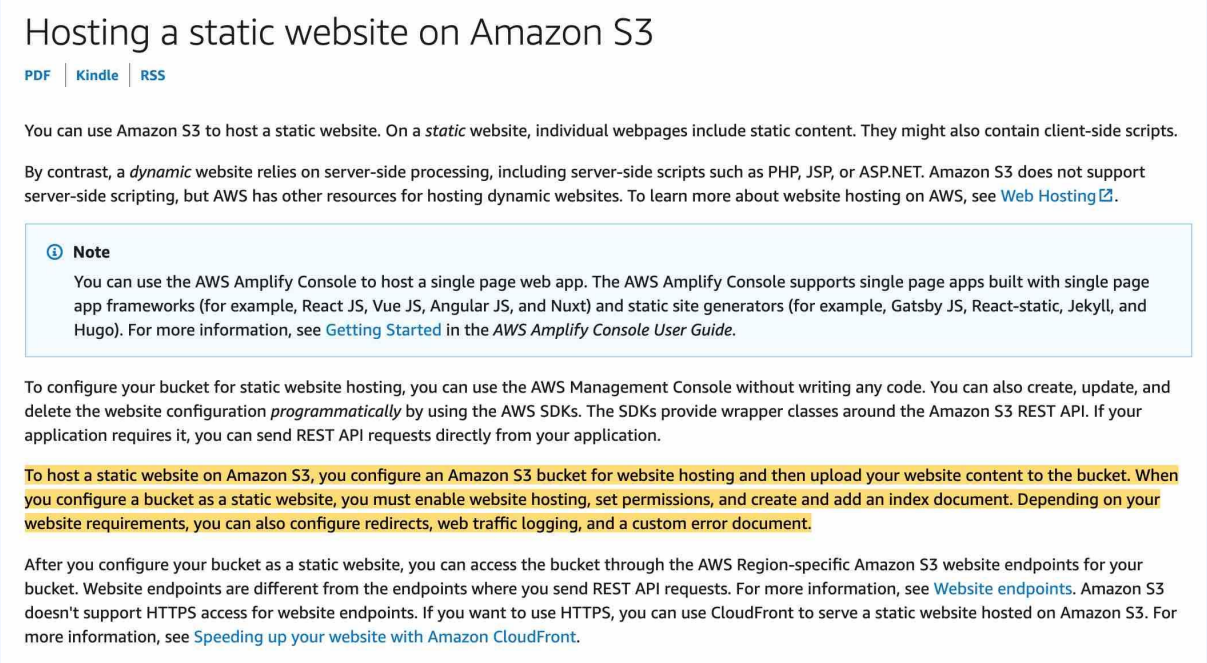
https://aws.amazon.com/premiumsupport/knowledge-center/api-gateway-lambda-stage-variable-500/

Question 25:
Your company is new to cloud computing and would like to host a static HTML5 website on the cloud and be able to access it via domain www.mycompany.com. You have created a bucket in Amazon Simple Storage Service (S3), enabled website hosting, and set the index.html as the default page. Finally, you create an Alias record in Amazon Route 53 that points to the S3 website endpoint of your S3 bucket.
When you test the domain www.mycompany.com you get the following error: 'HTTP response code 403 (Access Denied)'. What can you do to resolve this error?

- Enable CORS
- Enable Encryption
- Create a bucket policy(Correct)
- Create an IAM role

Explanation

Correct answer
Create a bucket policy
Bucket policy is an access policy option available for you to grant permission to your Amazon S3 resources. It uses JSON-based access policy language.
If you want to configure an existing bucket as a static website that has public access, you must edit block public access settings for that bucket. You may also have to edit your account-level block public access settings.



Hosting a static website on Amazon S3

PDF | Kindle | RSS

You can use Amazon S3 to host a static website. On a *static* website, individual webpages include static content. They might also contain client-side scripts.

By contrast, a *dynamic* website relies on server-side processing, including server-side scripts such as PHP, JSP, or ASP.NET. Amazon S3 does not support server-side scripting, but AWS has other resources for hosting dynamic websites. To learn more about website hosting on AWS, see Web Hosting.

Note

You can use the AWS Amplify Console to host a single page web app. The AWS Amplify Console supports single page apps built with single page app frameworks (for example, React JS, Vue JS, Angular JS, and Nuxt) and static site generators (for example, Gatsby JS, React-static, Jekyll, and Hugo). For more information, see Getting Started in the *AWS Amplify Console User Guide*.

To configure your bucket for static website hosting, you can use the AWS Management Console without writing any code. You can also create, update, and delete the website configuration *programmatically* by using the AWS SDKs. The SDKs provide wrapper classes around the Amazon S3 REST API. If your application requires it, you can send REST API requests directly from your application.

To host a static website on Amazon S3, you configure an Amazon S3 bucket for website hosting and then upload your website content to the bucket. When you configure a bucket as a static website, you must enable website hosting, set permissions, and create and add an index document. Depending on your website requirements, you can also configure redirects, web traffic logging, and a custom error document.

After you configure your bucket as a static website, you can access the bucket through the AWS Region-specific Amazon S3 website endpoints for your bucket. Website endpoints are different from the endpoints where you send REST API requests. For more information, see Website endpoints. Amazon S3 doesn't support HTTPS access for website endpoints. If you want to use HTTPS, you can use CloudFront to serve a static website hosted on Amazon S3. For more information, see Speeding up your website with Amazon CloudFront.

via - https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html
Incorrect:
Create an IAM role - This will not help because IAM roles are attached to services and in this case, we have public users.
Enable CORS - CORS defines a way for client web applications that are loaded in one domain to interact with resources in a different domain. Here we are not dealing with cross domains.

Enable Encryption - For the most part, encryption does not have an effect on access denied/forbidden errors. On the website endpoint, if a user requests an object that doesn't exist, Amazon S3 returns HTTP response code 404 (Not Found). If the object exists but you haven't granted read permission on it, the website endpoint returns HTTP response code 403 (Access Denied).
References:
https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html
https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteAccessPermissionsReqd.html

Question 26:
You have created a test environment in Elastic Beanstalk and as part of that environment, you have created an RDS database.
How can you make sure the database can be explored after the environment is destroyed?

- Make a selective delete in Elastic Beanstalk
- Convert the Elastic Beanstalk environment to a worker environment
- Make a snapshot of the database before it gets deleted(Correct)
- Change the Elastic Beanstalk environment variables

Explanation

Correct option:
Make a snapshot of the database before it gets deleted
Use an Elastic Beanstalk blue (environment A)/green (environment B) deployment to decouple an RDS DB instance from environment.
Create a new Elastic Beanstalk environment (environment B) with the necessary information to connect to the RDS DB instance.
Note: An RDS DB instance attached to an Elastic Beanstalk environment is ideal for development and testing environments. However, it's not ideal for production environments because the lifecycle of the database instance is tied to the lifecycle of your application environment. If you terminate the environment, then you lose your data because the RDS DB instance is deleted by the environment. For more information, see Using Elastic Beanstalk with Amazon RDS.
This is the only way to recover the database data before it gets deleted by Elastic Beanstalk.
Please review this excellent document that addresses this use-case :
https://aws.amazon.com/premiumsupport/knowledge-center/decouple-rds-from-beanstalk/
Incorrect options:
Make a selective delete in Elastic Beanstalk - This is not a feature in Elastic Beanstalk.
Change the Elastic Beanstalk environment variables - Environment variables won't help with the provisioned RDS database.
Convert the Elastic Beanstalk environment to a worker environment - You can't convert Elastic Beanstalk environments, you can only change their settings.
Reference:
https://aws.amazon.com/premiumsupport/knowledge-center/decouple-rds-from-beanstalk/

Question 27:
You are creating a web application in which users can follow each other. Some users will be more popular than others and thus their data will be requested very often. Currently, the user data sits in RDS and it has been recommended by your Developer to use ElastiCache as a caching layer to improve the read performance. The whole dataset of users cannot sit in ElastiCache without incurring tremendous costs and therefore you would like to cache only the most often requested users profiles there. As your website is high traffic, it is accepted to have stale data for users for a while, as long as the stale data is less than a minute old.
What caching strategy do you recommend implementing?

- Use a Lazy Loading strategy without TTL
- Use a Lazy Loading strategy with TTL(Correct)
- Use a Write Through strategy without TTL
- Use a Write Through strategy with TTL

Explanation

Correct option
Use a Lazy Loading strategy with TTL
Lazy loading is a caching strategy that loads data into the cache only when necessary. Whenever your application requests data, it first requests the ElastiCache cache. If the data exists in the cache and is current, ElastiCache returns the data to your application. If the data doesn't exist in the cache or has expired, your application requests the data from your data store. Your datastore then returns the data to your application.
In this case, data that is actively requested by users will be cached in ElastiCache, and thanks to the TTL, we can expire that data after a minute to limit the data staleness.

Lazy Loading

As the name implies, *lazy loading* is a caching strategy that loads data into the cache only when necessary. It works as described following.

Amazon ElastiCache is an in-memory key-value store that sits between your application and the data store (database) that it accesses. Whenever your application requests data, it first makes the request to the ElastiCache cache. If the data exists in the cache and is current, ElastiCache returns the data to your application. If the data doesn't exist in the cache or has expired, your application requests the data from your data store. Your data store then returns the data to your application. Your application next writes the data received from the store to the cache. This way, it can be more quickly retrieved the next time it's requested.

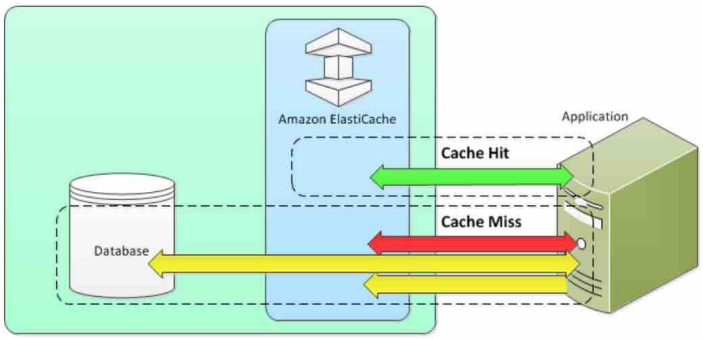
A *cache hit* occurs when data is in the cache and isn't expired:

1. Your application requests data from the cache.
2. The cache returns the data to the application.

A *cache miss* occurs when data isn't in the cache or is expired:

1. Your application requests data from the cache.
2. The cache doesn't have the requested data, so returns a `null`.
3. Your application requests and receives the data from the database.
4. Your application updates the cache with the new data.

The following diagram illustrates both these processes.



via - https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/Strategies.html#Strategies.LazyLoading

Incorrect option:

Use a Lazy Loading strategy without TTL - This fits the read requirements, but won't help expiring stale data, so we need TTL.

Use a Write Through strategy with TTL

Use a Write Through strategy without TTL

The problem with these two options for the write-through strategy is that we would fill up the cache with unnecessary data and as mentioned in the question we don't have enough space in the cache to fit all the dataset. Therefore we can't use a write-through strategy.

Reference:

https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/Strategies.html#Strategies.LazyLoading

Question 28:
The development team at an e-commerce company is preparing for the upcoming Thanksgiving sale. The product manager wants the development team to implement appropriate caching strategy on Amazon ElastiCache to withstand traffic spikes on the website during the sale. A key requirement is to facilitate consistent updates to the product prices and product description, so that the cache never goes out of sync with the backend.
As a Developer Associate, which of the following solutions would you recommend for the given use-case?

- Use a caching strategy to write to the cache directly and sync the backend at a later time
- Use a caching strategy to update the cache and the backend at the same time
- Use a caching strategy to write to the backend first and then invalidate the cache(Correct)
- Use a caching strategy to write to the backend first and wait for the cache to expire via TTL

Explanation

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

1. Lazy Loading
2. Write-Through

Lazy Loading

As the name implies, *lazy loading* is a caching strategy that loads data into the cache only when necessary. It works as described following.

Amazon ElastiCache is an in-memory key-value store that sits between your application and the data store (database) that it accesses. Whenever your application requests data, it first makes the request to the ElastiCache cache. If the data exists in the cache and is current, ElastiCache returns the data to your application. If the data doesn't exist in the cache or has expired, your application requests the data from your data store. Your data store then returns the data to your application. Your application next writes the data received from the store to the cache. This way, it can be more quickly retrieved the next time it's requested.

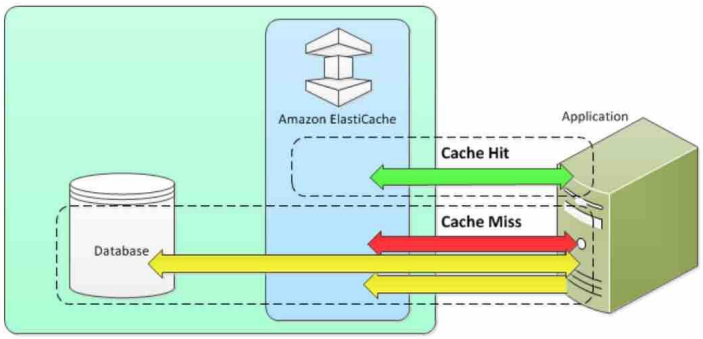
A *cache hit* occurs when data is in the cache and isn't expired:

1. Your application requests data from the cache.
2. The cache returns the data to the application.

A *cache miss* occurs when data isn't in the cache or is expired:

1. Your application requests data from the cache.
2. The cache doesn't have the requested data, so returns a `null`.
3. Your application requests and receives the data from the database.
4. Your application updates the cache with the new data.

The following diagram illustrates both these processes.



via - https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/Strategies.html

Write-Through

The write-through strategy adds data or updates data in the cache whenever data is written to the database.

Advantages and Disadvantages of Write-Through

The advantages of write-through are as follows:

- Data in the cache is never stale.

  Because the data in the cache is updated every time it's written to the database, the data in the cache is always current.

- Write penalty vs. read penalty.

  Every write involves two trips:

  1. A write to the cache
  2. A write to the database

  Which adds latency to the process. That said, end users are generally more tolerant of latency when updating data than when retrieving data. There is an inherent sense that updates are more work and thus take longer.

The disadvantages of write-through are as follows:

- Missing data.

  If you spin up a new node, whether due to a node failure or scaling out, there is missing data. This data continues to be missing until it's added or updated on the database. You can minimize this by implementing lazy loading with write-through.

- Cache churn.

  Most data is never read, which is a waste of resources. By adding a time to live (TTL) value, you can minimize wasted space.

via - https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/Strategies.html

Use a caching strategy to write to the backend first and then invalidate the cache

This option is similar to the write-through strategy wherein the application writes to the backend first and then invalidate the cache. As the cache gets invalidated, the caching engine would then fetch the latest value from the backend, thereby making sure that the product prices and product description stay consistent with the backend.

Incorrect options:

Use a caching strategy to update the cache and the backend at the same time - The cache and the backend cannot be updated at the same time via a single atomic operation as these are two separate systems. Therefore this option is incorrect.

Use a caching strategy to write to the backend first and wait for the cache to expire via TTL - This strategy could work if the TTL is really short. However, for the duration of this TTL, the cache would be out of sync with the backend, hence this option is not correct for the given use-case.

Use a caching strategy to write to the cache directly and sync the backend at a later time - This option is given as a distractor as this strategy is not viable to address the given use-case. The product prices and description on the cache must always stay consistent with the backend. You cannot sync the backend at a later time.

Reference:

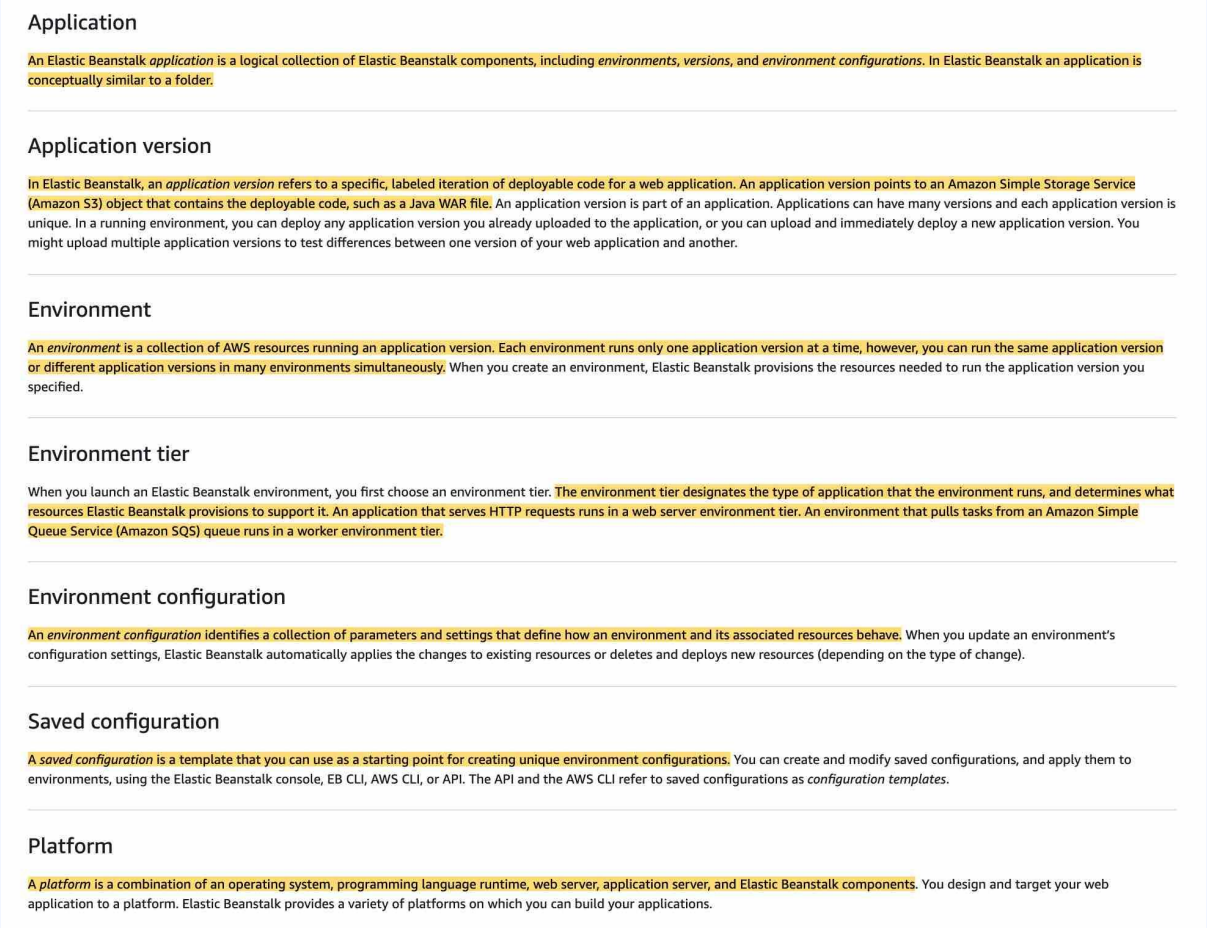
https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/Strategies.html

Question 29:
As part of your video processing application, you are looking to perform a set of repetitive and scheduled tasks asynchronously. Your application is deployed on Elastic Beanstalk.
Which Elastic Beanstalk environment should you set up for performing the repetitive tasks?

- Setup a Worker environment and a .ebextensions file
- Setup a Web Server environment and a cron.yaml file
- Setup a Worker environment and a cron.yaml file(Correct)
- Setup a Web Server environment and a .ebextensions file

Explanation

Correct option:
With Elastic Beanstalk, you can quickly deploy and manage applications in the AWS Cloud without having to learn about the infrastructure that runs those applications. Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring.
Elastic BeanStalk Key Concepts:



Application

An Elastic Beanstalk *application* is a logical collection of Elastic Beanstalk components, including *environments*, *versions*, and *environment configurations*. In Elastic Beanstalk an application is conceptually similar to a folder.

Application version

In Elastic Beanstalk, an *application version* refers to a specific, labeled iteration of deployable code for a web application. An application version points to an Amazon Simple Storage Service (Amazon S3) object that contains the deployable code, such as a Java WAR file. An application version is part of an application. Applications can have many versions and each application version is unique. In a running environment, you can deploy any application version you already uploaded to the application, or you can upload and immediately deploy a new application version. You might upload multiple application versions to test differences between one version of your web application and another.

Environment

An *environment* is a collection of AWS resources running an application version. Each environment runs only one application version at a time, however, you can run the same application version or different application versions in many environments simultaneously. When you create an environment, Elastic Beanstalk provisions the resources needed to run the application version you specified.

Environment tier

When you launch an Elastic Beanstalk environment, you first choose an environment tier. The environment tier designates the type of application that the environment runs, and determines what resources Elastic Beanstalk provisions to support it. An application that serves HTTP requests runs in a web server environment tier. An environment that pulls tasks from an Amazon Simple Queue Service (Amazon SQS) queue runs in a worker environment tier.

Environment configuration

An *environment configuration* identifies a collection of parameters and settings that define how an environment and its associated resources behave. When you update an environment's configuration settings, Elastic Beanstalk automatically applies the changes to existing resources or deletes and deploys new resources (depending on the type of change).

Saved configuration

A *saved configuration* is a template that you can use as a starting point for creating unique environment configurations. You can create and modify saved configurations, and apply them to environments, using the Elastic Beanstalk console, EB CLI, AWS CLI, or API. The API and the AWS CLI refer to saved configurations as *configuration templates*.

Platform

A *platform* is a combination of an operating system, programming language runtime, web server, application server, and Elastic Beanstalk components. You design and target your web application to a platform. Elastic Beanstalk provides a variety of platforms on which you can build your applications.

via - https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts.html
Setup a Worker environment and a cron.yaml file
An environment is a collection of AWS resources running an application version. An environment that pulls tasks from an Amazon Simple Queue Service (Amazon SQS) queue runs in a worker environment tier.

If your AWS Elastic Beanstalk application performs operations or workflows that take a long time to complete, you can offload those tasks to a dedicated worker environment. Decoupling your web application front end from a process that performs blocking operations is a common way to ensure that your application stays responsive under load.
For a worker environment, you need a cron.yaml file to define the cron jobs and do repetitive tasks.

Elastic Beanstalk worker environments

PDF | Kindle

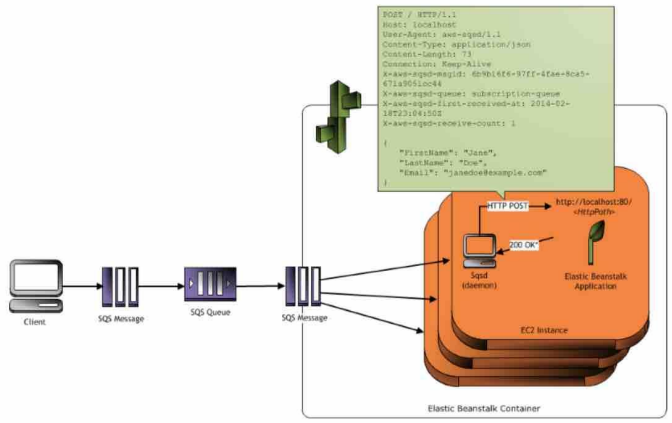
If your AWS Elastic Beanstalk application performs operations or workflows that take a long time to complete, you can offload those tasks to a dedicated *worker environment*. Decoupling your web application front end from a process that performs blocking operations is a common way to ensure that your application stays responsive under load.

