Question 1: **Incorrect**

A leading financial services company offers data aggregation services for Wall Street trading firms. The company bills its clients based on per unit of clickstream data provided to the clients. As the company operates in a regulated industry, it needs to have the same ordered clickstream data available for auditing within a window of 7 days.

As a Developer Associate, which of the following AWS services do you think provides the ability to run the billing process and auditing process on the given clickstream data in the same order?

- ○ **AWS Kinesis Data Streams** (Correct)
- ○ **AWS Kinesis Data Analytics**
- ◉ **Amazon SQS** (Incorrect)
- ○ **AWS Kinesis Data Firehose**

Explanation

Correct option:

AWS Kinesis Data Streams

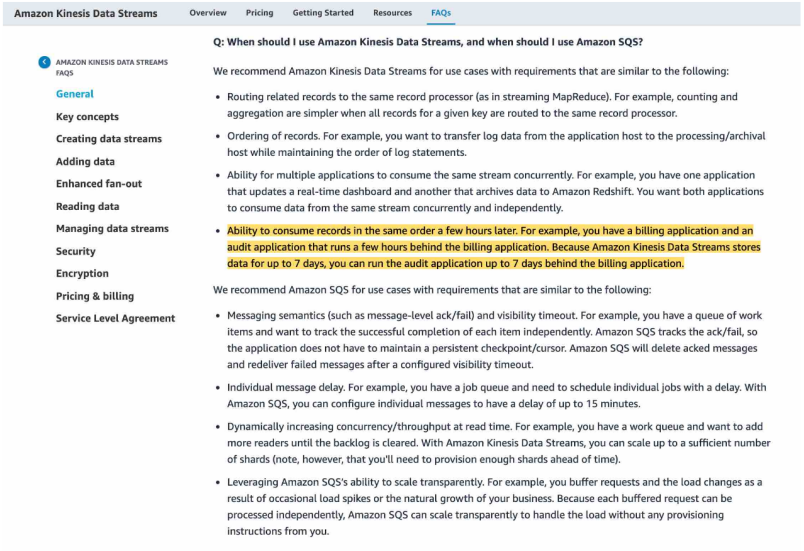
Amazon Kinesis Data Streams (KDS) is a massively scalable and durable real-time data streaming service. KDS can continuously capture gigabytes of data per second from hundreds of thousands of sources such as website clickstreams, database event streams, financial transactions, social media feeds, IT logs, and location-tracking events. The data collected is available in milliseconds to enable real-time analytics use cases such as real-time dashboards, real-time anomaly detection, dynamic pricing, and more.

Amazon Kinesis Data Streams enables real-time processing of streaming big data. It provides ordering of records, as well as the ability to read and/or replay records in the same order to multiple Amazon Kinesis Applications. The Amazon Kinesis Client Library (KCL) delivers all records for a given partition key to the same record processor, making it easier to build multiple applications reading from the same Amazon Kinesis data stream (for example, to perform counting, aggregation, and filtering). Amazon Kinesis Data Streams is recommended when you need the ability to consume records in the same order a few hours later.

For example, you have a billing application and an audit application that runs a few hours behind the billing application. By default, records of a stream are accessible for up to 24 hours from the time they are added to the stream. You can raise this limit to a maximum of 365 days. For the given use-case, Amazon Kinesis Data Streams can be configured to store data for up to 7 days and you can run the audit application up to 7 days behind the billing application.

billing application.

KDS provides the ability to consume records in the same order a few hours later



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

Incorrect options:

AWS Kinesis Data Firehose - Amazon Kinesis Data Firehose is the easiest way to load streaming data into data stores and analytics tools. It can capture, transform, and load streaming data into Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, and Splunk, enabling near real-time analytics with existing business intelligence tools and dashboards you're already using today. It is a fully managed service that automatically scales to match the throughput of your data and requires no ongoing administration. It can also batch, compress, and encrypt the data before loading it, minimizing the amount of storage used at the destination and increasing security. As Kinesis Data Firehose is used to load streaming data into data stores, therefore this option is incorrect.

AWS Kinesis Data Analytics - Amazon Kinesis Data Analytics is the easiest way to analyze streaming data in real-time. You can quickly build SQL queries and sophisticated Java applications using built-in templates and operators for common processing functions to organize, transform, aggregate, and analyze data at any scale. Kinesis Data Analytics enables you to easily and quickly build queries and sophisticated streaming applications in three simple steps: setup your streaming data sources, write your queries or streaming applications and set up your destination for processed data. As Kinesis Data Analytics is used to build SQL queries and sophisticated Java applications, therefore this option is incorrect.