A long-running task is anything that substantially increases the time it takes to complete a request, such as processing images or videos, sending email, or generating a ZIP archive. These operations can take only a second or two to complete, but a delay of a few seconds is a lot for a web request that would otherwise complete in less than 500 ms.

One option is to spawn a worker process locally, return success, and process the task asynchronously. This works if your instance can keep up with all of the tasks sent to it. Under high load, however, an instance can become overwhelmed with background tasks and become unresponsive to higher priority requests. If individual users can generate multiple tasks, the increase in load might not correspond to an increase in users, making it hard to scale out your web server tier effectively.

To avoid running long-running tasks locally, you can use the AWS SDK for your programming language to send them to an Amazon Simple Queue Service (Amazon SQS) queue, and run the process that performs them on a separate set of instances. You then design these worker instances to take items from the queue only when they have capacity to run them, preventing them from becoming overwhelmed.

Elastic Beanstalk worker environments simplify this process by managing the Amazon SQS queue and running a daemon process on each instance that reads from the queue for you. When the daemon pulls an item from the queue, it sends an HTTP POST request locally to `http://localhost/` on port 80 with the contents of the queue message in the body. All that your application needs to do is perform the long-running task in response to the POST. You can configure the daemon to post to a different path, use a MIME type other than application/JSON, connect to an existing queue, or customize connections (maximum concurrent requests), timeouts, and retries.



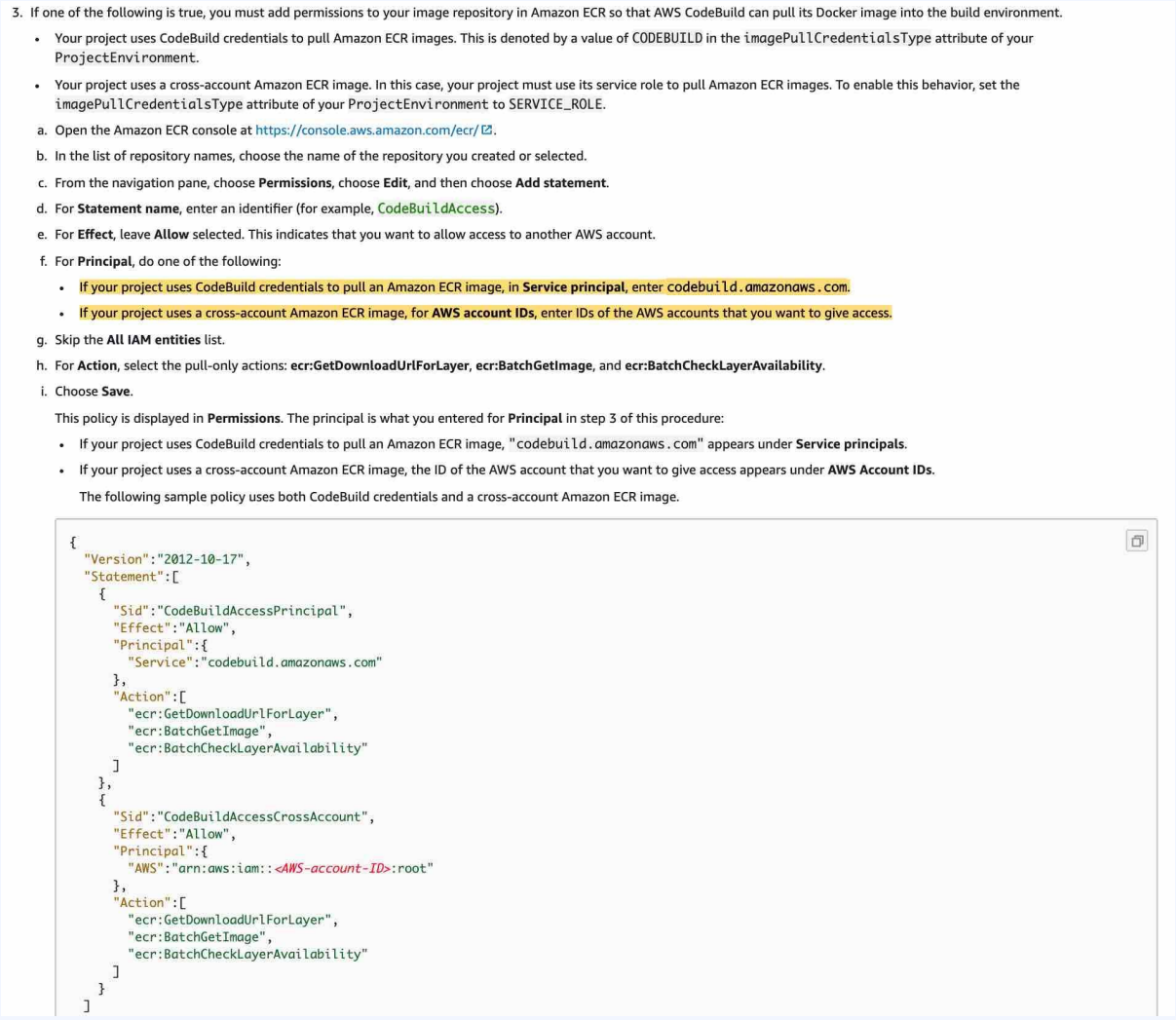
via -
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features-managing-env-tiers.html
Incorrect options:
Setup a Web Server environment and a cron.yaml file
Setup a Worker environment and a .ebextensions file
Setup a Web Server environment and a .ebextensions file
.ebextensions/ won't work to define cron jobs, and Web Server environments cannot be set up to perform repetitive and scheduled tasks. So these three options are incorrect.
References:
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/concepts.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/using-features-managing-env-tiers.html

Question 30:
Your team lead has finished creating a CodeBuild project in the management console and a build spec has been defined for the project. After the build is run, CodeBuild fails to pull a Docker image into the build environment.
What is the most likely cause?

- CodeBuild cannot work with custom Docker images
- The Docker image is missing some tags
- The Docker image is too big
- Missing IAM permissions for the CodeBuild Service(Correct)

Explanation

Correct option:
Missing IAM permissions for the CodeBuild Service
By default, IAM users don't have permission to create or modify Amazon Elastic Container Registry (Amazon ECR) resources or perform tasks using the Amazon ECR API. A user who uses the AWS CodeBuild console must have a minimum set of permissions that allows the user to describe other AWS resources for the AWS account.



3. If one of the following is true, you must add permissions to your image repository in Amazon ECR so that AWS CodeBuild can pull its Docker image into the build environment.
   - Your project uses CodeBuild credentials to pull Amazon ECR images. This is denoted by a value of `CODEBUILD` in the `imagePullCredentialsType` attribute of your `ProjectEnvironment`.
   - Your project uses a cross-account Amazon ECR image. In this case, your project must use its service role to pull Amazon ECR images. To enable this behavior, set the `imagePullCredentialsType` attribute of your `ProjectEnvironment` to `SERVICE_ROLE`.

   a. Open the Amazon ECR console at https://console.aws.amazon.com/ecr/.
   b. In the list of repository names, choose the name of the repository you created or selected.
   c. From the navigation pane, choose **Permissions**, choose **Edit**, and then choose **Add statement**.
   d. For **Statement name**, enter an identifier (for example, `CodeBuildAccess`).
   e. For **Effect**, leave **Allow** selected. This indicates that you want to allow access to another AWS account.
   f. For **Principal**, do one of the following:
      - If your project uses CodeBuild credentials to pull an Amazon ECR image, in **Service principal**, enter `codebuild.amazonaws.com`.
      - If your project uses a cross-account Amazon ECR image, for **AWS account IDs**, enter IDs of the AWS accounts that you want to give access.
   g. Skip the **All IAM entities** list.
   h. For **Action**, select the pull-only actions: **ecr:GetDownloadUrlForLayer**, **ecr:BatchGetImage**, and **ecr:BatchCheckLayerAvailability**.
   i. Choose **Save**.

   This policy is displayed in **Permissions**. The principal is what you entered for **Principal** in step 3 of this procedure:
   - If your project uses CodeBuild credentials to pull an Amazon ECR image, "`codebuild.amazonaws.com`" appears under **Service principals**.
   - If your project uses a cross-account Amazon ECR image, the ID of the AWS account that you want to give access appears under **AWS Account IDs**.

   The following sample policy uses both CodeBuild credentials and a cross-account Amazon ECR image.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"CodeBuildAccessPrincipal",
      "Effect":"Allow",
      "Principal":{
        "Service":"codebuild.amazonaws.com"
      },
      "Action":[
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ]
    },
    {
      "Sid":"CodeBuildAccessCrossAccount",
      "Effect":"Allow",
      "Principal":{
        "AWS":"arn:aws:iam::<AWS-account-ID>:root"
      },
      "Action":[
        "ecr:GetDownloadUrlForLayer",
        "ecr:BatchGetImage",
        "ecr:BatchCheckLayerAvailability"
      ]
    }
  ]
```

via - https://docs.aws.amazon.com/codebuild/latest/userguide/sample-ecr.html
Incorrect options:

The Docker image is missing some tags - Tags are optional for naming purposes
CodeBuild cannot work with custom Docker images - Custom docker images are supported, so this option is incorrect.
The Docker image is too big - It is good to properly design the image but in this case, it does not affect the CodeBuild. You can also look at multi-stage builds, which are a new feature requiring Docker 17.05 or higher on the daemon and client. Multistage builds are useful to anyone who has struggled to optimize Dockerfiles while keeping them easy to read and maintain.
Reference:
https://docs.aws.amazon.com/codebuild/latest/userguide/sample-ecr.html

Question 31:
You are using AWS SQS FIFO queues to get the ordering of messages on a per user_id basis.
As a developer, which message parameter should you set the value of user_id to guarantee the ordering?

- MessageOrderId
- MessageHash
- MessageGroupId(Correct)
- MessageDeduplicationId

Explanation

Correct option:
AWS FIFO queues are designed to enhance messaging between applications when the order of operations and events has to be enforced.

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, for example:

- Ensure that user-entered commands are executed in the right order.
- Display the correct product price by sending price modifications in the right order.
- Prevent a student from enrolling in a course before registering for an account.

FIFO queues also provide exactly-once processing but have a limited number of transactions per second (TPS):

- If you use batching, FIFO queues support up to 3,000 transactions per second, per API method (`SendMessageBatch`, `ReceiveMessage`, or `DeleteMessageBatch`). The 3000 transactions represent 300 API calls, each with a batch of 10 messages. To request a quota increase, submit a support request.
- Without batching, FIFO queues support up to 300 API calls per second, per API method (`SendMessage`, `ReceiveMessage`, or `DeleteMessage`).

Note

- Amazon SNS isn't currently compatible with FIFO queues.
- The name of a FIFO queue must end with the `.fifo` suffix. The suffix counts towards the 80-character queue name quota. To determine whether a queue is FIFO, you can check whether the queue name ends with the suffix.

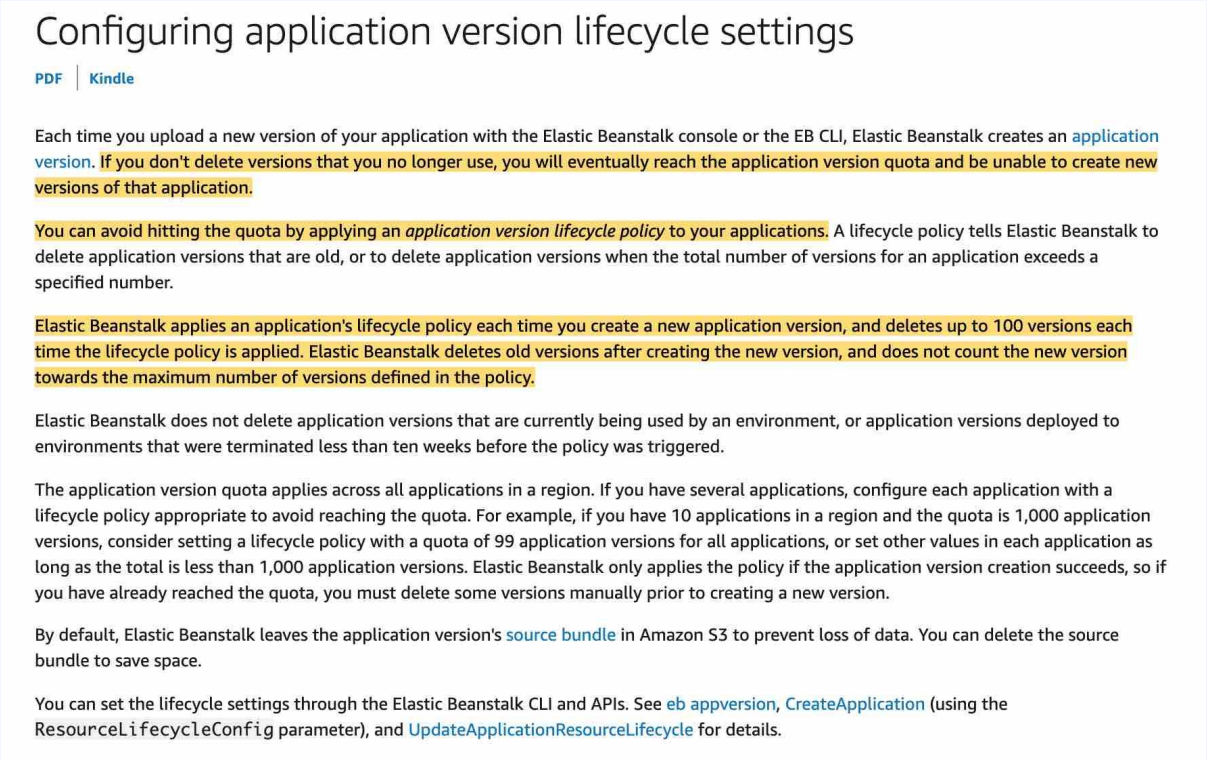
via - https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues.html
MessageGroupId
The message group ID is the tag that specifies that a message belongs to a specific message group. Messages that belong to the same message group are always processed one by one, in a strict order

relative to the message group (however, messages that belong to different message groups might be processed out of order).
Incorrect options:
MessageDeduplicationId - The message deduplication ID is the token used for the deduplication of sent messages. If a message with a particular message deduplication ID is sent successfully, any messages sent with the same message deduplication ID are accepted successfully but aren't delivered during the 5-minute deduplication interval.
MessageOrderId - This is a made-up option and has been added as a distractor.
MessageHash - This is a made-up option and has been added as a distractor.
References:
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues.html
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/using-messagegroupid-property.html
Question 32:
Your organization has set up a full CI/CD pipeline leveraging CodePipeline and the deployment is done on Elastic Beanstalk. This pipeline has worked for over a year now but you are approaching the limits of Elastic Beanstalk in terms of how many versions can be stored in the service.
How can you remove older versions that are not used by Elastic Beanstalk so that new versions can be created for your applications?

- Define a Lambda function
- Use Worker Environments
- Use a Lifecycle Policy(Correct)
- Setup an .ebextensions file

Explanation

Correct option:
Use a Lifecycle Policy
Each time you upload a new version of your application with the Elastic Beanstalk console or the EB CLI, Elastic Beanstalk creates an application version. If you don't delete versions that you no longer use, you will eventually reach the application version limit and be unable to create new versions of that application.
You can avoid hitting the limit by applying an application version lifecycle policy to your applications. A lifecycle policy tells Elastic Beanstalk to delete old application versions or to delete application versions when the total number of versions for an application exceeds a specified number.
Elastic Beanstalk applies an application's lifecycle policy each time you create a new application version and deletes up to 100 versions each time the lifecycle policy is applied. Elastic Beanstalk deletes old versions after creating the new version and does not count the new version towards the maximum number of versions defined in the policy.



Configuring application version lifecycle settings

PDF | Kindle

Each time you upload a new version of your application with the Elastic Beanstalk console or the EB CLI, Elastic Beanstalk creates an application version. If you don't delete versions that you no longer use, you will eventually reach the application version quota and be unable to create new versions of that application.

You can avoid hitting the quota by applying an *application version lifecycle policy* to your applications. A lifecycle policy tells Elastic Beanstalk to delete application versions that are old, or to delete application versions when the total number of versions for an application exceeds a specified number.

Elastic Beanstalk applies an application's lifecycle policy each time you create a new application version, and deletes up to 100 versions each time the lifecycle policy is applied. Elastic Beanstalk deletes old versions after creating the new version, and does not count the new version towards the maximum number of versions defined in the policy.

Elastic Beanstalk does not delete application versions that are currently being used by an environment, or application versions deployed to environments that were terminated less than ten weeks before the policy was triggered.

The application version quota applies across all applications in a region. If you have several applications, configure each application with a lifecycle policy appropriate to avoid reaching the quota. For example, if you have 10 applications in a region and the quota is 1,000 application versions, consider setting a lifecycle policy with a quota of 99 application versions for all applications, or set other values in each application as long as the total is less than 1,000 application versions. Elastic Beanstalk only applies the policy if the application version creation succeeds, so if you have already reached the quota, you must delete some versions manually prior to creating a new version.

By default, Elastic Beanstalk leaves the application version's source bundle in Amazon S3 to prevent loss of data. You can delete the source bundle to save space.

You can set the lifecycle settings through the Elastic Beanstalk CLI and APIs. See eb appversion, CreateApplication (using the `ResourceLifecycleConfig` parameter), and UpdateApplicationResourceLifecycle for details.

via - https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/applications-lifecycle.html

Incorrect options: Setup an .ebextensions files - You can add AWS Elastic Beanstalk configuration files (.ebextensions) to your web application's source code to configure your environment and customize the AWS resources that it contains. This does not help with managing versions.

Define a Lambda function - This could work but would require a lot of manual scripting, to achieve the same desired effect as the Lifecycle Policy EB feature.

Use Worker Environments - This won't help. If your application performs operations or workflows that take a long time to complete, you can offload those tasks to a dedicated worker environment. Decoupling your web application front end from a process that performs blocking operations is a common way to ensure that your application stays responsive under load.

Reference:

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/applications-lifecycle.html

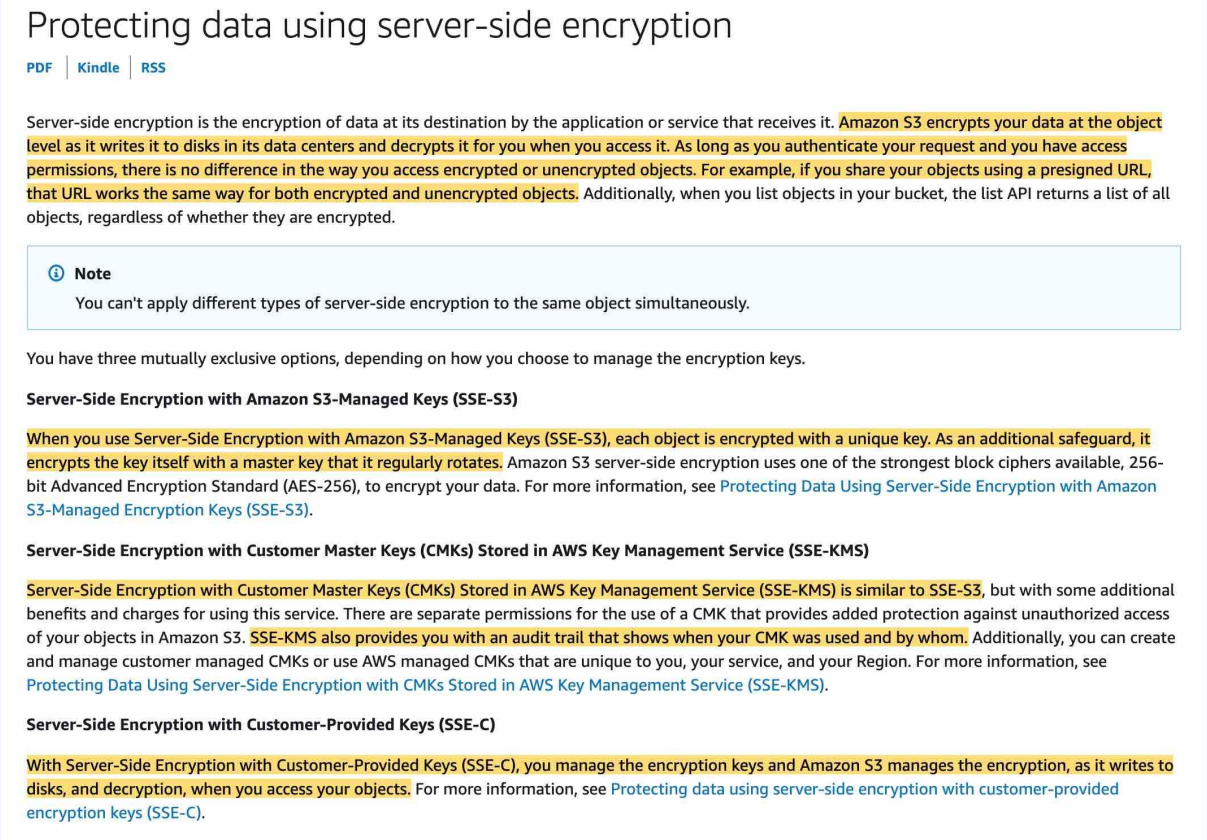
Question 33:
A security company is requiring all developers to perform server-side encryption with customer-provided encryption keys when performing operations in AWS S3. Developers should write software with C# using the AWS SDK and implement the requirement in the PUT, GET, Head, and Copy operations.
Which of the following encryption methods meets this requirement?

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

Explanation

Correct option:
SSE-C
You have the following options for protecting data at rest in Amazon S3:
Server-Side Encryption – Request Amazon S3 to encrypt your object before saving it on disks in its data centers and then decrypt it when you download the objects.
Client-Side Encryption – Encrypt data client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, the encryption keys, and related tools.
For the given use-case, the company wants to manage the encryption keys via its custom application and let S3 manage the encryption, therefore you must use Server-Side Encryption with Customer-Provided Keys (SSE-C).
Using server-side encryption with customer-provided encryption keys (SSE-C) allows you to set your encryption keys. With the encryption key you provide as part of your request, Amazon S3 manages both the encryption, as it writes to disks, and decryption, when you access your objects.

Please review these three options for Server Side Encryption on S3:



Protecting data using server-side encryption

PDF | Kindle | RSS

Server-side encryption is the encryption of data at its destination by the application or service that receives it. Amazon S3 encrypts your data at the object level as it writes it to disks in its data centers and decrypts it for you when you access it. As long as you authenticate your request and you have access permissions, there is no difference in the way you access encrypted or unencrypted objects. For example, if you share your objects using a presigned URL, that URL works the same way for both encrypted and unencrypted objects. Additionally, when you list objects in your bucket, the list API returns a list of all objects, regardless of whether they are encrypted.

Note

You can't apply different types of server-side encryption to the same object simultaneously.

You have three mutually exclusive options, depending on how you choose to manage the encryption keys.

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

When you use Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3), each object is encrypted with a unique key. 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. For more information, see Protecting Data Using Server-Side Encryption with Amazon S3-Managed Encryption Keys (SSE-S3).

Server-Side Encryption with Customer Master Keys (CMKs) Stored in AWS Key Management Service (SSE-KMS)

Server-Side Encryption with Customer Master Keys (CMKs) Stored in AWS Key Management Service (SSE-KMS) is similar to SSE-S3, but with some additional benefits and charges for using this service. There are separate permissions for the use of a CMK that provides added protection against unauthorized access of your objects in Amazon S3. SSE-KMS also provides you with an audit trail that shows when your CMK was used and by whom. Additionally, you can create and manage customer managed CMKs or use AWS managed CMKs that are unique to you, your service, and your Region. For more information, see Protecting Data Using Server-Side Encryption with CMKs Stored in AWS Key Management Service (SSE-KMS).

Server-Side Encryption with Customer-Provided Keys (SSE-C)

With Server-Side Encryption with Customer-Provided Keys (SSE-C), you manage the encryption keys and Amazon S3 manages the encryption, as it writes to disks, and decryption, when you access your objects. For more information, see Protecting data using server-side encryption with customer-provided encryption keys (SSE-C).

via - https://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html

Incorrect options:

SSE-KMS - Server-Side Encryption with Customer Master Keys (CMKs) stored in AWS Key Management Service (SSE-KMS) is similar to SSE-S3. SSE-KMS provides you with an audit trail that shows when your CMK was used and by whom. Additionally, you can create and manage customer-managed CMKs or use AWS managed CMKs that are unique to you, your service, and your Region.

Client-Side Encryption - You can encrypt the 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-S3 - When you use Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3), each object is encrypted with a unique key. As an additional safeguard, it encrypts the key itself with a master key that it regularly rotates. So this option is incorrect.

Reference:

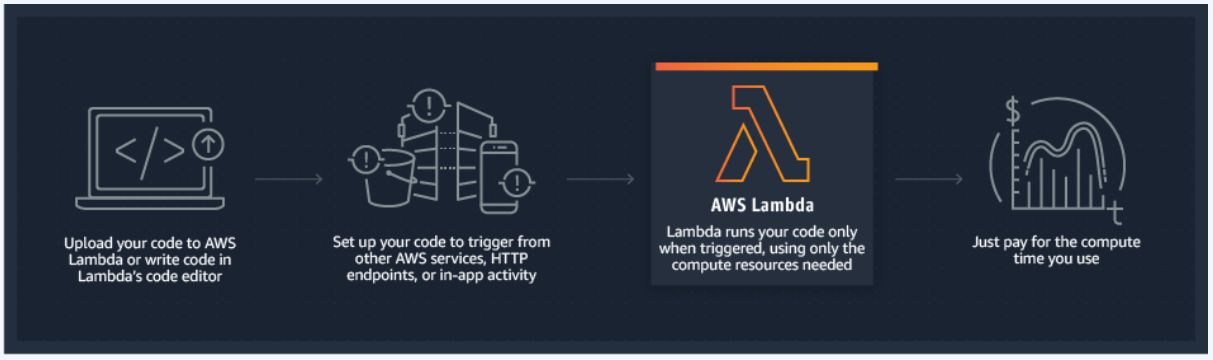
https://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html

Question 34:
Your company wants to move away from manually managing Lambda in the AWS console and wants to upload and update them using AWS CloudFormation.
How do you declare an AWS Lambda function in CloudFormation? (Select two)

- Upload all the code to CodeCommit and refer to the CodeCommit Repository in AWS::Lambda::Function block
- Write the AWS Lambda code inline in CloudFormation in the AWS::Lambda::Function block as long as there are no third-party dependencies(Correct)
- Upload all the code as a zip to S3 and refer the object in AWS::Lambda::Function block(Correct)
- Upload all the code as a folder to S3 and refer the folder in AWS::Lambda::Function block
- Write the AWS Lambda code inline in CloudFormation in the AWS::Lambda::Function block and reference the dependencies as a zip file stored in S3

Explanation

Correct options:
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/
Upload all the code as a zip to S3 and refer the object in AWS::Lambda::Function block
You can upload all the code as a zip to S3 and refer the object in AWS::Lambda::Function block.
The AWS::Lambda::Function resource creates a Lambda function. To create a function, you need a deployment package and an execution role. The deployment package contains your function code.
Write the AWS Lambda code inline in CloudFormation in the AWS::Lambda::Function block as long as there are no third-party dependencies
The other option is to write the code inline for Node.js and Python as long as there are no dependencies for your code, besides the dependencies already provided by AWS in your Lambda Runtime (aws-sdk and cfn-response and many other AWS related libraries are preloaded via, for example, boto3 (python) in the lambda instances.)
YAML template for creating a Lambda function:
Type: AWS::Lambda::Function
Properties:
  Code:
    Code
  DeadLetterConfig:

DeadLetterConfig
Description: String
Environment:
Environment
FileSystemConfigs:
- FileSystemConfig
FunctionName: String
Handler: String
KmsKeyArn: String
Layers:
- String
MemorySize: Integer
ReservedConcurrentExecutions: Integer
Role: String
Runtime: String
Tags:
- Tag
Timeout: Integer
TracingConfig:
TracingConfig
VpcConfig:
VpcConfig

Incorrect options:
Upload all the code to CodeCommit and refer to the CodeCommit Repository in AWS::Lambda::Function block
Upload all the code as a folder to S3 and refer the folder in AWS::Lambda::Function block
Write the AWS Lambda code inline in CloudFormation in the AWS::Lambda::Function block and reference the dependencies as a zip file stored in S3
These three options contradict the explanation provided earlier. So these are incorrect.
Reference:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-lambda-function-code.html
Question 35:
A business-critical mobile application uses Amazon Cognito user pools with multi-factor authentication (MFA) enabled for all its users. The application manages confidential data about the company's sales forecasts and product launches. Considering the highly critical nature of the application, the company wants to track every user login activity via a notification sent as an email to the security team.
Which of the following would you recommend as the MOST optimal way of implementing this requirement within a short period?

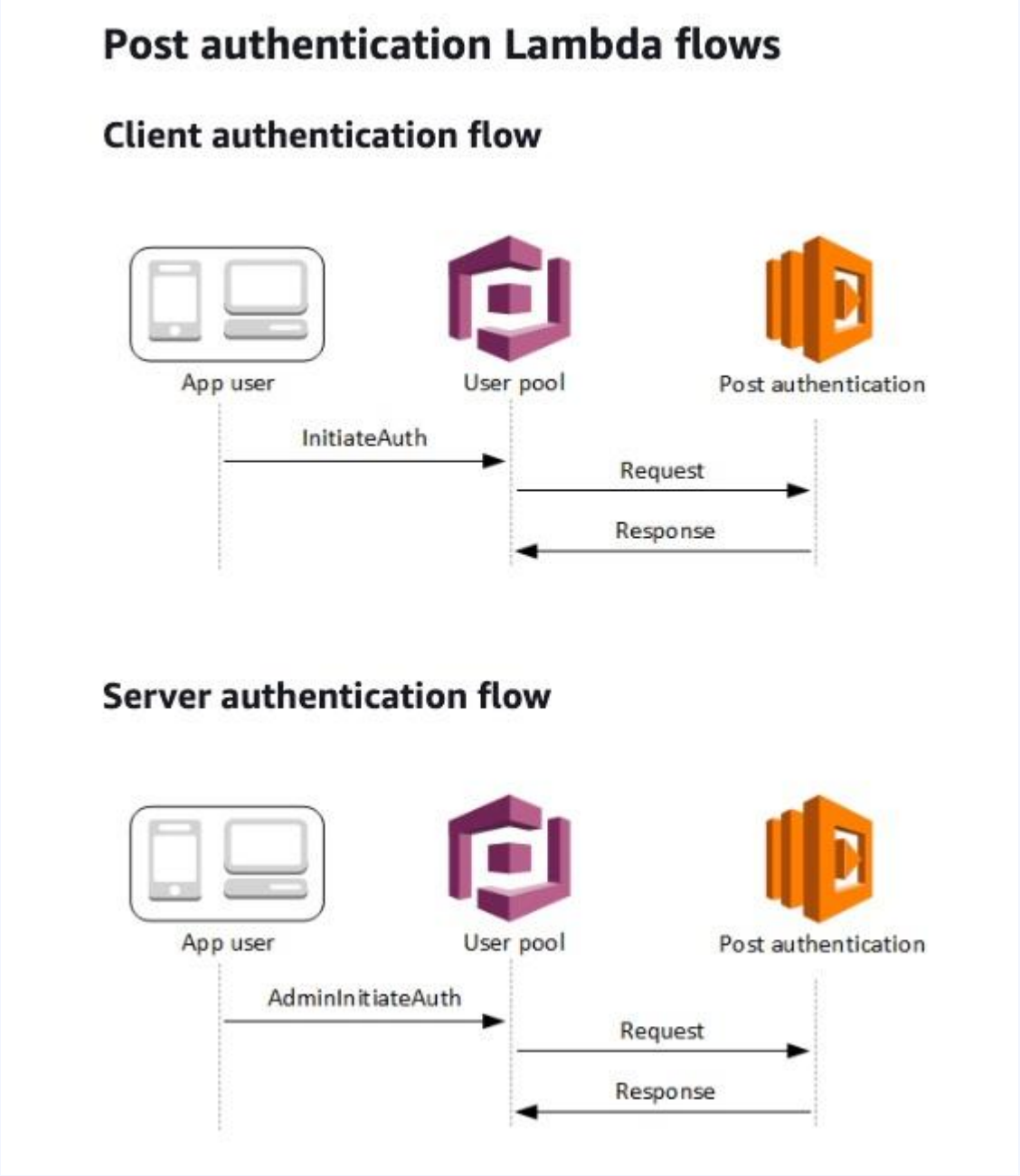
- Create an AWS Lambda function that uses Amazon Simple Email Service to send an email notification to the concerned security team. Configure this function as Amazon Cognito pre-authentication Lambda trigger

- Configure Amazon Cognito user pools authenticated API operations and MFA API operations to send all login data to Amazon Kinesis Data Streams. Configure an AWS Lambda function to analyze these streams and trigger an SNS notification to the security team based on user access
- Configure an AWS Lambda function as a trigger to Amazon Cognito identity pools authenticated API operations. Create the Lambda function to utilize the Amazon Simple Email Service to send an email notification to the concerned security team
- Create an AWS Lambda function that uses Amazon Simple Email Service to send an email notification to the concerned security team. Configure this function as Amazon Cognito post-authentication Lambda trigger(Correct)

Explanation

Correct option:
Create an AWS Lambda function that uses Amazon Simple Email Service to send an email notification to the concerned security team. Configure this function as Amazon Cognito post-authentication Lambda trigger
Amazon Cognito invokes Post authentication Lambda trigger after signing in a user, you can add custom logic after Amazon Cognito authenticates the user.

Post-authentication Lambda flows:



via - https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-lambda-post-authentication.html

Incorrect options:

Create an AWS Lambda function that uses Amazon Simple Email Service to send an email notification to the concerned security team. Configure this function as Amazon Cognito pre-authentication Lambda trigger - Pre-authentication Lambda trigger: Amazon Cognito invokes this trigger when a user attempts to sign in so that you can create custom validation that accepts or denies the authentication request. This is not useful for the current use case since we want to track user login activity which happens post-authentication.

Configure an AWS Lambda function as a trigger to Amazon Cognito identity pools authenticated API operations. Create the Lambda function to utilize the Amazon Simple Email Service to send an email notification to the concerned security team - This statement is incorrect. Amazon Cognito identity

pools (federated identities) enable you to create unique identities for your users and federate them with identity providers. Cognito identity pools are for authorization and not for authentication. Configure Amazon Cognito user pools authenticated API operations and MFA API operations to send all login data to Amazon Kinesis Data Streams. Configure an AWS Lambda function to analyze these streams and trigger an SNS notification to the security team based on user access - This is a made-up option given only as a distractor.

References:

https://docs.aws.amazon.com/cognito/latest/developerguide/user-pool-lambda-post-authentication.html

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

Question 36:
A company has a new media application that utilizes an Amazon CloudFront distribution that accesses the S3 bucket by using an origin access identity (OAI). The S3 bucket has an explicit access denial for all other users. A developer wants to allow access to the login page for unauthenticated users while ensuring the security of all private content that has restricted viewer access.
Which of the following will you recommend?

- Configure a second cache behavior to the distribution having the same origin as the default cache behavior and have the path pattern for the second cache behavior as * with viewer access as restricted. Modify the default cache behavior's path pattern to the path of the login page and have the viewer access as unrestricted
- Configure a second cache behavior to the distribution having the same origin as the default cache behavior and have the path pattern for the second cache behavior as the path of the login page with viewer access as unrestricted. Keep the default cache behavior's settings unchanged(Correct)
- Configure a second origin as the failover origin for the default behavior of the original distribution and have the path pattern for the second origin as the path of the login page with viewer access as unrestricted. Keep the behavior for the primary origin unchanged
- Configure a new distribution having the same origin as the original distribution and set the path pattern for the default cache behavior of the new distribution as the path of the login page with viewer access as unrestricted. Keep the default cache behavior of the original distribution unchanged

Explanation

Correct option:
Configure a second cache behavior to the distribution having the same origin as the default cache behavior and have the path pattern for the second cache behavior as the path of the login page with viewer access as unrestricted. Keep the default cache behavior's settings unchanged
Cache behavior describes how CloudFront processes requests. You must create at least as many cache behaviors (including the default cache behavior) as you have origins if you want CloudFront to serve objects from all of the origins. Each cache behavior specifies the one origin from which you want CloudFront to get objects. If you have two origins and only the default cache behavior, the default cache behavior will cause CloudFront to get objects from one of the origins, but the other origin is never used.
The pattern (for example, images/*.jpg) specifies which requests to apply the behavior to. When CloudFront receives a viewer request, the requested path is compared with path patterns in the order in which cache behaviors are listed in the distribution. The path pattern for the default cache behavior is * and cannot be changed. If the request for an object does not match the path pattern for any cache behaviors, CloudFront applies the behavior in the default cache behavior.
For the given use case, you need to add a second cache behavior to the distribution having the same origin as the default cache behavior and list it above the default cache behavior in the distribution. The second cache behavior should have the path pattern as the path of the login page with viewer access set as unrestricted. This would allow access to the login page for unauthenticated users. Since the default cache behavior's settings remain unchanged, it ensures the security of all private content that continues to have restricted viewer access.
Incorrect options:

Configure a second cache behavior to the distribution having the same origin as the default cache behavior and have the path pattern for the second cache behavior as * with viewer access as restricted. Modify the default cache behavior's path pattern to the path of the login page and have the viewer access as unrestricted - This option is incorrect since the path pattern for the default cache behavior is always * and cannot be changed.

Configure a new distribution having the same origin as the original distribution and set the path pattern for the default cache behavior of the new distribution as the path of the login page with viewer access as unrestricted. Keep the default cache behavior of the original distribution unchanged - If you have two origins and only the default cache behavior, the default cache behavior will cause CloudFront to get objects from one of the origins, but the other origin is never used. So this option is incorrect.

Configure a second origin as the failover origin for the default behavior of the original distribution and have the path pattern for the second origin as the path of the login page with viewer access as unrestricted. Keep the behavior for the primary origin unchanged - You can set up CloudFront with origin failover for scenarios that require high availability. To get started, you create an origin group with two origins: a primary and a secondary. If the primary origin is unavailable or returns specific HTTP response status codes that indicate a failure, CloudFront automatically switches to the secondary origin. 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. Finally, you create or update a cache behavior to use the origin group.

This option is incorrect since the failover kicks in only when the primary is unavailable. Therefore, access to the login page and the rest of the content will never work together.

References:

https://docs.aws.amazon.com/cloudfront/latest/APIReference/API_CacheBehavior.html

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

Question 37:
You are looking to invoke an AWS Lambda function every hour (similar to a cron job) in a serverless way.
Which event source should you use for your AWS Lambda function?

- CloudWatch Events(Correct)
- Kinesis
- SQS
- Amazon S3

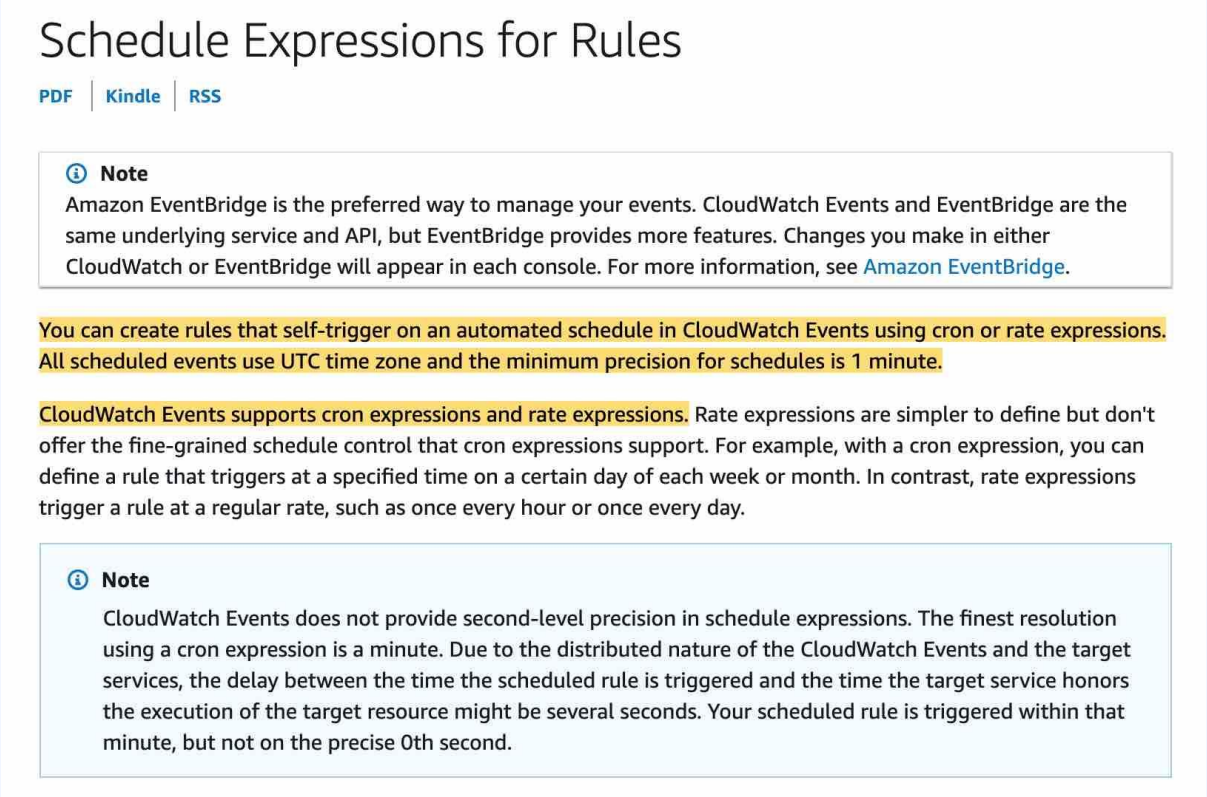
Explanation

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

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

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

Schedule Expressions for CloudWatch Events Rules:



Schedule Expressions for Rules

PDF | Kindle | RSS

Note

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

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

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

Note

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

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

Incorrect options:

Amazon S3

SQS

Kinesis

These three AWS services don't have cron capabilities, so these options are incorrect.

References:

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

https://docs.aws.amazon.com/lambda/latest/dg/with-scheduled-events.html

Question 38:
An IT company leverages CodePipeline to automate its release pipelines. The development team wants to write a Lambda function that will send notifications for state changes within the pipeline. As a Developer Associate, which steps would you suggest to associate the Lambda function with the event source?

- Set up an Amazon CloudWatch Events rule that uses CodePipeline as an event source with the target as the Lambda function(Correct)
- Use the Lambda console to configure a trigger that invokes the Lambda function with CodePipeline as the event source
- Set up an Amazon CloudWatch alarm that monitors status changes in Code Pipeline and triggers the Lambda function
- Use the CodePipeline console to set up a trigger for the Lambda function

Explanation

Correct option:
Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in Amazon Web Services (AWS) resources. Using simple rules that you can quickly set up, you can match events and route them to one or more target functions or streams.
CloudWatch Events Key Concepts:

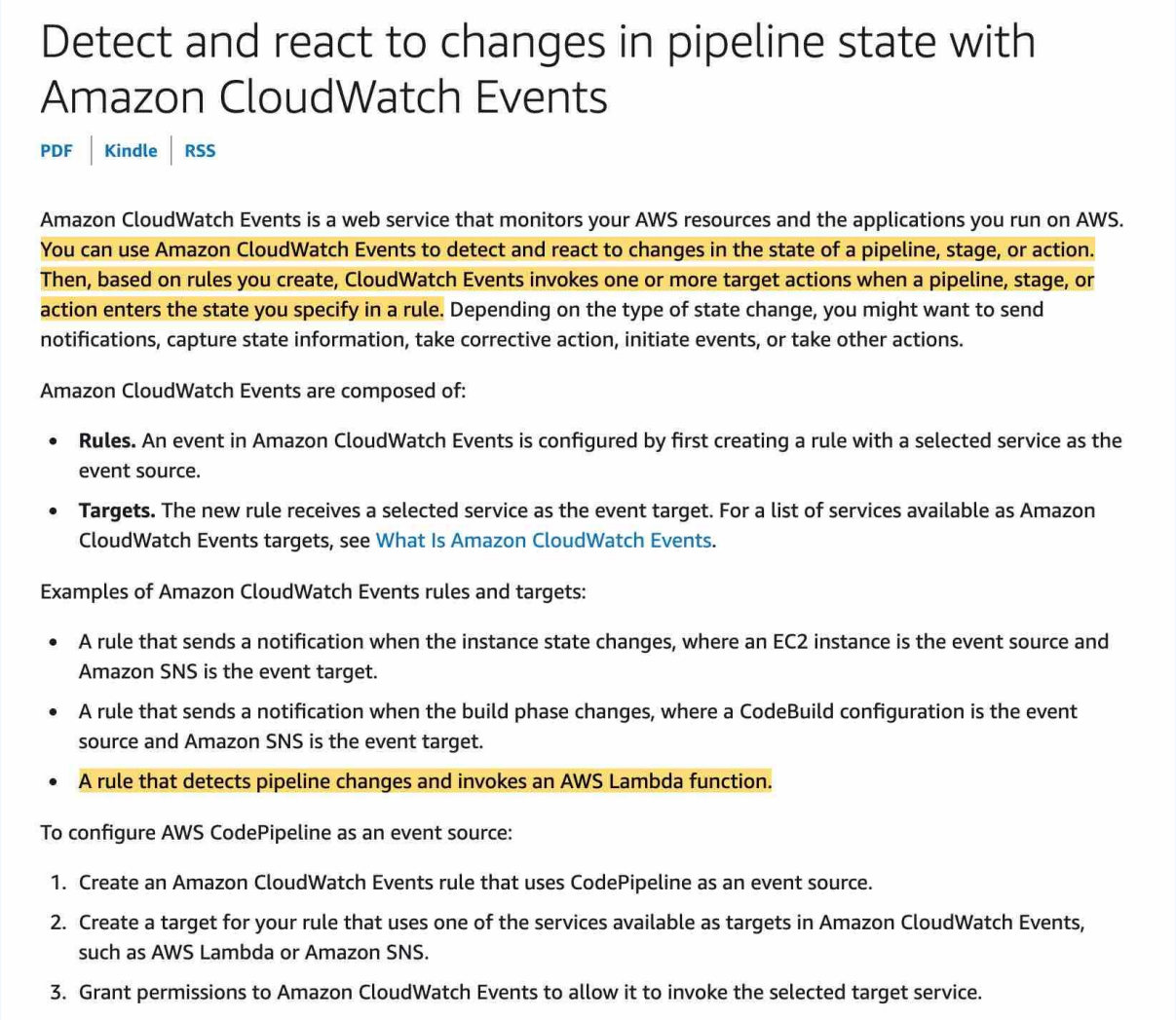
Before you begin using CloudWatch Events, you should understand the following concepts:

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

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

via -
https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/WhatIsCloudWatchEvents.html
Set up an Amazon CloudWatch Events rule that uses CodePipeline as an event source with the target as the Lambda function

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. For the given use-case, you can set up a rule that detects pipeline changes and invokes an AWS Lambda function.
Amazon CloudWatch Events With CodePipeline:



Detect and react to changes in pipeline state with Amazon CloudWatch Events

PDF | Kindle | RSS

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. Depending on the type of state change, you might want to send notifications, capture state information, take corrective action, initiate events, or take other actions.

Amazon CloudWatch Events are composed of:

- **Rules.** An event in Amazon CloudWatch Events is configured by first creating a rule with a selected service as the event source.
- **Targets.** The new rule receives a selected service as the event target. For a list of services available as Amazon CloudWatch Events targets, see What Is Amazon CloudWatch Events.

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.

To configure AWS CodePipeline as an event source:

1. Create an Amazon CloudWatch Events rule that uses CodePipeline as an event source.
2. Create a target for your rule that uses one of the services available as targets in Amazon CloudWatch Events, such as AWS Lambda or Amazon SNS.
3. Grant permissions to Amazon CloudWatch Events to allow it to invoke the selected target service.

https://docs.aws.amazon.com/codepipeline/latest/userguide/detect-state-changes-cloudwatch-events.html
Incorrect options:
Set up an Amazon CloudWatch alarm that monitors status changes in Code Pipeline and triggers the Lambda function - As mentioned in the explanation above, you need to use a CloudWatch event and not CloudWatch alarm for this use-case.
Use the Lambda console to configure a trigger that invokes the Lambda function with CodePipeline as the event source - You cannot create a trigger with CodePipeline as the event source via the Lambda Console.
Use the CodePipeline console to set up a trigger for the Lambda function - CodePipeline console cannot be used to configure a trigger for a Lambda function.
References:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/WhatIsCloudWatchEvents.html

https://docs.aws.amazon.com/codepipeline/latest/userguide/detect-state-changes-cloudwatch-events.html

Question 39:
Your Lambda function processes files for your customers and as part of that process, it creates a lot of intermediary files it needs to store on its disk and then discard.
What is the best way to store temporary files for your Lambda functions that will be discarded when the function stops running?

- Create a tmp/ directory in the source zip file and use it
- Use the local directory /tmp(Correct)
- Use the local directory /opt
- Use an S3 bucket

Explanation

Correct option:
Use the local directory /tmp
This is 512MB of temporary space you can use for your Lambda functions.
Incorrect options:
Create a tmp/ directory in the source zip file and use it - This option has been added as a distractor, as you can't access a directory within a zip file.
Use the local directory /opt - This option has been added as a distractor. This path is not accessible.
Use an S3 bucket - This won't be temporary after the Lambda function is deleted, so this option is incorrect.
Reference:
https://docs.aws.amazon.com/lambda/latest/dg/limits.html
Question 40:
You are using AWS SQS FIFO queues to get the ordering of messages on a per user_id basis. On top of this, you would like to make sure that duplicate messages should not be sent to SQS as this would cause application failure.
As a developer, which message parameter should you set for deduplicating messages?

- ReceiveRequestAttemptId
- MessageGroupId
- ContentBasedDeduplication
- MessageDeduplicationId(Correct)

Explanation

Correct option:
AWS FIFO queues are designed to enhance messaging between applications when the order of operations and events has to be enforced.

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, for example:

- Ensure that user-entered commands are executed in the right order.
- Display the correct product price by sending price modifications in the right order.
- Prevent a student from enrolling in a course before registering for an account.

FIFO queues also provide exactly-once processing but have a limited number of transactions per second (TPS):

- If you use batching, FIFO queues support up to 3,000 transactions per second, per API method (`SendMessageBatch`, `ReceiveMessage`, or `DeleteMessageBatch`). The 3000 transactions represent 300 API calls, each with a batch of 10 messages. To request a quota increase, submit a support request.
- Without batching, FIFO queues support up to 300 API calls per second, per API method (`SendMessage`, `ReceiveMessage`, or `DeleteMessage`).

Note

- Amazon SNS isn't currently compatible with FIFO queues.
- The name of a FIFO queue must end with the `.fifo` suffix. The suffix counts towards the 80-character queue name quota. To determine whether a queue is FIFO, you can check whether the queue name ends with the suffix.

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

MessageDeduplicationId

The message deduplication ID is the token used for the deduplication of sent messages. If a message with a particular message deduplication ID is sent successfully, any messages sent with the same message deduplication ID are accepted successfully but aren't delivered during the 5-minute deduplication interval.

Incorrect options:

MessageGroupId - The message group ID is the tag that specifies that a message belongs to a specific message group. Messages that belong to the same message group are always processed one by one, in a strict order relative to the message group (however, messages that belong to different message groups might be processed out of order).

ReceiveRequestAttemptId - This parameter applies only to FIFO (first-in-first-out) queues. The token is used for deduplication of ReceiveMessage calls. If a networking issue occurs after a ReceiveMessage action, and instead of a response you receive a generic error, you can retry the same action with an identical ReceiveRequestAttemptId to retrieve the same set of messages, even if their visibility timeout has not yet expired.

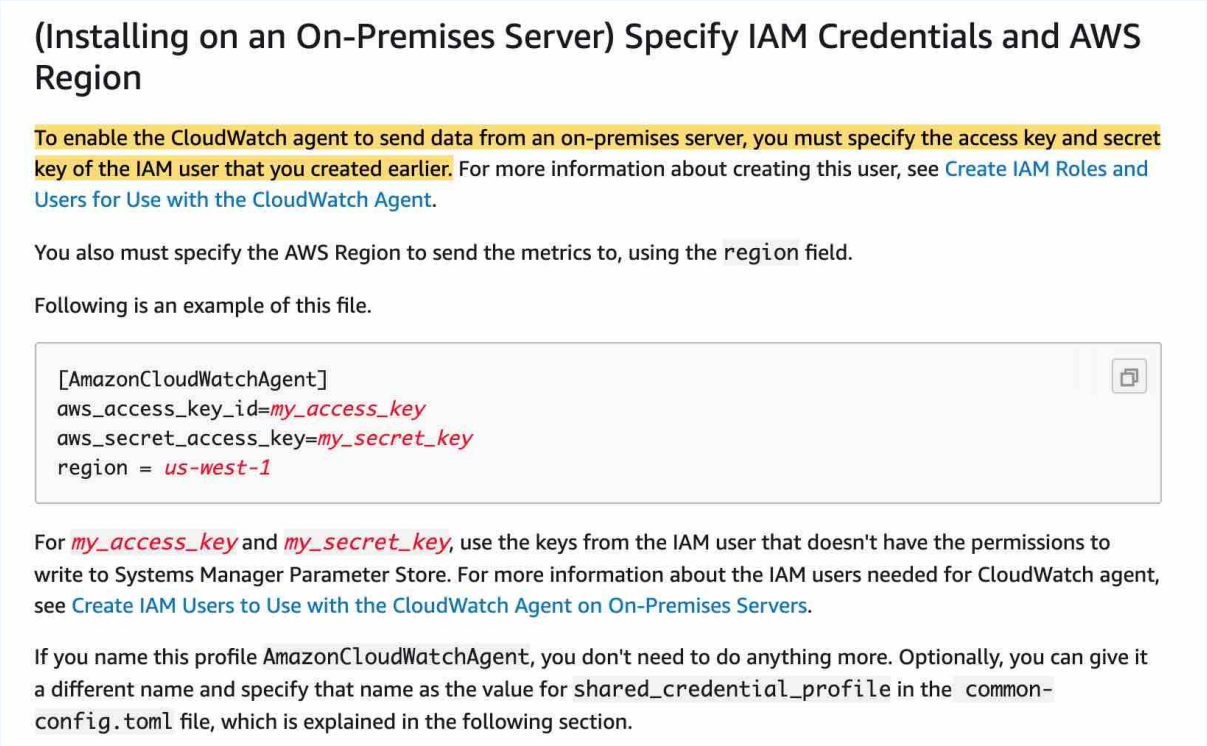
ContentBasedDeduplication - This is not a message parameter, but a queue setting. Enable content-based deduplication to instruct Amazon SQS to use an SHA-256 hash to generate the message deduplication ID using the body of the message - but not the attributes of the message.
Reference:
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/using-messagededuplicationid-property.html

Question 41:
A development team has a mix of applications hosted on-premises as well as on EC2 instances. The on-premises application controls all applications deployed on the EC2 instances. In case of any errors, the team wants to leverage Amazon CloudWatch to monitor and troubleshoot the on-premises application.
As a Developer Associate, which of the following solutions would you suggest to address this use-case?

- Configure CloudWatch Logs to directly read the logs from the on-premises server
- Upload log files from the on-premises server to an EC2 instance which further forwards the logs to CloudWatch
- Configure the CloudWatch agent on the on-premises server by using IAM user credentials with permissions for CloudWatch(Correct)
- Upload log files from the on-premises server to S3 and let CloudWatch process the files from S3

Explanation

Correct option:
Configure the CloudWatch agent on the on-premises server by using IAM user credentials with permissions for CloudWatch
The CloudWatch agent enables you to do the following:
Collect system-level metrics from on-premises servers. These can include servers in a hybrid environment as well as servers not managed by AWS.
Collect logs from Amazon EC2 instances and on-premises servers, running either Linux or Windows Server.
To enable the CloudWatch agent to send data from an on-premises server, you must specify the access key and secret key of the IAM user that you created earlier.



(Installing on an On-Premises Server) Specify IAM Credentials and AWS Region

To enable the CloudWatch agent to send data from an on-premises server, you must specify the access key and secret key of the IAM user that you created earlier. For more information about creating this user, see Create IAM Roles and Users for Use with the CloudWatch Agent.

You also must specify the AWS Region to send the metrics to, using the `region` field.

Following is an example of this file.

```
[AmazonCloudWatchAgent]
aws_access_key_id=my_access_key
aws_secret_access_key=my_secret_key
region = us-west-1
```

For *my_access_key* and *my_secret_key*, use the keys from the IAM user that doesn't have the permissions to write to Systems Manager Parameter Store. For more information about the IAM users needed for CloudWatch agent, see Create IAM Users to Use with the CloudWatch Agent on On-Premises Servers.

If you name this profile `AmazonCloudWatchAgent`, you don't need to do anything more. Optionally, you can give it a different name and specify that name as the value for `shared_credential_profile` in the `common-config.toml` file, which is explained in the following section.

via - https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/install-CloudWatch-Agent-on-premise.html

Incorrect options:

Configure CloudWatch Logs to directly read the logs from the on-premises server - This is a made-up option as you cannot have CloudWatch Logs directly communicate with the on-premises server. You have to go via the CloudWatch Agent.

Upload log files from the on-premises server to an EC2 instance which further forwards the logs to CloudWatch

Upload log files from the on-premises server to S3 and let CloudWatch process the files from S3

Both these options require significant customizations and still will not be as neatly integrated with CloudWatch as compared to just using the CloudWatch Agent which is available off-the-shelf.

Reference:

https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/install-CloudWatch-Agent-on-premise.html

Question 42:
A company has recently launched a new gaming application that the users are adopting rapidly. The company uses RDS MySQL as the database. The development team wants an urgent solution to this issue where the rapidly increasing workload might exceed the available database storage.
As a developer associate, which of the following solutions would you recommend so that it requires minimum development effort to address this requirement?

- Create read replica for RDS MySQL
- Migrate RDS MySQL database to Aurora which offers storage auto-scaling
- Enable storage auto-scaling for RDS MySQL(Correct)
- Migrate RDS MySQL database to DynamoDB which automatically allocates storage space when required

Explanation

Correct option:
Enable storage auto-scaling for RDS MySQL

- If your workload is unpredictable, you can enable storage autoscaling for an Amazon RDS DB instance. With storage autoscaling enabled, when Amazon RDS detects that you are running out of free database space it automatically scales up your storage. Amazon RDS starts a storage modification for an autoscaling-enabled DB instance when these factors apply:

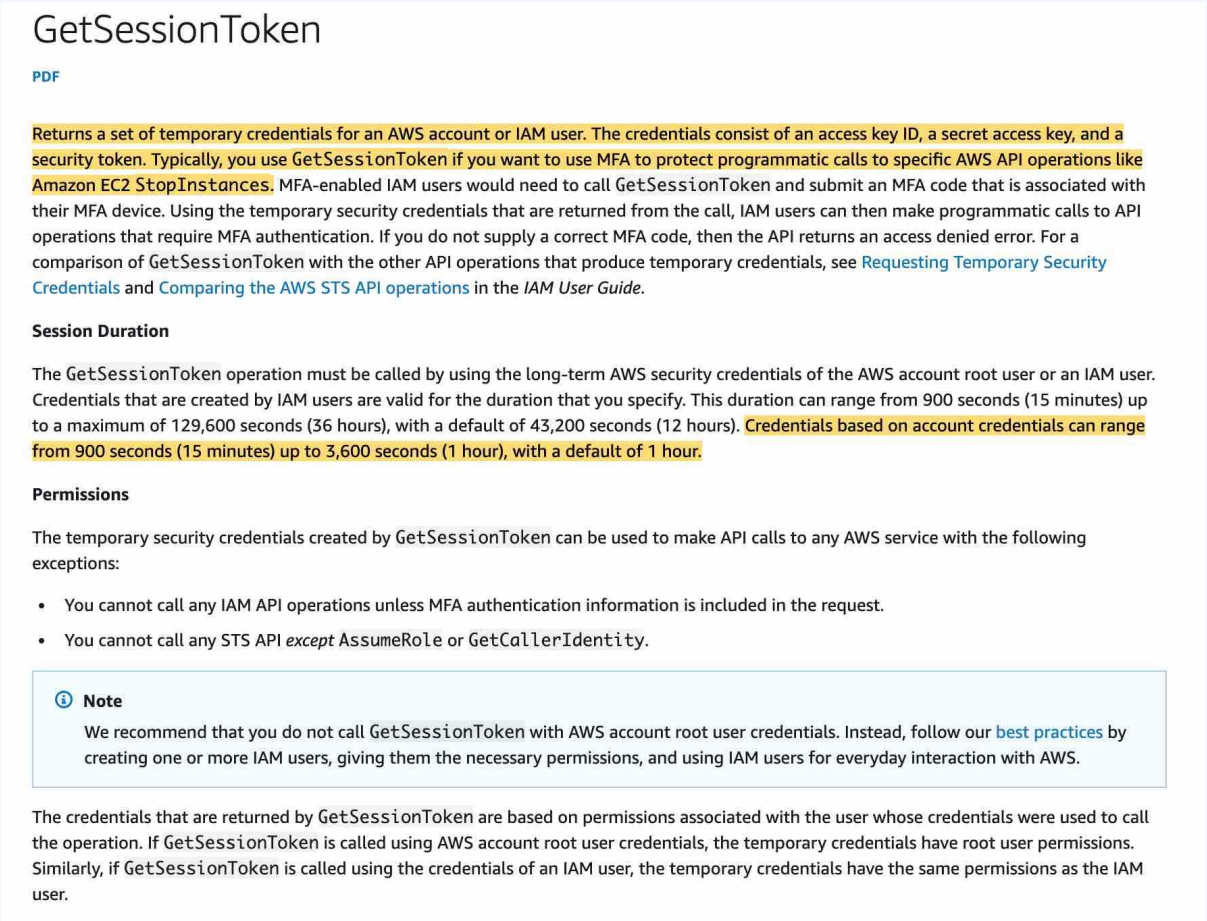
Free available space is less than 10 percent of the allocated storage.
The low-storage condition lasts at least five minutes.
At least six hours have passed since the last storage modification.
The maximum storage threshold is the limit that you set for autoscaling the DB instance. You can't set the maximum storage threshold for autoscaling-enabled instances to a value greater than the maximum allocated storage.
Incorrect options:
Migrate RDS MySQL to Aurora which offers storage auto-scaling - Although Aurora offers automatic storage scaling, this option is ruled out since it involves significant systems administration effort to migrate from RDS MySQL to Aurora. It is much easier to just enable storage auto-scaling for RDS MySQL.
Migrate RDS MySQL database to DynamoDB which automatically allocates storage space when required - This option is ruled out since DynamoDB is a NoSQL database which implies significant development effort to change the application logic to connect and query data from the underlying database. It is much easier to just enable storage auto-scaling for RDS MySQL.
Create read replica for RDS MySQL - Read replicas make it easy to take advantage of supported engines' built-in replication functionality to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads. You can create multiple read replicas for a given source DB Instance and distribute your application’s read traffic amongst them. This option acts as a distractor as read replicas cannot help to automatically scale storage for the primary database.
Reference:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PIOPS.StorageTypes.html

Question 43:
A developer created an online shopping application that runs on EC2 instances behind load balancers. The same web application version is hosted on several EC2 instances and the instances run in an Auto Scaling group. The application uses STS to request credentials but after an hour your application stops working.
What is the most likely cause of this issue?

- Your IAM policy is wrong
- The IAM service is experiencing downtime once an hour
- A lambda function revokes your access every hour
- Your application needs to renew the credentials after 1 hour when they expire(Correct)

Explanation

Correct option:
Your application needs to renew the credentials after 1 hour when they expire
AWS Security Token Service (AWS STS) is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users). By default, AWS Security Token Service (STS) is available as a global service, and all AWS STS requests go to a single endpoint at https://sts.amazonaws.com. Credentials that are created by using account credentials can range from 900 seconds (15 minutes) up to a maximum of 3,600 seconds (1 hour), with a default of 1 hour. Hence you need to renew the credentials post expiry.



GetSessionToken

PDF

Returns a set of temporary credentials for an AWS account or IAM user. The credentials consist of an access key ID, a secret access key, and a security token. Typically, you use `GetSessionToken` if you want to use MFA to protect programmatic calls to specific AWS API operations like Amazon EC2 `StopInstances`. MFA-enabled IAM users would need to call `GetSessionToken` and submit an MFA code that is associated with their MFA device. Using the temporary security credentials that are returned from the call, IAM users can then make programmatic calls to API operations that require MFA authentication. If you do not supply a correct MFA code, then the API returns an access denied error. For a comparison of `GetSessionToken` with the other API operations that produce temporary credentials, see Requesting Temporary Security Credentials and Comparing the AWS STS API operations in the *IAM User Guide*.

Session Duration

The `GetSessionToken` operation must be called by using the long-term AWS security credentials of the AWS account root user or an IAM user. Credentials that are created by IAM users are valid for the duration that you specify. This duration can range from 900 seconds (15 minutes) up to a maximum of 129,600 seconds (36 hours), with a default of 43,200 seconds (12 hours). Credentials based on account credentials can range from 900 seconds (15 minutes) up to 3,600 seconds (1 hour), with a default of 1 hour.

Permissions

The temporary security credentials created by `GetSessionToken` can be used to make API calls to any AWS service with the following exceptions:

- You cannot call any IAM API operations unless MFA authentication information is included in the request.
- You cannot call any STS API *except* `AssumeRole` or `GetCallerIdentity`.

Note
We recommend that you do not call `GetSessionToken` with AWS account root user credentials. Instead, follow our best practices by creating one or more IAM users, giving them the necessary permissions, and using IAM users for everyday interaction with AWS.

The credentials that are returned by `GetSessionToken` are based on permissions associated with the user whose credentials were used to call the operation. If `GetSessionToken` is called using AWS account root user credentials, the temporary credentials have root user permissions. Similarly, if `GetSessionToken` is called using the credentials of an IAM user, the temporary credentials have the same permissions as the IAM user.

via - https://docs.aws.amazon.com/STS/latest/APIReference/API_GetSessionToken.html

Incorrect options:
Your IAM policy is wrong - If your policy was wrong, a reboot would not solve the issue.
A lambda function revokes your access every hour - Revoking can be done by an IAM policy. Lambda function cannot revoke access.
The IAM service is experiencing downtime once an hour - The IAM service is reliable as it's managed by AWS.
Reference:
https://docs.aws.amazon.com/STS/latest/APIReference/API_GetSessionToken.html

Question 44:
You were assigned to a project that requires the use of the AWS CLI to build a project with AWS CodeBuild. Your project's root directory includes the buildspec.yml file to run build commands and would like your build artifacts to be automatically encrypted at the end.
How should you configure CodeBuild to accomplish this?

- Use In Flight encryption (SSL)
- Specify a KMS key to use(Correct)
- Use the AWS Encryption SDK
- Use an AWS Lambda Hook

Explanation

Correct option:
Specify a KMS key to use
AWS Key Management Service (KMS) makes it easy for you to create and manage cryptographic keys and control their use across a wide range of AWS services and in your applications.
For AWS CodeBuild to encrypt its build output artifacts, it needs access to an AWS KMS customer master key (CMK). By default, AWS CodeBuild uses the AWS-managed CMK for Amazon S3 in your AWS account. The following environment variable provides these details:
CODEBUILD_KMS_KEY_ID: The identifier of the AWS KMS key that CodeBuild is using to encrypt the build output artifact (for example, arn:aws:kms:region-ID:account-ID:key/key-ID or alias/key-alias).
Incorrect options:
Use an AWS Lambda Hook - Code hook is used for integration with Lambda and is not relevant for the given use-case.
Use the AWS Encryption SDK - The SDK just makes it easier for you to implement encryption best practices in your application and is not relevant for the given use-case.
Use In-Flight encryption (SSL) - SSL is usually for internet traffic which in this case will be using internal traffic through AWS and is not relevant for the given use-case.
References:
https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-env-vars.html
https://docs.aws.amazon.com/codebuild/latest/userguide/setting-up.html
Question 45:
An IT company uses AWS CloudFormation templates to provision their AWS infrastructure for Amazon EC2, Amazon VPC, and Amazon S3 resources. Using cross-stack referencing, a developer creates a stack called NetworkStack which will export the subnetId that can be used when creating EC2 instances in another stack.
To use the exported value in another stack, which of the following functions must be used?

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

Explanation

Correct option:
!ImportValue

The intrinsic function Fn::ImportValue returns the value of an output exported by another stack. You typically use this function to create cross-stack references.
Incorrect options:
!Ref - Returns the value of the specified parameter or resource.
!GetAtt - Returns the value of an attribute from a resource in the template.
!Sub - Substitutes variables in an input string with values that you specify.
Reference:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-importvalue.html

Question 46:
You are responsible for an application that runs on multiple Amazon EC2 instances. In front of the instances is an Internet-facing load balancer that takes requests from clients over the internet and distributes them to the EC2 instances. A health check is configured to ping the index.html page found in the root directory for the health status. When accessing the website via the internet visitors of the website receive timeout errors.
What should be checked first to resolve the issue?

- Security Groups(Correct)
- The application is down
- IAM Roles
- The ALB is warming up

Explanation

Correct option:
Security Groups
A security group acts as a virtual firewall for your EC2 instances to control incoming and outgoing traffic. Inbound rules control the incoming traffic to your instance, and outbound rules control the outgoing traffic from your instance.
Check the security group rules of your EC2 instance. You need a security group rule that allows inbound traffic from your public IPv4 address on the proper port.
Incorrect options:
IAM Roles - Usually you run into issues with authorization of APIs with roles but not for timeout, so this option does not fit the given use-case.
The application is down - Although you can set a health check for application ping or HTTP, timeouts are usually caused by blocked firewall access.
The ALB is warming up - ALB has a slow start mode which allows a warm-up period before being able to respond to requests with optimal performance. So this is not the issue.
Reference:
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/TroubleshootingInstancesConnecting.html#TroubleshootingInstancesConnectionTimeout

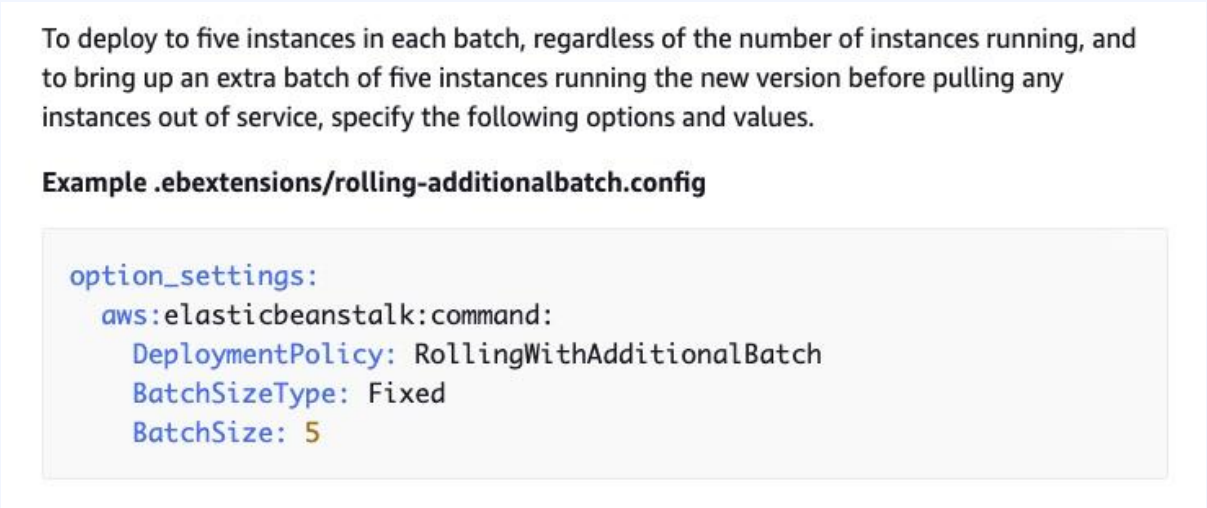
Question 47:
An e-commerce company has deployed its application on AWS Elastic Beanstalk. The Auto Scaling group associated with the Beanstalk environment has three Amazon EC2 instances. When the number of instances falls below two, it severely impacts the performance of the web application. The company currently uses the default all-at-once deployment policy and is looking for an effective strategy for future deployments.
Which of the following represents the most cost-effective deployment strategy for the company?

- Configure an Elastic Load Balancer to front the Auto Scaling Group and choose two different Availability Zones (AZs) for deployment
- Opt for rolling with additional batch deployment strategy. Set the batch size parameter to 1(Correct)
- Opt for rolling deployment strategy. Set the batch size to 2
- Opt for traffic-splitting deployment strategy with traffic split parameter set to 50% of the total traffic

Explanation

Correct option:
Opt for rolling with additional batch deployment strategy. Set the batch size parameter to 1
With rolling deployments, Elastic Beanstalk splits the environment's Amazon EC2 instances into batches and deploys the new version of the application to one batch at a time. It leaves the rest of the instances in the environment running the old version of the application. During a rolling deployment, some instances serve requests with the old version of the application, while instances in completed batches serve other requests with the new version.
To maintain full capacity during deployments, you can configure your environment to launch a new batch of instances before taking any instances out of service. This option is known as a rolling deployment with an additional batch. When the deployment completes, Elastic Beanstalk terminates the additional batch of instances.
For the rolling and rolling with additional batch deployment policies, you can configure Batch size – The size of the set of instances to deploy in each batch. Choose Percentage to configure a percentage of the total number of EC2 instances in the Auto Scaling group (up to 100 percent), or choose Fixed to configure a fixed number of instances (up to the maximum instance count in your environment's Auto Scaling configuration).

Example configuration for rolling With additional batch:



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

Incorrect options:

Opt for a traffic-splitting deployment strategy with a traffic split parameter set to 50% of the total traffic - Traffic-splitting deployments let you perform canary testing as part of your application deployment. In a traffic-splitting deployment, Elastic Beanstalk launches a full set of new instances just like during an immutable deployment. It then forwards a specified percentage of incoming client traffic to the new application version for a specified evaluation period. This turns out to be a costly solution for the given use case.

Configure an Elastic Load Balancer to front the Auto Scaling Group and choose two different Availability Zones (AZs) for deployment - Adding AZs makes the entire configuration resilient for failures post-deployment. However, it does not represent a valid deployment strategy for the given use case. This option has been added as a distractor.

Opt for a rolling deployment strategy. Set the batch size to 2 - As already discussed, with rolling deployments, Elastic Beanstalk splits the environment's Amazon EC2 instances into batches and deploys the new version of the application to one batch at a time. With a batch size set to 2, only 1 instance will be left for traffic during deployment, impacting the performance of the e-commerce application. Hence, this option is incorrect.

References:

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

Question 48:
You would like your Elastic Beanstalk environment to expose an HTTPS endpoint and an HTTP endpoint. The HTTPS endpoint should be used to get in-flight encryption between your clients and your web servers, while the HTTP endpoint should only be used to redirect traffic to HTTPS and support URLs starting with http://.
What must be done to configure this setup? (Select three)

- Configure your EC2 instances to redirect HTTPS traffic to HTTP
- Open up port 80 & port 443(Correct)
- Assign an SSL certificate to the Load Balancer(Correct)
- Only open up port 80
- Configure your EC2 instances to redirect HTTP traffic to HTTPS(Correct)
- Only open up port 443

Explanation

Correct options:
Assign an SSL certificate to the Load Balancer
This ensures that the Load Balancer can expose an HTTPS endpoint.
Open up port 80 & port 443
This ensures that the Load Balancer will allow both the HTTP (80) and HTTPS (443) protocol for incoming connections
Configure your EC2 instances to redirect HTTP traffic to HTTPS
This ensures traffic originating from HTTP onto the Load Balancer forces a redirect to HTTPS by the EC2 instances before being correctly served, thus ensuring the traffic served is fully encrypted.
Incorrect options:
Only open up port 80 - This is not correct as it would not allow HTTPS traffic (port 443).
Only open up port 443 - This is not correct as it would not allow HTTP traffic (port 80).
Configure your EC2 instances to redirect HTTPS traffic to HTTP - This is not correct as it would force HTTP traffic, instead of HTTPS.
References:
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/configuring-https-httpredirect.html
https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/configuring-https-elb.html
Question 49:
An e-commerce application posts its order transactions in bulk to an accounting application for further processing. Due to changes in the compliance rules, all the transactions are being encrypted with AWS Key Management Service (AWS KMS) key before posting to the accounting application. Post this change, the testers have raised tickets regarding the application receiving a ThrottlingException error.
What measures should a developer take to fix this issue MOST optimally? (Select two)

- Reduce the rate of requests and consider using the backoff and retry logic(Correct)
- Use the data key caching feature with the AWS Encryption SDK encryption library(Correct)
- Send AWS CloudTrail events generated by AWS KMS to Amazon CloudWatch Logs
- Use a bucket-level key for SSE-KMS which will decrease the requested traffic to AWS KMS thereby avoiding the ThrottlingException error
- Write queries in Amazon CloudWatch Logs Insights to track your API request usage and submit an AWS Support case to request a quota increase

Explanation

Correct options:

Reduce the rate of requests and consider using the backoff and retry logic

Each AWS SDK implements automatic retry logic. The AWS SDK for Java automatically retries requests, and you can configure the retry settings. In addition to simple retries, each AWS SDK implements an exponential backoff algorithm for better flow control. The idea behind exponential backoff is to use progressively longer waits between retries for consecutive error responses. You should implement a maximum delay interval, as well as a maximum number of retries. The maximum delay interval and the maximum number of retries are not necessarily fixed values and should be set based on the operation being performed, as well as other local factors, such as network latency.

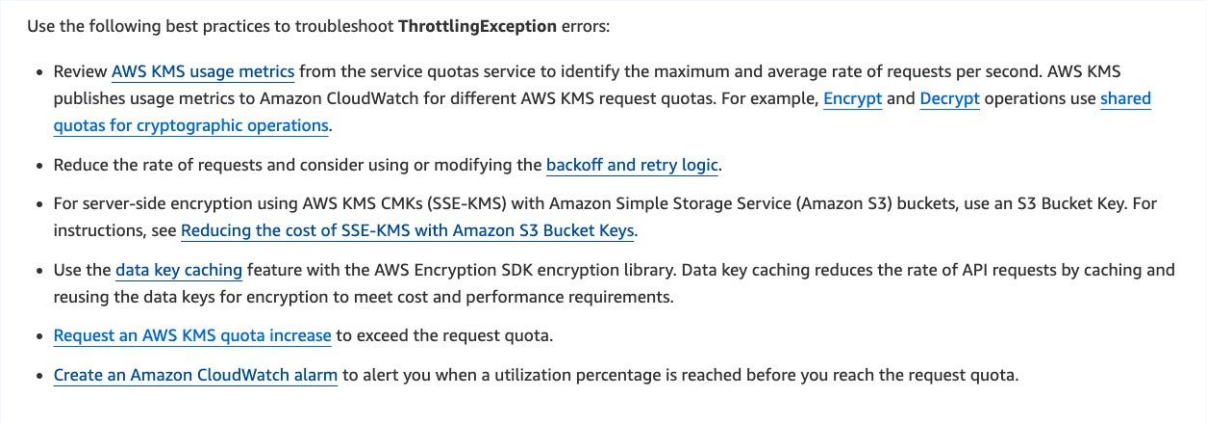
Use the data key caching feature with the AWS Encryption SDK encryption library

Data key caching stores data keys and related cryptographic material in a cache. When you encrypt or decrypt data, the AWS Encryption SDK looks for a matching data key in the cache. If it finds a match, it uses the cached data key rather than generating a new one. Data key caching can improve performance, reduce cost, and help you stay within service limits as your application scales.

Your application can benefit from data key caching if: 1. It can reuse data keys. 2. It generates numerous data keys. 3. Your cryptographic operations are unacceptably slow, expensive, limited, or resource-intensive.

Data key caching is an optional feature of the AWS Encryption SDK that you should use cautiously. By default, the AWS Encryption SDK generates a new data key for every encryption operation. This technique supports cryptographic best practices, which discourage excessive reuse of data keys. In general, use data key caching only when it is required to meet your performance goals. Then, use the data key caching security thresholds to ensure that you use the minimum amount of caching required to meet your cost and performance goals.

Best practices to troubleshoot ThrottlingException errors:



Use the following best practices to troubleshoot **ThrottlingException** errors:

- Review AWS KMS usage metrics from the service quotas service to identify the maximum and average rate of requests per second. AWS KMS publishes usage metrics to Amazon CloudWatch for different AWS KMS request quotas. For example, Encrypt and Decrypt operations use shared quotas for cryptographic operations.
- Reduce the rate of requests and consider using or modifying the backoff and retry logic.
- For server-side encryption using AWS KMS CMKs (SSE-KMS) with Amazon Simple Storage Service (Amazon S3) buckets, use an S3 Bucket Key. For instructions, see Reducing the cost of SSE-KMS with Amazon S3 Bucket Keys.
- Use the data key caching feature with the AWS Encryption SDK encryption library. Data key caching reduces the rate of API requests by caching and reusing the data keys for encryption to meet cost and performance requirements.
- Request an AWS KMS quota increase to exceed the request quota.
- Create an Amazon CloudWatch alarm to alert you when a utilization percentage is reached before you reach the request quota.

via - https://aws.amazon.com/premiumsupport/knowledge-center/kms-throttlingexception-error/

Incorrect options:

Send AWS CloudTrail events generated by AWS KMS to Amazon CloudWatch Logs - Deeper analysis of CloudTrail data sent to CloudWatch logs can help spot throttled API calls. While it is certainly possible to track API usage using CloudTrail data, it is not an optimal way to fix the ThrottlingException error.

Write queries in Amazon CloudWatch Logs Insights to track your API request usage and submit an AWS Support case to request a quota increase - Historically, to understand how close to a request rate quota you were, you had to perform three tasks: (i) send AWS CloudTrail events generated by AWS KMS to Amazon CloudWatch Logs; (ii) write queries in Amazon CloudWatch Logs Insights to

track your API request usage; and (iii) submit an AWS Support case to request a quota increase. Now, you can view your AWS KMS API usage and request quota increases within the AWS Service Quotas console itself without doing any special configuration.
Use a bucket-level key for SSE-KMS which will decrease the requested traffic to AWS KMS thereby avoiding the ThrottlingException error - Amazon S3 bucket has not been mentioned in the given use case and hence this option is irrelevant.
References:
https://docs.aws.amazon.com/general/latest/gr/api-retries.html
https://docs.aws.amazon.com/encryption-sdk/latest/developer-guide/data-key-caching.html
https://aws.amazon.com/blogs/security/manage-your-aws-kms-api-request-rates-using-service-quotas-and-amazon-cloudwatch/

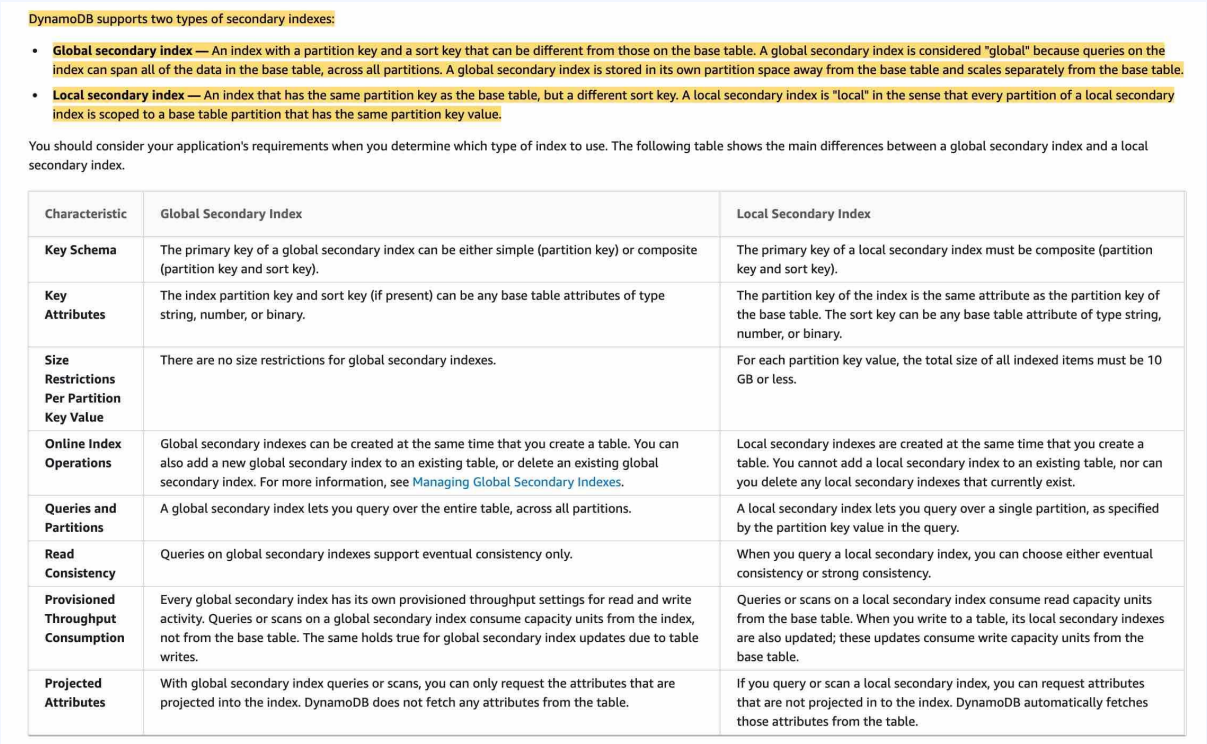
Question 50:
You have created a DynamoDB table to support your application and provisioned RCU and WCU to it so that your application has been running for over a year now without any throttling issues. Your application now requires a second type of query over your table and as such, you have decided to create an LSI and a GSI on a new table to support that use case. One month after having implemented such indexes, it seems your table is experiencing throttling.
Upon looking at the table's metrics, it seems the RCU and WCU provisioned are still sufficient. What's happening?

- Metrics are lagging in your CloudWatch dashboard and you should see the RCU and WCU peaking for the main table in a few minutes
- The GSI is throttling so you need to provision more RCU and WCU to the GSI(Correct)
- The LSI is throttling so you need to provision more RCU and WCU to the LSI
- Adding both an LSI and a GSI to a table is not recommended by AWS best practices as this is a known cause for creating throttles

Explanation

Correct option:
The GSI is throttling so you need to provision more RCU and WCU to the GSI
DynamoDB supports two types of secondary indexes:
Global secondary index — An index with a partition key and a sort key that can be different from those on the base table. A global secondary index is considered "global" because queries on the index can span all of the data in the base table, across all partitions. A global secondary index is stored in its own partition space away from the base table and scales separately from the base table.
Local secondary index — An index that has the same partition key as the base table, but a different sort key. A local secondary index is "local" in the sense that every partition of a local secondary index is scoped to a base table partition that has the same partition key value.

Differences between GSI and LSI:



DynamoDB supports two types of secondary indexes:

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

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

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

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

If you perform heavy write activity on the table, but a global secondary index on that table has insufficient write capacity, then the write activity on the table will be throttled. To avoid potential throttling, the provisioned write capacity for a global secondary index should be equal or greater than the write capacity of the base table since new updates will write to both the base table and global secondary index.

Incorrect options

The LSI is throttling so you need to provision more RCU and WCU to the LSI - LSI use the RCU and WCU of the main table, so you can't provision more RCU and WCU to the LSI.

Adding both an LSI and a GSI to a table is not recommended by AWS best practices as this is a known cause for creating throttles - This option has been added as a distractor. It is fine to have LSI and GSI together.

Metrics are lagging in your CloudWatch dashboard and you should see the RCU and WCU peaking for the main table in a few minutes - This could be a reason, but in this case, the GSI is at fault as the application has been running fine for months.

References:

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

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GSI.html#GSI.ThroughputConsiderations

Question 51:

A company's e-commerce application becomes slow when traffic spikes. The application has a three-tier architecture (web, application and database tier) that uses synchronous transactions. The development team at the company has identified certain bottlenecks in the application tier and it is looking for a long term solution to improve the application's performance.

As a developer associate, which of the following solutions would you suggest to meet the required application response times while accounting for any traffic spikes?

- Leverage horizontal scaling for the web and application tiers by using Auto Scaling groups and Application Load Balancer(Correct)
- Leverage SQS with asynchronous AWS Lambda calls to decouple the application and data tiers
- Leverage horizontal scaling for the application's persistence layer by adding Oracle RAC on AWS
- Leverage vertical scaling for the application instance by provisioning a larger Amazon EC2 instance size

Explanation

Correct option:
Leverage horizontal scaling for the web and application tiers by using Auto Scaling groups and Application Load Balancer - A horizontally scalable system is one that can increase capacity by adding more computers to the system. This is in contrast to a vertically scalable system, which is constrained to running its processes on only one computer; in such systems, the only way to increase performance is to add more resources into one computer in the form of faster (or more) CPUs, memory or storage.
Horizontally scalable systems are oftentimes able to outperform vertically scalable systems by enabling parallel execution of workloads and distributing those across many different computers.
Elastic Load Balancing is used to automatically distribute your incoming application traffic across all the EC2 instances that you are running. You can use Elastic Load Balancing to manage incoming requests by optimally routing traffic so that no one instance is overwhelmed.
To use Elastic Load Balancing with your Auto Scaling group, you attach the load balancer to your Auto Scaling group to register the group with the load balancer. Your load balancer acts as a single point of contact for all incoming web traffic to your Auto Scaling group.
When you use Elastic Load Balancing with your Auto Scaling group, it's not necessary to register individual EC2 instances with the load balancer. Instances that are launched by your Auto Scaling group are automatically registered with the load balancer. Likewise, instances that are terminated by your Auto Scaling group are automatically deregistered from the load balancer.
This option will require fewer design changes, it's mostly configuration changes and the ability for the web/application tier to be able to communicate across instances. Hence, this is the right solution for the current use case.
Incorrect options:
Leverage SQS with asynchronous AWS Lambda calls to decouple the application and data tiers - This is incorrect as it uses asynchronous AWS Lambda calls and the application uses synchronous transactions. The question says there should be no change in the application architecture.
Leverage horizontal scaling for the application's persistence layer by adding Oracle RAC on AWS - The issue is not with the persistence layer at all. This option has only been used as a distractor.
You can deploy scalable Oracle Real Application Clusters (RAC) on Amazon EC2 using Amazon Machine Images (AMI) on AWS Marketplace. Oracle RAC is a shared-everything database cluster technology from Oracle that allows a single database (a set of data files) to be concurrently accessed and served by one or many database server instances.

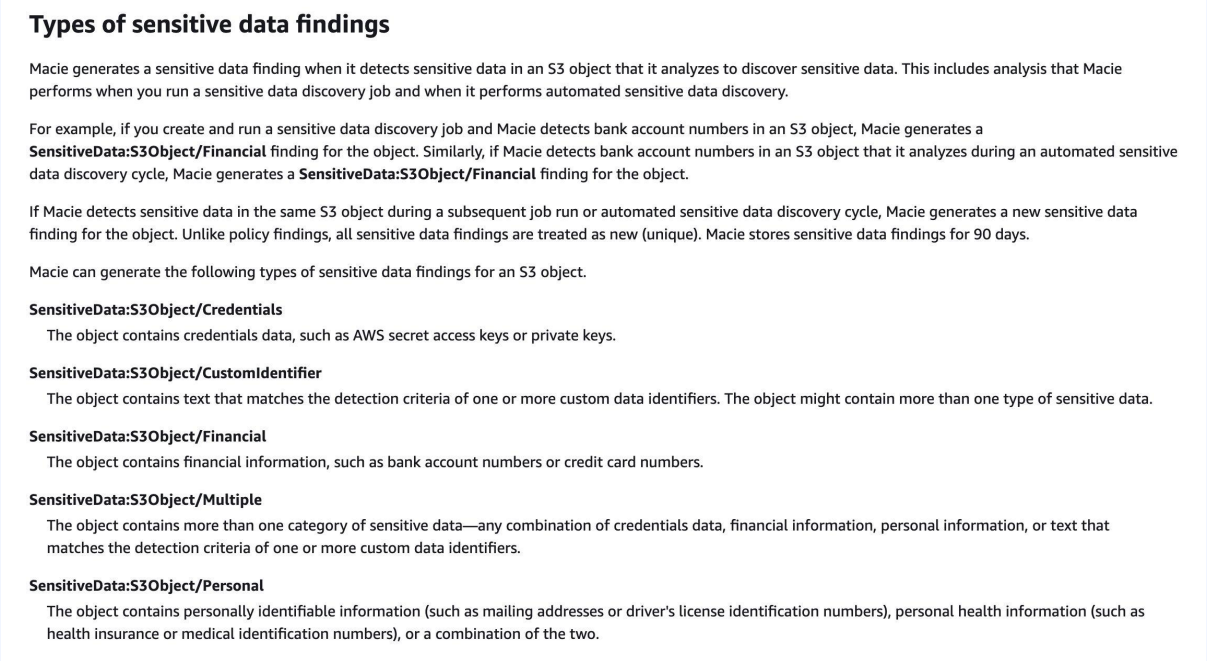
Leverage vertical scaling for the application instance by provisioning a larger Amazon EC2 instance size - Vertical scaling is just a band-aid solution and will not work long term.
References:
https://docs.aws.amazon.com/autoscaling/ec2/userguide/autoscaling-load-balancer.html
https://aws.amazon.com/blogs/compute/operating-lambda-understanding-event-driven-architecture-part-1/

Question 52:
A financial services company uses Amazon S3 to store transformed and anonymized customer data that is generated by a daily batch job. The development team has been tasked to build a solution that analyzes the output of the daily job for any sensitive financial information about the company's customers.
As an AWS Certified Developer Associate, which of the following options would you recommend to address this use case MOST efficiently?

- Configure a S3 event notification for every object upload that triggers a Lambda function based Python script to detect sensitive customer information
- Leverage Macie to analyze the output of the daily batch job and look for any sensitive data findings of type SensitiveData:S3Object/Personal
- Leverage Macie to analyze the output of the daily batch job and look for any sensitive data findings of type SensitiveData:S3Object/CustomIdentifier
- Leverage Macie to analyze the output of the daily batch job and look for any sensitive data findings of type SensitiveData:S3Object/Financial(Correct)

Explanation

Correct option:
Leverage Macie to analyze the output of the daily batch job and look for any sensitive data findings of type SensitiveData:S3Object/Financial
Amazon Macie is a data security service that discovers sensitive data by using machine learning and pattern matching, provides visibility into data security risks, and enables automated protection against those risks. To help you manage the security posture of your organization's Amazon Simple Storage Service (Amazon S3) data estate, Macie provides you with an inventory of your S3 buckets, and automatically evaluates and monitors the buckets for security and access control. If Macie detects a potential issue with the security or privacy of your data, such as a bucket that becomes publicly accessible, Macie generates a finding for you to review and remediate as necessary.
Macie also automates the discovery and reporting of sensitive data to provide you with a better understanding of the data that your organization stores in Amazon S3. To detect sensitive data, you can use built-in criteria and techniques that Macie provides, custom criteria that you define, or a combination of the two. If Macie detects sensitive data in an S3 object, Macie generates a finding to notify you of the sensitive data that Macie found.
Macie generates a sensitive data finding when it detects sensitive data in an S3 object that it analyzes to discover sensitive data. This includes analysis that Macie performs when you run a sensitive data discovery job and when it performs automated sensitive data discovery.
For the given use case, you can use Macie to analyze the output of the daily batch job and look for any sensitive data findings of type SensitiveData:S3Object/Financial which implies that the S3 object contains financial information, such as bank account numbers or credit card numbers.



Types of sensitive data findings

Macie generates a sensitive data finding when it detects sensitive data in an S3 object that it analyzes to discover sensitive data. This includes analysis that Macie performs when you run a sensitive data discovery job and when it performs automated sensitive data discovery.

For example, if you create and run a sensitive data discovery job and Macie detects bank account numbers in an S3 object, Macie generates a **SensitiveData:S3Object/Financial** finding for the object. Similarly, if Macie detects bank account numbers in an S3 object that it analyzes during an automated sensitive data discovery cycle, Macie generates a **SensitiveData:S3Object/Financial** finding for the object.

If Macie detects sensitive data in the same S3 object during a subsequent job run or automated sensitive data discovery cycle, Macie generates a new sensitive data finding for the object. Unlike policy findings, all sensitive data findings are treated as new (unique). Macie stores sensitive data findings for 90 days.

Macie can generate the following types of sensitive data findings for an S3 object.

SensitiveData:S3Object/Credentials
The object contains credentials data, such as AWS secret access keys or private keys.

SensitiveData:S3Object/CustomIdentifier
The object contains text that matches the detection criteria of one or more custom data identifiers. The object might contain more than one type of sensitive data.

SensitiveData:S3Object/Financial
The object contains financial information, such as bank account numbers or credit card numbers.

SensitiveData:S3Object/Multiple
The object contains more than one category of sensitive data—any combination of credentials data, financial information, personal information, or text that matches the detection criteria of one or more custom data identifiers.

SensitiveData:S3Object/Personal
The object contains personally identifiable information (such as mailing addresses or driver's license identification numbers), personal health information (such as health insurance or medical identification numbers), or a combination of the two.

via - https://docs.aws.amazon.com/macie/latest/user/findings-types.html

Incorrect options:

Leverage Macie to analyze the output of the daily batch job and look for any sensitive data findings of type SensitiveData:S3Object/Personal - The object contains personally identifiable information (such as mailing addresses or driver's license identification numbers), personal health information (such as health insurance or medical identification numbers), or a combination of the two.

Leverage Macie to analyze the output of the daily batch job and look for any sensitive data findings of type SensitiveData:S3Object/CustomIdentifier - The object contains text that matches the detection criteria of one or more custom data identifiers. The object might contain more than one type of sensitive data.

As explained above, for the given use case, you need to look for any sensitive data findings of type SensitiveData:S3Object/Financial. So both these options are incorrect.

Configure a S3 event notification for every object upload that triggers a Lambda function based custom application to detect sensitive customer information - Maintaining a custom application to detect sensitive information would be highly inefficient as you need to consistently upgrade the application to handle new formats and types of financial information. This is better handled by leveraging Macie's Findings.

References:

https://docs.aws.amazon.com/macie/latest/user/what-is-macie.html

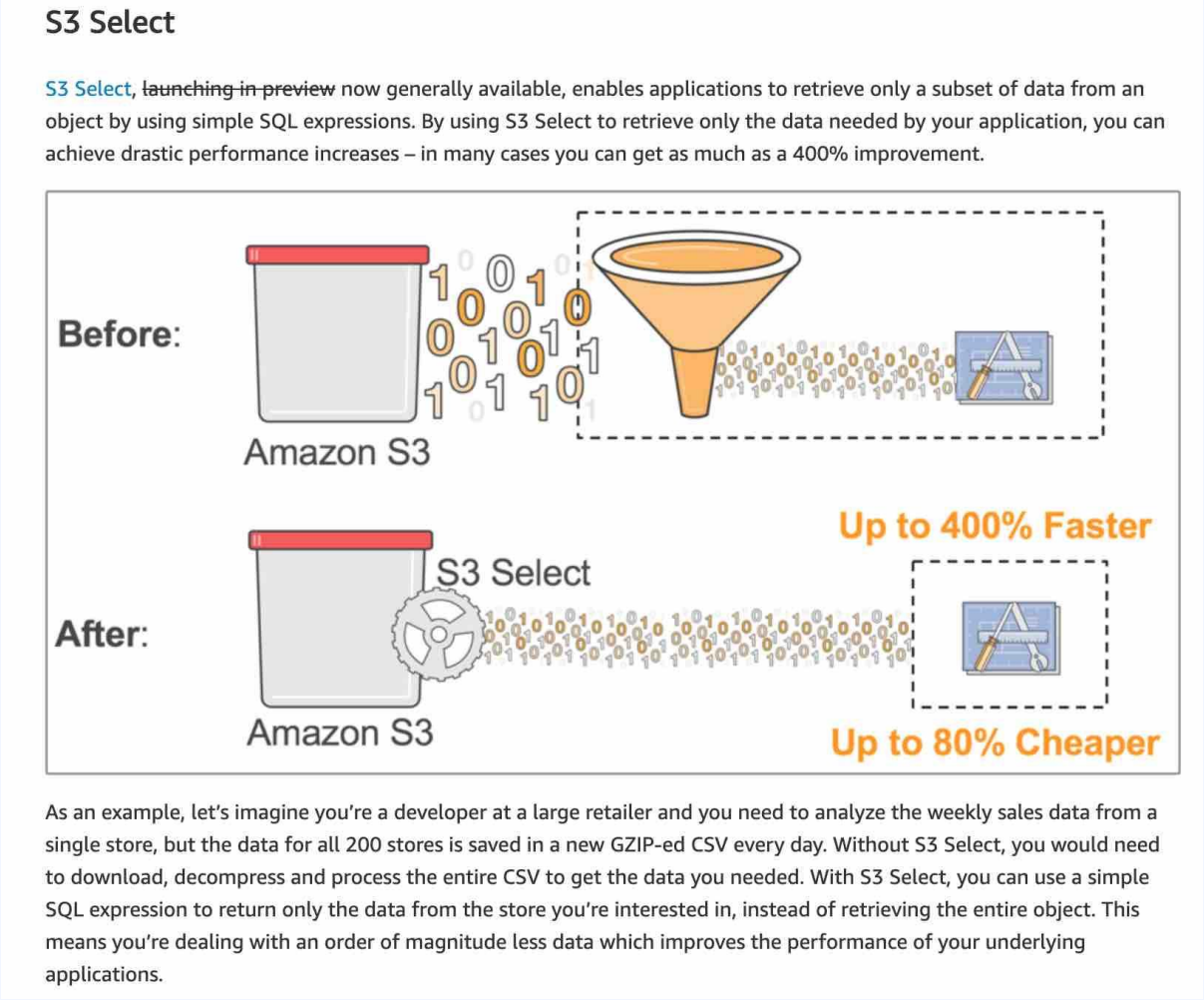
https://docs.aws.amazon.com/macie/latest/user/findings-types.html

Question 53:
You would like to retrieve a subset of your dataset stored in S3 with the CSV format. You would like to retrieve a month of data and only 3 columns out of the 10.
You need to minimize compute and network costs for this, what should you use?

- S3 Select(Correct)
- S3 Access Logs
- S3 Analytics
- S3 Inventory

Explanation

Correct option:
S3 Select
S3 Select enables applications to retrieve only a subset of data from an object by using simple SQL expressions. By using S3 Select to retrieve only the data needed by your application, you can achieve drastic performance increases in many cases you can get as much as a 400% improvement.



via - https://aws.amazon.com/blogs/aws/s3-glacier-select/
Incorrect options:
S3 Inventory - Amazon S3 inventory is one of the tools Amazon S3 provides to help manage your storage. You can use it to audit and report on the replication and encryption status of your objects for business, compliance, and regulatory needs.

S3 Analytics - By using Amazon S3 analytics storage class analysis you can analyze storage access patterns to help you decide when to transition the right data to the right storage class. This new Amazon S3 analytics feature observes data access patterns to help you determine when to transition less frequently accessed STANDARD storage to the STANDARD_IA (IA, for infrequent access) storage class.
S3 Access Logs - Server access logging provides detailed records for the requests that are made to a bucket. Server access logs are useful for many applications. For example, access log information can be useful in security and access audits. It can also help you learn about your customer base and understand your Amazon S3 bill.
Reference:
https://aws.amazon.com/blogs/aws/s3-glacier-select/

Question 54:
An EC2 instance has an IAM instance role attached to it, providing it read and write access to the S3 bucket 'my_bucket'. You have tested the IAM instance role and both reads and writes are working. You then remove the IAM role from the EC2 instance and test both read and write again. Writes stopped working but reads are still working.
What is the likely cause of this behavior?

- The EC2 instance is using cached temporary IAM credentials
- Removing an instance role from an EC2 instance can take a few minutes before being active
- The S3 bucket policy authorizes reads(Correct)
- When a read is done on a bucket, there's a grace period of 5 minutes to do the same read again

Explanation

Correct option:
The S3 bucket policy authorizes reads
When evaluating an IAM policy of an EC2 instance doing actions on S3, the least-privilege union of both the IAM policy of the EC2 instance and the bucket policy of the S3 bucket are taken into account.
For the given use-case, as IAM role has been removed, therefore only the S3 bucket policy comes into effect which authorizes reads.
Here is a great reference blog for understanding the various scenarios for using IAM policy vs S3 bucket policy -
https://aws.amazon.com/blogs/security/iam-policies-and-bucket-policies-and-acls-oh-my-controlling-access-to-s3-resources/
Incorrect options:
The EC2 instance is using cached temporary IAM credentials - As the IAM instance role has been removed that wouldn't be the case
Removing an instance role from an EC2 instance can take a few minutes before being active - It is immediately active and even if it wasn't, it wouldn't make sense as we can still do reads but not writes.
When a read is done on a bucket, there's a grace period of 5 minutes to do the same read again - This is not true. Every single request is evaluated against IAM in the AWS model.
Reference:
https://aws.amazon.com/blogs/security/iam-policies-and-bucket-policies-and-acls-oh-my-controlling-access-to-s3-resources/
Question 55:
An e-commerce company has multiple EC2 instances operating in a private subnet which is part of a custom VPC. These instances are running an image processing application that needs to access images stored on S3. Once each image is processed, the status of the corresponding record needs to be marked as completed in a DynamoDB table.
How would you go about providing private access to these AWS resources which are not part of this custom VPC?

- Create a gateway endpoint for DynamoDB and add it as a target in the route table of the custom VPC. Create an API endpoint for S3 and then connect to the S3 service using the private IP address

- Create a separate interface endpoint for S3 and DynamoDB each. Then connect to these services using the private IP address
- Create a separate gateway endpoint for S3 and DynamoDB each. Add two new target entries for these two gateway endpoints in the route table of the custom VPC
- (Correct)
- Create a gateway endpoint for S3 and add it as a target in the route table of the custom VPC. Create an interface endpoint for DynamoDB and then connect to the DynamoDB service using the private IP address

Explanation

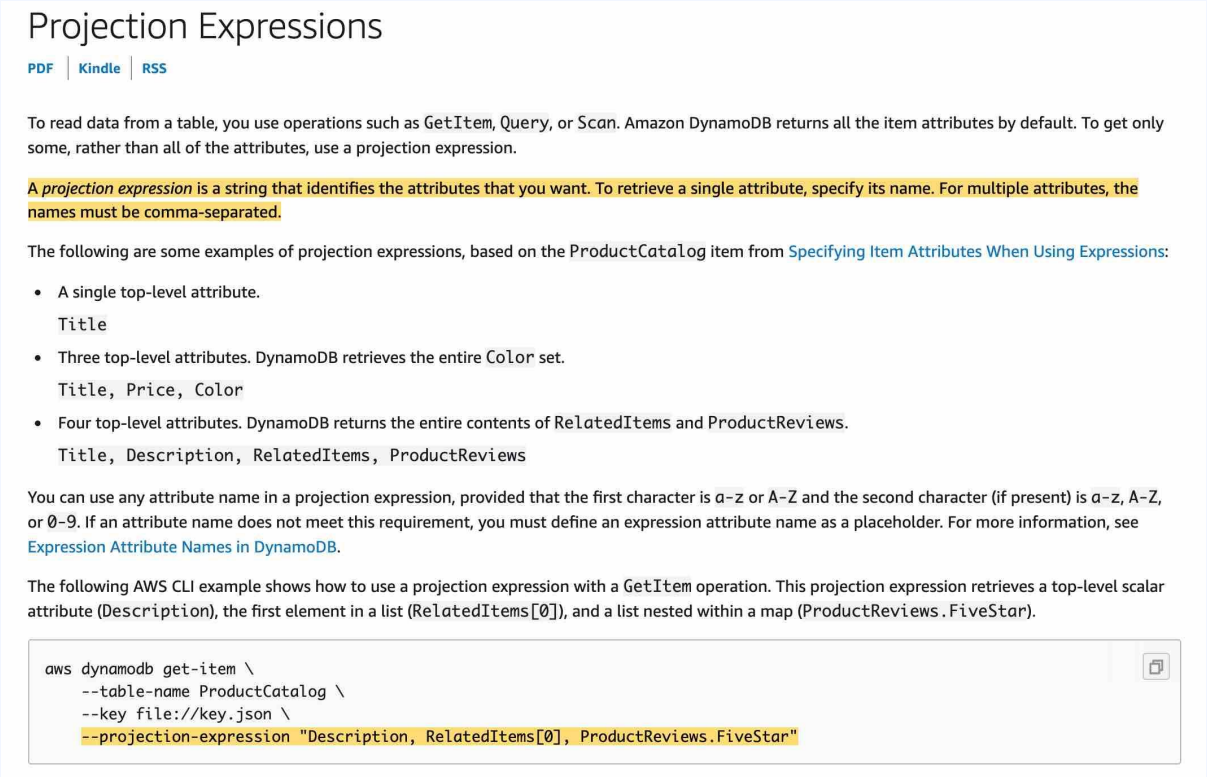
Correct option:
Create a separate gateway endpoint for S3 and DynamoDB each. Add two new target entries for these two gateway endpoints in the route table of the custom VPC
Endpoints are virtual devices. They are horizontally scaled, redundant, and highly available VPC components. They allow communication between instances in your VPC and services without imposing availability risks or bandwidth constraints on your network traffic.
A VPC endpoint enables you to privately connect your VPC to supported AWS services and VPC endpoint services powered by AWS PrivateLink without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. Instances in your VPC do not require public IP addresses to communicate with resources in the service. Traffic between your VPC and the other service does not leave the Amazon network.
There are two types of VPC endpoints: interface endpoints and gateway endpoints. An interface endpoint is an elastic network interface with a private IP address from the IP address range of your subnet that serves as an entry point for traffic destined to a supported service.
A gateway endpoint is a gateway that you specify as a target for a route in your route table for traffic destined to a supported AWS service. The following AWS services are supported:
Amazon S3
DynamoDB
You should note that S3 now supports both gateway endpoints as well as the interface endpoints.
Incorrect options:
Create a gateway endpoint for S3 and add it as a target in the route table of the custom VPC. Create an interface endpoint for DynamoDB and then connect to the DynamoDB service using the private IP address
Create a separate interface endpoint for S3 and DynamoDB each. Then connect to these services using the private IP address
DynamoDB does not support interface endpoints, so these two options are incorrect.
Create a gateway endpoint for DynamoDB and add it as a target in the route table of the custom VPC. Create an API endpoint for S3 and then connect to the S3 service using the private IP address - There is no such thing as an API endpoint for S3. API endpoints are used with AWS API Gateway. This option has been added as a distractor.
References:
https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html
https://docs.aws.amazon.com/AmazonS3/latest/userguide/privatelink-interface-endpoints.html

Question 56:
Which of the following CLI options will allow you to retrieve a subset of the attributes coming from a DynamoDB scan?

- --page-size
- --filter-expression
- --max-items
- --projection-expression(Correct)

Explanation

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



Projection Expressions

PDF | Kindle | RSS

To read data from a table, you use operations such as `GetItem`, `Query`, or `Scan`. Amazon DynamoDB returns all the item attributes by default. To get only some, rather than all of the attributes, use a projection expression.

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

The following are some examples of projection expressions, based on the `ProductCatalog` item from Specifying Item Attributes When Using Expressions:

- A single top-level attribute.
  `Title`
- Three top-level attributes. DynamoDB retrieves the entire `Color` set.
  `Title, Price, Color`
- Four top-level attributes. DynamoDB returns the entire contents of `RelatedItems` and `ProductReviews`.
  `Title, Description, RelatedItems, ProductReviews`

You can use any attribute name in a projection expression, provided that the first character is `a-z` or `A-Z` and the second character (if present) is `a-z`, `A-Z`, or `0-9`. If an attribute name does not meet this requirement, you must define an expression attribute name as a placeholder. For more information, see Expression Attribute Names in DynamoDB.

The following AWS CLI example shows how to use a projection expression with a `GetItem` operation. This projection expression retrieves a top-level scalar attribute (`Description`), the first element in a list (`RelatedItems[0]`), and a list nested within a map (`ProductReviews.FiveStar`).

```
aws dynamodb get-item \
    --table-name ProductCatalog \
    --key file://key.json \
    --projection-expression "Description, RelatedItems[0], ProductReviews.FiveStar"
```

via - https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Expressions.ProjectionExpressions.html
To read data from a table, you use operations such as GetItem, Query, or Scan. DynamoDB returns all of the item attributes by default. To get just some, rather than all of the attributes, use a projection expression.
Incorrect options:
--filter-expression - If you need to further refine the Query results, you can optionally provide a filter expression. A filter expression determines which items within the Query results should be returned to you. All of the other results are discarded. A filter expression is applied after Query finishes, but before the results are returned. Therefore, a Query will consume the same amount of read capacity, regardless of whether a filter expression is present.

--page-size - You can use the --page-size option to specify that the AWS CLI requests a smaller number of items from each call to the AWS service. The CLI still retrieves the full list but performs a larger number of service API calls in the background and retrieves a smaller number of items with each call.
--max-items - To include fewer items at a time in the AWS CLI output, use the --max-items option. The AWS CLI still handles pagination with the service as described above, but prints out only the number of items at a time that you specify.
Reference:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Expressions.ProjectionExpressions.html

Question 57:
Your organization has a single Amazon Simple Storage Service (S3) bucket that contains folders labeled with customer names. Several administrators have IAM access to the S3 bucket and versioning is enabled to easily recover from unintended user actions.
Which of the following statements about versioning is NOT true based on this scenario?

- Overwriting a file increases its versions
- Any file that was unversioned before enabling versioning will have the 'null' version
- Versioning can be enabled only for a specific folder(Correct)
- Deleting a file is a recoverable operation

Explanation

Correct option:
Versioning can be enabled only for a specific folder
The versioning state applies to all (never some) of the objects in that bucket. The first time you enable a bucket for versioning, objects in it are thereafter always versioned and given a unique version ID.
Versioning Overview:

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. When you enable versioning for a bucket, if Amazon S3 receives multiple write requests for the same object simultaneously, it stores all of the objects.

If you enable versioning for a bucket, Amazon S3 automatically generates a unique version ID for the object being stored. In one bucket, for example, you can have two objects with the same key, but different version IDs, such as `photo.gif` (version 111111) and `photo.gif` (version 121212).

Versioning-enabled buckets enable you to recover objects from accidental deletion or overwrite. For example:

- If you delete an object, instead of removing it permanently, Amazon S3 inserts a delete marker, which becomes the current object version. You can always restore the previous version. For more information, see Deleting object versions.
- If you overwrite an object, it results in a new object version in the bucket. You can always restore the previous version.

Important
If you have an object expiration lifecycle policy in your non-versioned bucket and you want to maintain the same permanent delete behavior when you enable versioning, you must add a noncurrent expiration policy. The noncurrent expiration lifecycle policy will manage the deletes of the noncurrent object versions in the version-enabled bucket. (A version-enabled bucket maintains one current and zero or more noncurrent object versions.) For more information, see How Do I Create a Lifecycle Policy for an S3 Bucket? in the *Amazon Simple Storage Service Console User Guide*.

Buckets can be in one of three states: unversioned (the default), versioning-enabled, or versioning-suspended.

Important
Once you version-enable a bucket, it can never return to an unversioned state. You can, however, suspend versioning on that bucket.

The versioning state applies to all (never some) of the objects in that bucket. The first time you enable a bucket for versioning, objects in it are thereafter always versioned and given a unique version ID. Note the following:

- Objects stored in your bucket before you set the versioning state have a version ID of `null`. When you enable versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles the objects in future requests. For more information, see Managing objects in a versioning-enabled bucket.
- The bucket owner (or any user with appropriate permissions) can suspend versioning to stop accruing object versions. When you suspend versioning, existing objects in your bucket do not change. What changes is how Amazon S3 handles objects in future requests. For more information, see Managing objects in a versioning-suspended bucket.

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

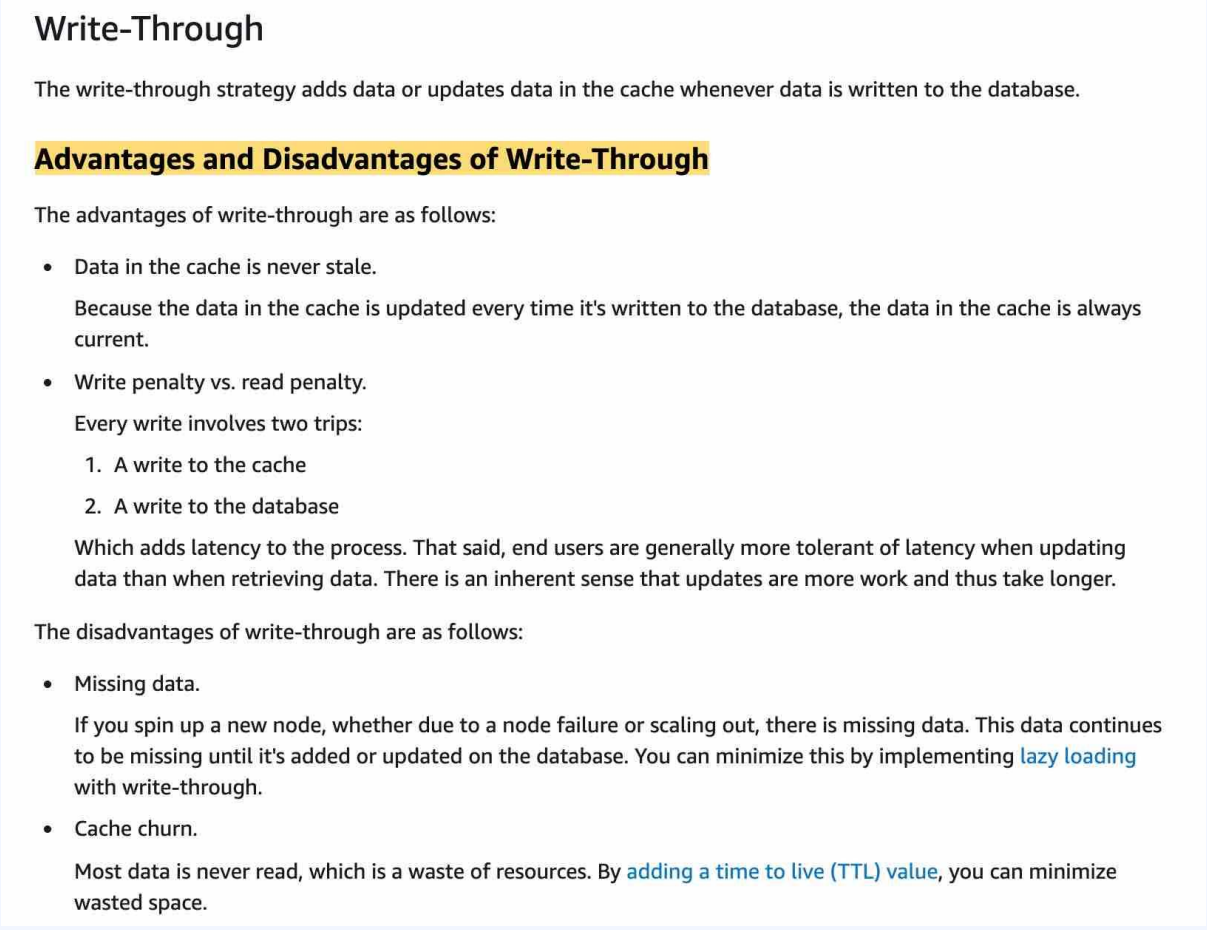
Overwriting a file increases its versions - If you overwrite an object (file), it results in a new object version in the bucket. You can always restore the previous version.
Deleting a file is a recoverable operation - Correct, when you delete an object (file), Amazon S3 inserts a delete marker, which becomes the current object version and you can restore the previous version.
Any file that was unversioned before enabling versioning will have the 'null' version - Objects stored in your bucket before you set the versioning state have a version ID of null. Those existing objects in your bucket do not change.
Reference:
https://docs.aws.amazon.com/AmazonS3/latest/dev/Versioning.html

Question 58:
You are running a web application where users can author blogs and share them with their followers. Most of the workflow is read based, but when a blog is updated, you would like to ensure that the latest data is served to the users (no stale data). The Developer has already suggested using ElastiCache to cope with the read load but has asked you to implement a caching strategy that complies with the requirements of the site.
Which strategy would you recommend?

- Use a Write Through strategy(Correct)
- Use DAX
- Use a Lazy Loading strategy with TTL
- Use a Lazy Loading strategy without TTL

Explanation

Correct option:
Use a Write Through strategy
The write-through strategy adds data or updates data in the cache whenever data is written to the database.
In a Write Through strategy, any new blog or update to the blog will be written to both the database layer and the caching layer, thus ensuring that the latest data is always served from the cache.



Write-Through

The write-through strategy adds data or updates data in the cache whenever data is written to the database.

Advantages and Disadvantages of Write-Through

The advantages of write-through are as follows:

- Data in the cache is never stale.

  Because the data in the cache is updated every time it's written to the database, the data in the cache is always current.

- Write penalty vs. read penalty.

  Every write involves two trips:

  1. A write to the cache
  2. A write to the database

  Which adds latency to the process. That said, end users are generally more tolerant of latency when updating data than when retrieving data. There is an inherent sense that updates are more work and thus take longer.

The disadvantages of write-through are as follows:

- Missing data.

  If you spin up a new node, whether due to a node failure or scaling out, there is missing data. This data continues to be missing until it's added or updated on the database. You can minimize this by implementing lazy loading with write-through.

- Cache churn.

  Most data is never read, which is a waste of resources. By adding a time to live (TTL) value, you can minimize wasted space.

via - https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/Strategies.html
Incorrect options:
Use a Lazy Loading strategy without TTL

Lazy Loading is a caching strategy that loads data into the cache only when necessary. Whenever your application requests data, it first requests the ElastiCache cache. If the data exists in the cache and is current, ElastiCache returns the data to your application. If the data doesn't exist in the cache or has expired, your application requests the data from your data store.

Use a Lazy Loading strategy with TTL

In the case of Lazy Loading, the data is loaded onto the cache whenever the data is missing from the cache. In case the blog gets updated, it won't be updated from the cache unless that cache expires (in case you used a TTL). Time to live (TTL) is an integer value that specifies the number of seconds until the key expires. When an application attempts to read an expired key, it is treated as though the key is not found. The database is queried for the key and the cache is updated. Therefore, for a while, old data will be served to users which is a problem from a requirements perspective as we don't want any stale data.

Use DAX - This is a cache for DynamoDB based implementations, but in this question, we are considering ElastiCache. Therefore this option is not relevant.

Reference:

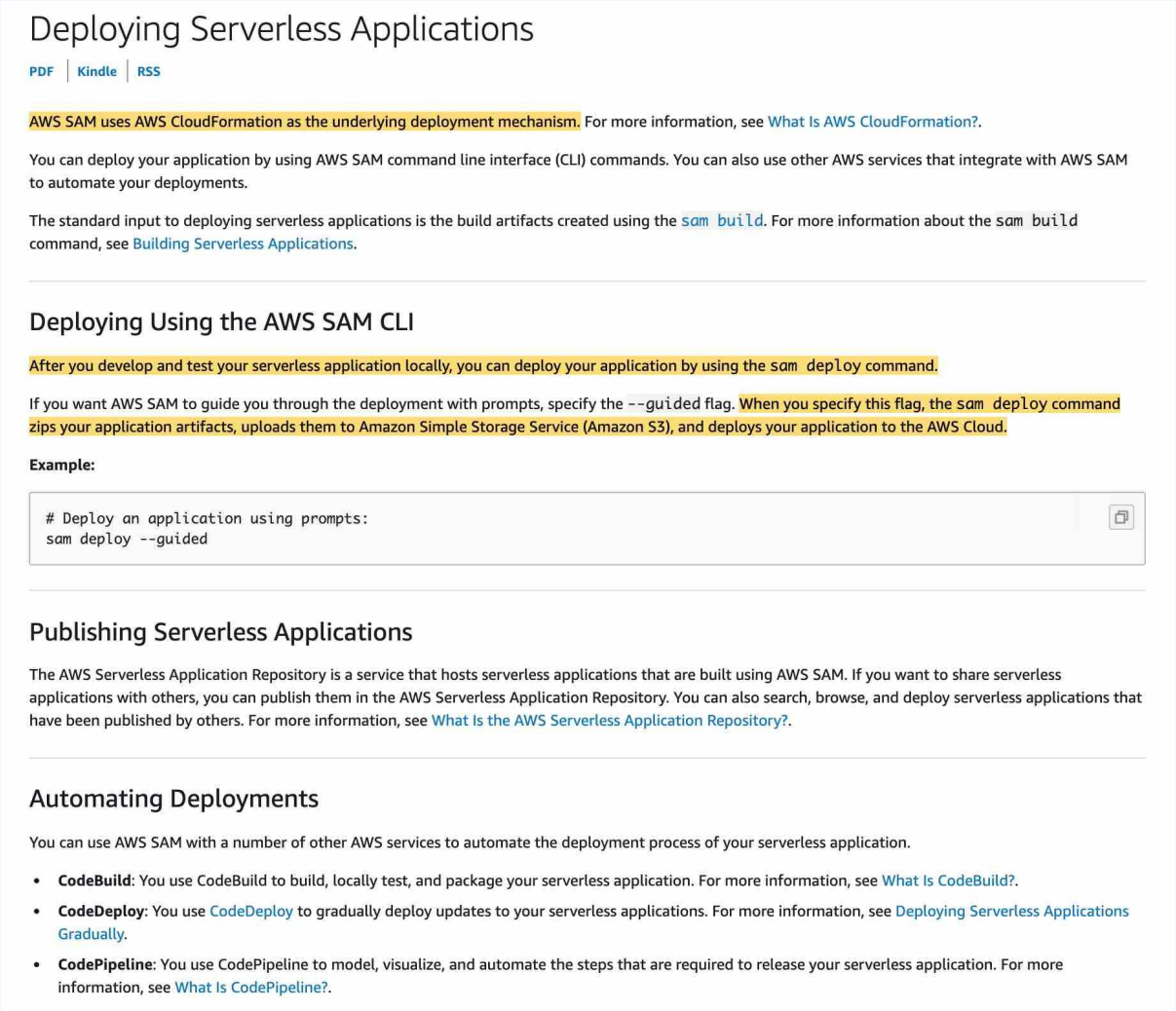
https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/Strategies.html

Question 59:
A media analytics company has built a streaming application on Lambda using Serverless Application Model (SAM).
As a Developer Associate, which of the following would you identify as the correct order of execution to successfully deploy the application?

- Develop the SAM template locally => upload the template to Lambda => deploy your application to the cloud
- Develop the SAM template locally => upload the template to CodeCommit => deploy your application to CodeDeploy
- Develop the SAM template locally => deploy the template to S3 => use your application in the cloud
- Develop the SAM template locally => upload the template to S3 => deploy your application to the cloud(Correct)

Explanation

Correct option:
Develop the SAM template locally => upload the template to S3 => deploy your application to the cloud
The AWS Serverless Application Model (SAM) is an open-source framework for building serverless applications. It provides shorthand syntax to express functions, APIs, databases, and event source mappings. With just a few lines per resource, you can define the application you want and model it using YAML.
You can develop and test your serverless application locally, and then you can deploy your application by using the sam deploy command. The sam deploy command zips your application artifacts, uploads them to Amazon Simple Storage Service (Amazon S3), and deploys your application to the AWS Cloud. AWS SAM uses AWS CloudFormation as the underlying deployment mechanism.



Deploying Serverless Applications

PDF | Kindle | RSS

AWS SAM uses AWS CloudFormation as the underlying deployment mechanism. For more information, see What Is AWS CloudFormation?.

You can deploy your application by using AWS SAM command line interface (CLI) commands. You can also use other AWS services that integrate with AWS SAM to automate your deployments.

The standard input to deploying serverless applications is the build artifacts created using the `sam build`. For more information about the `sam build` command, see Building Serverless Applications.

Deploying Using the AWS SAM CLI

After you develop and test your serverless application locally, you can deploy your application by using the `sam deploy` command.

If you want AWS SAM to guide you through the deployment with prompts, specify the `--guided` flag. When you specify this flag, the `sam deploy` command zips your application artifacts, uploads them to Amazon Simple Storage Service (Amazon S3), and deploys your application to the AWS Cloud.

Example:

```
# Deploy an application using prompts:
sam deploy --guided
```

Publishing Serverless Applications

The AWS Serverless Application Repository is a service that hosts serverless applications that are built using AWS SAM. If you want to share serverless applications with others, you can publish them in the AWS Serverless Application Repository. You can also search, browse, and deploy serverless applications that have been published by others. For more information, see What Is the AWS Serverless Application Repository?.

Automating Deployments

You can use AWS SAM with a number of other AWS services to automate the deployment process of your serverless application.

- **CodeBuild**: You use CodeBuild to build, locally test, and package your serverless application. For more information, see What Is CodeBuild?.
- **CodeDeploy**: You use CodeDeploy to gradually deploy updates to your serverless applications. For more information, see Deploying Serverless Applications Gradually.
- **CodePipeline**: You use CodePipeline to model, visualize, and automate the steps that are required to release your serverless application. For more information, see What Is CodePipeline?.

via - https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-deploying.html

Incorrect options:

Develop the SAM template locally => upload the template to Lambda => deploy your application to the cloud

Develop the SAM template locally => upload the template to CodeCommit => deploy your application to CodeDeploy

Develop the SAM template locally => deploy the template to S3 => use your application in the cloud

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

Reference:

https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-deploying.html

Question 60:
When your company first created an AWS account, you began with a single sign-in principal called a root user account that had complete access to all AWS services and resources.
What should you do to adhere to best practices for using the root user account?

- It should be accessible using the access key id and secret access key
- It should be accessible by no one, throw away the passwords after creating the account
- It should be accessible by 3 to 6 members of the IT team
- It should be accessible by one admin only after enabling Multi-factor authentication(Correct)

Explanation

Correct option:
It should be accessible by one admin only after enabling Multi-factor authentication

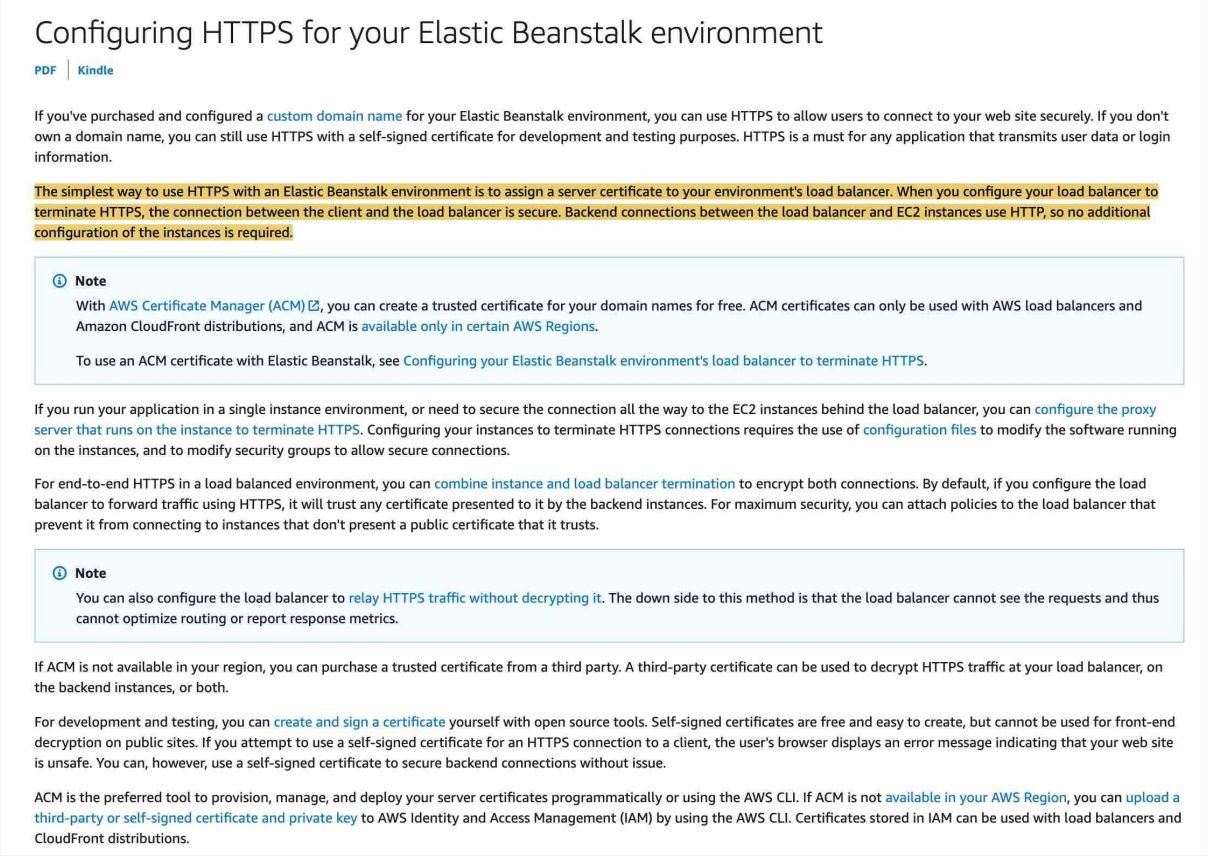
AWS Root Account Security Best Practices:  via - https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#lock-away-credentials
If you continue to use the root user credentials, we recommend that you follow the security best practice to enable multi-factor authentication (MFA) for your account. Because your root user can perform sensitive operations in your account, adding a layer of authentication helps you to better secure your account. Multiple types of MFA are available.
Incorrect options:
It should be accessible by 3 to 6 members of the IT team - Only the owner of the AWS account should have access to the root account credentials. You should create an IT group with admin permissions via IAM and then assign a few users to this group.
It should be accessible using the access key id and secret access key - AWS recommends that you should not use the access key id and secret access key for the AWS account root user.
It should be accessible by no one, throw away the passwords after creating the account - You will still need to store the password somewhere for your root account.
Reference:
https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html#lock-away-credentials

Question 61:
You would like your Elastic Beanstalk environment to expose an HTTPS endpoint instead of an HTTP endpoint to get in-flight encryption between your clients and your web servers.
What must be done to set up HTTPS on Beanstalk?

- Configure Health Checks
- Use a separate CloudFormation template to load the SSL certificate onto the Load Balancer
- Create a config file in the .ebextensions folder to configure the Load Balancer(Correct)
- Open up the port 80 for the security group

Explanation

Correct option:
The simplest way to use HTTPS with an Elastic Beanstalk environment is to assign a server certificate to your environment's load balancer. When you configure your load balancer to terminate HTTPS, the connection between the client and the load balancer is secure. Backend connections between the load balancer and EC2 instances use HTTP, so no additional configuration of the instances is required.



Configuring HTTPS for your Elastic Beanstalk environment

PDF | Kindle

If you've purchased and configured a custom domain name for your Elastic Beanstalk environment, you can use HTTPS to allow users to connect to your web site securely. If you don't own a domain name, you can still use HTTPS with a self-signed certificate for development and testing purposes. HTTPS is a must for any application that transmits user data or login information.

The simplest way to use HTTPS with an Elastic Beanstalk environment is to assign a server certificate to your environment's load balancer. When you configure your load balancer to terminate HTTPS, the connection between the client and the load balancer is secure. Backend connections between the load balancer and EC2 instances use HTTP, so no additional configuration of the instances is required.

Note

With AWS Certificate Manager (ACM), you can create a trusted certificate for your domain names for free. ACM certificates can only be used with AWS load balancers and Amazon CloudFront distributions, and ACM is available only in certain AWS Regions.

To use an ACM certificate with Elastic Beanstalk, see Configuring your Elastic Beanstalk environment's load balancer to terminate HTTPS.

If you run your application in a single instance environment, or need to secure the connection all the way to the EC2 instances behind the load balancer, you can configure the proxy server that runs on the instance to terminate HTTPS. Configuring your instances to terminate HTTPS connections requires the use of configuration files to modify the software running on the instances, and to modify security groups to allow secure connections.

For end-to-end HTTPS in a load balanced environment, you can combine instance and load balancer termination to encrypt both connections. By default, if you configure the load balancer to forward traffic using HTTPS, it will trust any certificate presented to it by the backend instances. For maximum security, you can attach policies to the load balancer that prevent it from connecting to instances that don't present a public certificate that it trusts.

Note

You can also configure the load balancer to relay HTTPS traffic without decrypting it. The down side to this method is that the load balancer cannot see the requests and thus cannot optimize routing or report response metrics.

If ACM is not available in your region, you can purchase a trusted certificate from a third party. A third-party certificate can be used to decrypt HTTPS traffic at your load balancer, on the backend instances, or both.

For development and testing, you can create and sign a certificate yourself with open source tools. Self-signed certificates are free and easy to create, but cannot be used for front-end decryption on public sites. If you attempt to use a self-signed certificate for an HTTPS connection to a client, the user's browser displays an error message indicating that your web site is unsafe. You can, however, use a self-signed certificate to secure backend connections without issue.

ACM is the preferred tool to provision, manage, and deploy your server certificates programmatically or using the AWS CLI. If ACM is not available in your AWS Region, you can upload a third-party or self-signed certificate and private key to AWS Identity and Access Management (IAM) by using the AWS CLI. Certificates stored in IAM can be used with load balancers and CloudFront distributions.

via - https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/configuring-https.html
Create a config file in the .ebextensions folder to configure the Load Balancer
To update your AWS Elastic Beanstalk environment to use HTTPS, you need to configure an HTTPS listener for the load balancer in your environment. Two types of load balancers support an HTTPS listener: Classic Load Balancer and Application Load Balancer.
Example .ebextensions/securelistener-alb.config

Use this example when your environment has an Application Load Balancer. The example uses options in the aws:elbv2:listener namespace to configure an HTTPS listener on port 443 with the specified certificate. The listener routes traffic to the default process.

```
option_settings:
  aws:elbv2:listener:443:
    ListenerEnabled: 'true'
    Protocol: HTTPS
    SSLCertificateArns:
arn:aws:acm:us-east-2:1234567890123:certificate/####################################
```

Incorrect options:

Use a separate CloudFormation template to load the SSL certificate onto the Load Balancer - A separate CloudFormation template won't be able to mutate the state of a Load Balancer managed by Elastic Beanstalk, so this option is incorrect.

Open up the port 80 for the security group - Port 80 is for HTTP traffic, so this option is incorrect.

Configure Health Checks - Health Checks are not related to SSL certificates, so this option is ruled out.

References:

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/configuring-https-elb.html

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/configuring-https.html

Question 62:
A business-critical application is hosted on an Amazon EC2 instance and the latest update is in the testing phase. The business has requested a solution to track the average response time of the application and send a notification to the manager if it exceeds a particular threshold. The update will eventually be implemented in the production environment where the solution will be deployed on multiple EC2 instances.
Which of the following options would you combine to address the requirements of the business? (Select two)

- Configure the application to write the response time to a log file on the EC2 instance. Install and configure the Amazon CloudWatch agent on the EC2 instance to stream the application logs to CloudWatch Logs. Create a metric filter for the response time from the log file(Correct)
- Configure an EventBridge custom rule to send an Amazon Simple Notification Service (Amazon SNS) notification when the average of the response time metric exceeds the threshold
- Install and configure AWS Systems Manager Agent (SSM Agent) on the EC2 instances to monitor the response time and send the data to Amazon CloudWatch as a custom metric
- Configure the application to write the response time to a log file on the instance. Install and configure the Amazon Inspector agent on the EC2 instances to read the logs and send the response time to Amazon EventBridge
- Create a CloudWatch alarm to send an Amazon Simple Notification Service (Amazon SNS) notification when the average of the response time metric exceeds the threshold(Correct)

Explanation

Correct options:
Configure the application to write the response time to a log file on the EC2 instance. Install and configure the Amazon CloudWatch agent on the EC2 instance to stream the application logs to CloudWatch Logs. Create a metric filter for the response time from the log file
Create a CloudWatch alarm to send an Amazon Simple Notification Service (Amazon SNS) notification when the average of the response time metric exceeds the threshold
You can collect metrics and logs from Amazon EC2 instances and on-premises servers with the CloudWatch agent. The unified CloudWatch agent enables you to do the following:

1. Collect internal system-level metrics from Amazon EC2 instances across operating systems. The metrics can include in-guest metrics, in addition to the metrics for EC2 instances.
2. Collect system-level metrics from on-premises servers. These can include servers in a hybrid environment as well as servers not managed by AWS.
3. Retrieve custom metrics from your applications or services using the StatsD and collectd protocols. StatsD is supported on both Linux servers and servers running Windows Server. collectd is supported only on Linux servers.
4. Collect logs from Amazon EC2 instances and on-premises servers, running either Linux or Windows Server.

You can store and view the metrics that you collect with the CloudWatch agent in CloudWatch just as you can with any other CloudWatch metrics. The default namespace for metrics collected by the CloudWatch agent is CWAgent, although you can specify a different namespace when you configure the agent.

A metric alarm watches a single CloudWatch metric or the result of a math expression based on CloudWatch metrics. The alarm performs one or more actions based on the value of the metric or expression relative to a threshold over a number of time periods. The action can be sending a notification to an Amazon SNS topic, performing an Amazon EC2 action or an Amazon EC2 Auto Scaling action, or creating an OpsItem or incident in Systems Manager. For this use case, we will configure the metric alarm action to send an SNS notification when the alarm is in an ALARM state.
Incorrect options:
Configure the application to write the response time to a log file on the instance. Install and configure the Amazon Inspector agent on the EC2 instances to read the logs and send the response time to Amazon EventBridge - This statement is incorrect. Amazon Inspector is an automated vulnerability management service that continually scans Amazon Elastic Compute Cloud (EC2), AWS Lambda functions, and container workloads for software vulnerabilities and unintended network exposure. Inspector is not a log-collecting agent.
Configure an EventBridge custom rule to send an Amazon Simple Notification Service (Amazon SNS) notification when the average of the response time metric exceeds the threshold - Amazon EventBridge integrates with Amazon CloudWatch so that when CloudWatch alarms are triggered, a matching EventBridge rule can execute targets. So we are using CloudWatch alarms in this scenario too and hence a straightforward way would be to use CloudWatch alarm.
Install and configure AWS Systems Manager Agent (SSM Agent) on the EC2 instances to monitor the response time and send the data to Amazon CloudWatch as a custom metric - This statement is incorrect. AWS Systems Manager Agent (SSM Agent) is Amazon software that runs on Amazon Elastic Compute Cloud (Amazon EC2) instances, edge devices, on-premises servers, and virtual machines (VMs). SSM Agent makes it possible for the Systems Manager to update, manage, and configure these resources. SSM agent is required on each instance that needs to be managed from the Systems Manager service. SSM agents cannot collect log data and push it to CloudWatch logs. However, Systems Manager can be used to automate the task of CloudWatch agent installation on the instances.
References:
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/Install-CloudWatch-Agent.html
https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/AlarmThatSendsEmail.html
Question 63:
The customer feedback functionality for a company's flagship web application is handled via an Amazon API Gateway based REST API that invokes an AWS Lambda function for further processing. Although the performance of the function is satisfactory, the development team has been tasked to optimize the startup time of the Lambda function to further improve the customer experience.
How will you optimize the Lambda function for faster initialization?

- Configure provisioned concurrency for the Lambda function to respond immediately to the function's invocations(Correct)
- Configure an interface VPC endpoint powered by AWS PrivateLink to access the Amazon API Gateway REST API with milliseconds latency
- Enable API caching in Amazon API Gateway to cache AWS Lambda function response
- Configure reserved concurrency to guarantee the maximum number of concurrent instances of the Lambda function

Explanation

Correct option:

Configure provisioned concurrency for the Lambda function to respond immediately to the function's invocations

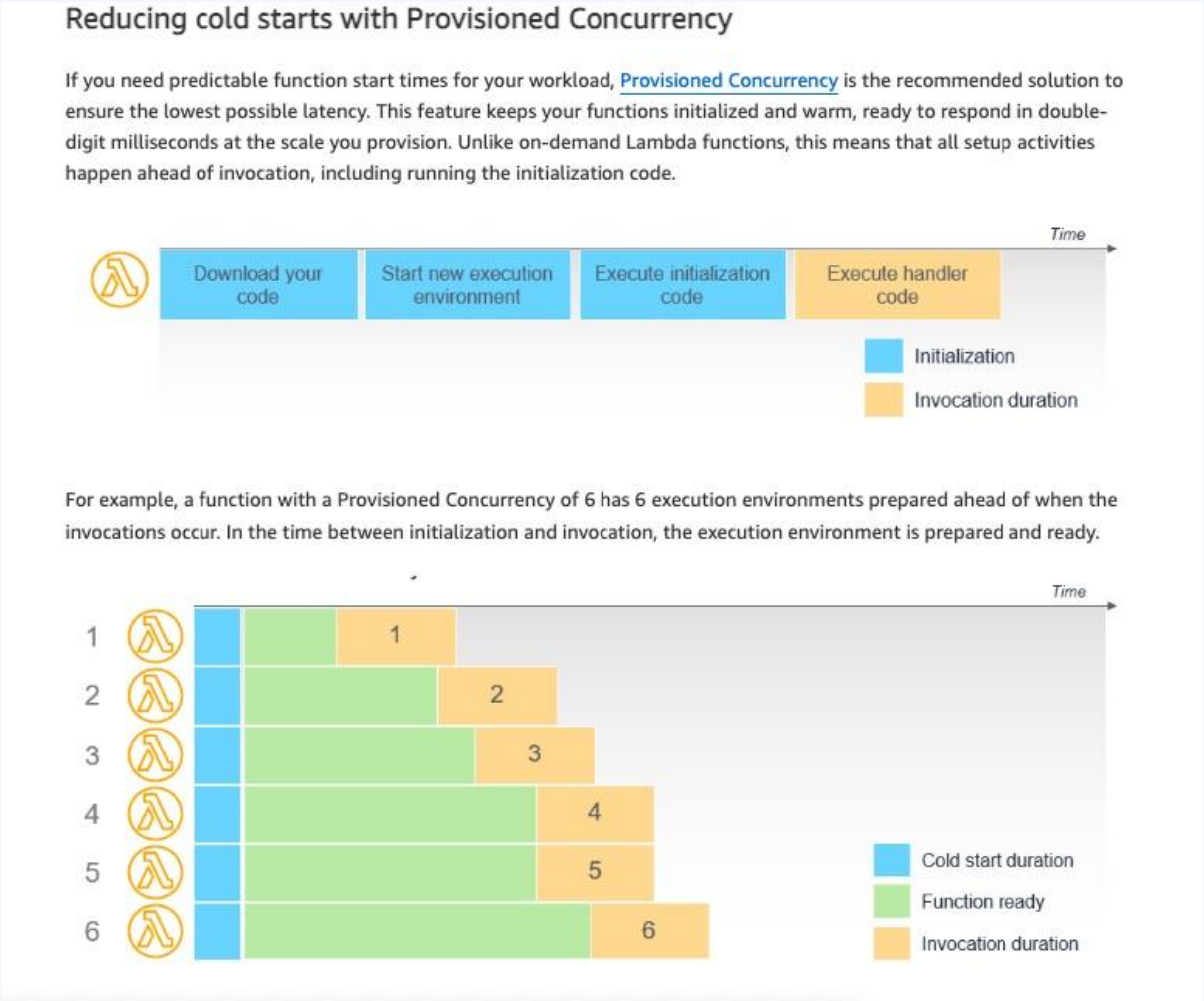
When Lambda allocates an instance of your function, the runtime loads your function's code and runs the initialization code that you define outside of the handler. If your code and dependencies are large, or you create SDK clients during initialization, this process can take some time. When your function has not been used for some time, needs to scale up, or when you update a function, Lambda creates new execution environments. This causes the portion of requests that are served by new instances to have higher latency than the rest, otherwise known as a cold start.

By allocating provisioned concurrency before an increase in invocations, you can ensure that all requests are served by initialized instances with low latency. Lambda functions configured with provisioned concurrency run with consistent start-up latency, making them ideal for building interactive mobile or web backends, latency-sensitive microservices, and synchronously invoked APIs.

Functions with Provisioned Concurrency differ from on-demand functions in some important ways:

1. Initialization code does not need to be optimized. Since this happens long before the invocation, lengthy initialization does not impact the latency of invocations. If you are using runtimes that typically take longer to initialize, like Java, the performance of these can benefit from using Provisioned Concurrency.
2. Initialization code is run more frequently than the total number of invocations. Since Lambda is highly available, for every one unit of Provisioned Concurrency, there are a minimum of two execution environments prepared in separate Availability Zones. This is to ensure that your code is available in the event of a service disruption. As environments are reaped and load balancing occurs, Lambda over-provisions environments to ensure availability. You are not charged for this activity. If your code initializer implements logging, you will see additional log files anytime that this code is run, even though the main handler is not invoked.
3. Provisioned Concurrency cannot be used with the $LATEST version. This feature can only be used with published versions and aliases of a function. If you see cold starts for functions configured to use Provisioned Concurrency, you may be invoking the $LATEST version, instead of the version or alias with Provisioned Concurrency configured.

Reducing cold starts with Provisioned Concurrency:



via - https://aws.amazon.com/blogs/compute/operating-lambda-performance-optimization-part-1/

Incorrect options:

Configure reserved concurrency to guarantee the maximum number of concurrent instances of the Lambda function - Reserved concurrency guarantees the maximum number of concurrent instances for the function. When a function has reserved concurrency, no other function can use that concurrency. There is no charge for configuring reserved concurrency for a function. Whereas, provisioned concurrency initializes a requested number of execution environments so that they are prepared to respond immediately to your function's invocations.

Configure an interface VPC endpoint powered by AWS PrivateLink to access the Amazon API Gateway REST API with milliseconds latency - An interface VPC endpoint can be used to connect your VPC resources to the AWS Lambda function without crossing the public internet. VPC endpoint is irrelevant to the current discussion.

Enable API caching in Amazon API Gateway to cache AWS Lambda function response - With caching, you can reduce the number of calls made to your AWS Lambda function and also improve the latency of requests to your API. Caching is best-effort and applications making frequent API calls to retrieve static data can benefit from a caching layer. Caching does not reduce the initialization time Lambda takes and hence is not an optimal solution for this use case.

References:

https://docs.aws.amazon.com/lambda/latest/dg/provisioned-concurrency.html

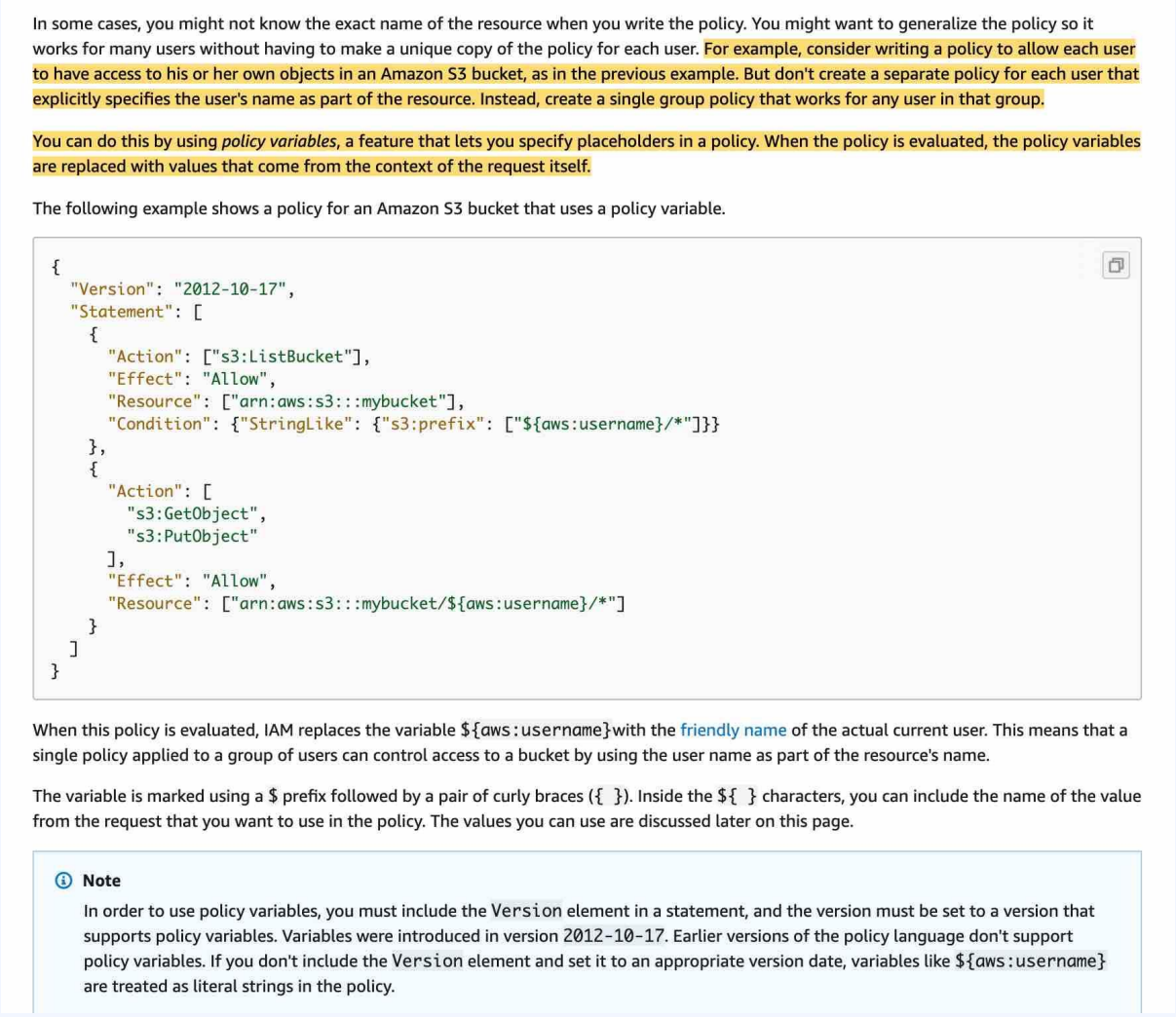
https://aws.amazon.com/blogs/compute/operating-lambda-performance-optimization-part-1/
https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-caching.html

Question 64:
Your AWS account is now growing to 200 users and you would like to provide each of these users a personal space in the S3 bucket 'my_company_space' with the prefix /home/<username>, where they have read/write access.
How can you do this efficiently?

- Create an S3 bucket policy and change it as users are added and removed
- Create one customer-managed policy with policy variables and attach it to a group of all users(Correct)
- Create one customer-managed policy per user and attach them to the relevant users
- Create inline policies for each user as they are onboarded

Explanation

Correct option:
Create one customer-managed policy with policy variables and attach it to a group of all users
You can assign access to "dynamically calculated resources" by using policy variables, a feature that lets you specify placeholders in a policy. When the policy is evaluated, the policy variables are replaced with values that come from the context of the request itself.
This is ideal when you want want to generalize the policy so it works for many users without having to make a unique copy of the policy for each user. For example, consider writing a policy to allow each user to have access to his or her own objects in an Amazon S3 bucket, as in the previous example. But don't create a separate policy for each user that explicitly specifies the user's name as part of the resource. Instead, create a single group policy that works for any user in that group.



In some cases, you might not know the exact name of the resource when you write the policy. You might want to generalize the policy so it works for many users without having to make a unique copy of the policy for each user. For example, consider writing a policy to allow each user to have access to his or her own objects in an Amazon S3 bucket, as in the previous example. But don't create a separate policy for each user that explicitly specifies the user's name as part of the resource. Instead, create a single group policy that works for any user in that group.

You can do this by using *policy variables*, a feature that lets you specify placeholders in a policy. When the policy is evaluated, the policy variables are replaced with values that come from the context of the request itself.

The following example shows a policy for an Amazon S3 bucket that uses a policy variable.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket"],
      "Condition": {"StringLike": {"s3:prefix": ["${aws:username}/*"]}}
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/${aws:username}/*"]
    }
  ]
}
```

When this policy is evaluated, IAM replaces the variable `${aws:username}` with the friendly name of the actual current user. This means that a single policy applied to a group of users can control access to a bucket by using the user name as part of the resource's name.

The variable is marked using a `$` prefix followed by a pair of curly braces (`{ }`). Inside the `${ }` characters, you can include the name of the value from the request that you want to use in the policy. The values you can use are discussed later on this page.

Note

In order to use policy variables, you must include the `Version` element in a statement, and the version must be set to a version that supports policy variables. Variables were introduced in version `2012-10-17`. Earlier versions of the policy language don't support policy variables. If you don't include the `Version` element and set it to an appropriate version date, variables like `${aws:username}` are treated as literal strings in the policy.

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

Incorrect options:

Create an S3 bucket policy and change it as users are added and removed

This doesn't scale and the S3 bucket policy size may be maxed out. The IAM policies bump up against a size limit (up to 2 kb for users, 5 kb for groups, and 10 kb for roles). S3 supports bucket policies of up 20 kb.

Create inline policies for each user as they are onboarded: This would work but doesn't scale and it's inefficient.

Create one customer-managed policy per user and attach them to the relevant users: This would work but doesn't scale and would be a nightmare to manage.

Reference:

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

Question 65:

A financial services company has developed a REST API which is deployed in an Auto Scaling Group behind an Application Load Balancer. The API stores the data payload in DynamoDB and the static content is served through S3. Upon analyzing the usage pattern, it's found that 80% of the read requests are shared across all users.

As a Developer Associate, how can you improve the application performance while optimizing the cost with the least development effort?

- Enable DynamoDB Accelerator (DAX) for DynamoDB and CloudFront for S3(Correct)
- Enable ElastiCache Redis for DynamoDB and CloudFront for S3
- Enable DAX for DynamoDB and ElastiCache Memcached for S3
- Enable ElastiCache Redis for DynamoDB and ElastiCache Memcached for S3

Explanation

Correct option:

Enable DynamoDB Accelerator (DAX) for DynamoDB and CloudFront for S3

DynamoDB Accelerator (DAX) is a fully managed, highly available, in-memory cache for Amazon DynamoDB that delivers up to a 10 times performance improvement—from milliseconds to microseconds—even at millions of requests per second.

DAX is tightly integrated with DynamoDB—you simply provision a DAX cluster, use the DAX client SDK to point your existing DynamoDB API calls at the DAX cluster, and let DAX handle the rest. Because DAX is API-compatible with DynamoDB, you don't have to make any functional application code changes. DAX is used to natively cache DynamoDB reads.

CloudFront is a content delivery network (CDN) service that delivers static and dynamic web content, video streams, and APIs around the world, securely and at scale. By design, delivering data out of CloudFront can be more cost-effective than delivering it from S3 directly to your users.

When a user requests content that you serve with CloudFront, their request is routed to a nearby Edge Location. If CloudFront has a cached copy of the requested file, CloudFront delivers it to the user, providing a fast (low-latency) response. If the file they've requested isn't yet cached, CloudFront retrieves it from your origin – for example, the S3 bucket where you've stored your content.

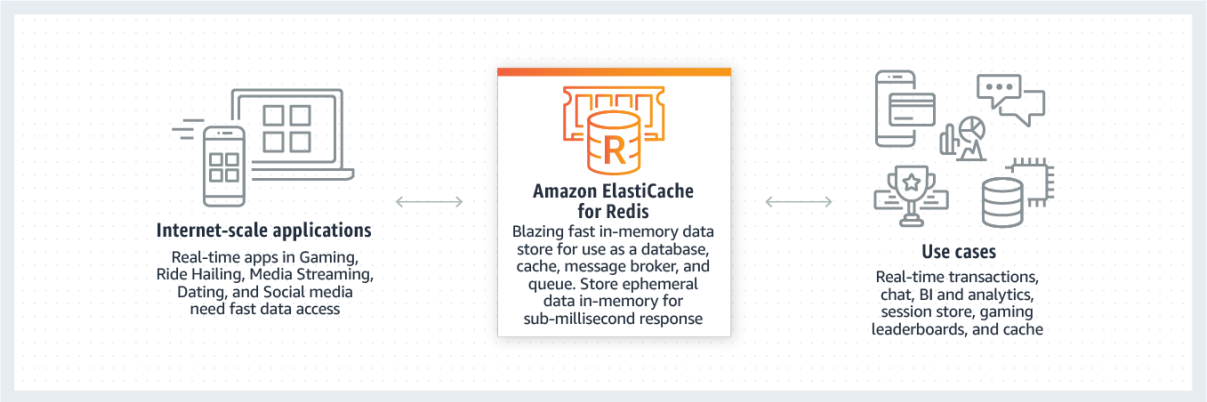
So, you can use CloudFront to improve application performance to serve static content from S3.

Incorrect options:

Enable ElastiCache Redis for DynamoDB and CloudFront for S3

Amazon ElastiCache for Redis is a blazing fast in-memory data store that provides sub-millisecond latency to power internet-scale real-time applications. Amazon ElastiCache for Redis is a great choice for real-time transactional and analytical processing use cases such as caching, chat/messaging, gaming leaderboards, geospatial, machine learning, media streaming, queues, real-time analytics, and session store.

ElastiCache for Redis Overview:



via - https://aws.amazon.com/elasticache/redis/

Although, you can integrate Redis with DynamoDB, it's much more involved from a development perspective. For the given use-case, you should use DAX which is a much better fit.

Enable DAX for DynamoDB and ElastiCache Memcached for S3

Enable ElastiCache Redis for DynamoDB and ElastiCache Memcached for S3

Amazon ElastiCache for Memcached is a Memcached-compatible in-memory key-value store service that can be used as a cache or a data store. Amazon ElastiCache for Memcached is a great choice for implementing an in-memory cache to decrease access latency, increase throughput, and ease the load off your relational or NoSQL database.

ElastiCache cannot be used as a cache to serve static content from S3, so both these options are incorrect.

References:

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

https://aws.amazon.com/blogs/networking-and-content-delivery/amazon-s3-amazon-cloudfront-a-match-made-in-the-cloud/

https://aws.amazon.com/elasticache/redis/