Amazon SQS - Amazon Simple Queue Service (SQS) is a fully managed message queuing service that enables you to decouple and scale microservices, distributed systems, and serverless applications. SQS offers two types of message queues. Standard queues offer maximum throughput, best-effort ordering, and at-least-once delivery. SQS FIFO queues are designed to guarantee that messages are processed exactly once, in the exact order that they are sent. For SQS, you cannot have the same message being consumed by multiple consumers in the same order a few hours later, therefore this option is incorrect.

References:

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

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

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

Question 2: **Incorrect**

A company has a workload that requires 14,000 consistent IOPS for data that must be durable and secure. The compliance standards of the company state that the data should be secure at every stage of its lifecycle on all of the EBS volumes they use.

Which of the following statements are true regarding data security on EBS?

- (●) **EBS volumes do not support in-flight encryption but do support encryption at rest using KMS** **(Incorrect)**
- ( ) **EBS volumes don't support any encryption**
- ( ) **EBS volumes support both in-flight encryption and encryption at rest using KMS** **(Correct)**
- ( ) **EBS volumes support in-flight encryption but does not support encryption at rest**

Explanation
Correct option:

Amazon EBS works with AWS KMS to encrypt and decrypt your EBS volume. You can encrypt both the boot and data volumes of an EC2 instance. When you create an encrypted EBS volume and attach it to a supported instance type, the following types of data are encrypted:

1. Data at rest inside the volume

2. All data moving between the volume and the instance
3. All snapshots created from the volume
4. All volumes created from those snapshots

EBS volumes support both in-flight encryption and encryption at rest using KMS - This is a correct statement. Encryption operations occur on the servers that host EC2 instances, ensuring the security of both data-at-rest and data-in-transit between an instance and its attached EBS storage.

Incorrect options:

EBS volumes support in-flight encryption but do not support encryption at rest - This is an incorrect statement. As discussed above, all data moving between the volume and the instance is encrypted.

EBS volumes do not support in-flight encryption but do support encryption at rest using KMS - This is an incorrect statement. As discussed above, data at rest is also encrypted.

EBS volumes don't support any encryption - This is an incorrect statement. Amazon EBS encryption offers a straight-forward encryption solution for your EBS resources associated with your EC2 instances. With Amazon EBS encryption, you aren't required to build, maintain, and secure your own key management infrastructure.

Reference:

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

Question 3: **Correct**

You have an Auto Scaling group configured to a minimum capacity of 1 and a maximum capacity of 5, designed to launch EC2 instances across 3 Availability Zones. During a low utilization period, an entire Availability Zone went down and your application experienced downtime.

What can you do to ensure that your application remains highly available?

- ○ **Change the scaling metric of auto-scaling policy to network bytes**
- ○ **Configure ASG fast failover**
- ○ **Enable RDS Multi-AZ**
- ◉ **Increase the minimum instance capacity of the Auto Scaling Group to 2** **(Correct)**

Explanation
Correct option:

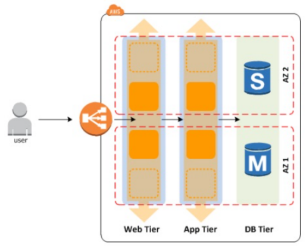
Increase the minimum instance capacity of the Auto Scaling Group to 2 -

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

Since a minimum capacity of 1 was defined, an instance was launched in only one AZ. This AZ went down, taking the application with it. If the minimum capacity is set to 2. As per Auto Scale AZ configuration, it would have launched 2 instances- one in each AZ, making the architecture disaster-proof and hence highly available.

Instance Distribution

Amazon EC2 Auto Scaling attempts to distribute instances evenly between the Availability Zones that are enabled for your Auto Scaling group. Amazon EC2 Auto Scaling does this by attempting to launch new instances in the Availability Zone with the fewest instances. If the attempt fails, however, Amazon EC2 Auto Scaling attempts to launch the instances in another Availability Zone until it succeeds. For Auto Scaling groups in a VPC, if there are multiple subnets in an Availability Zone, Amazon EC2 Auto Scaling selects a subnet from the Availability Zone at random.



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

Incorrect options:

Change the scaling metric of auto-scaling policy to network bytes - With target tracking scaling policies, you select a scaling metric and set a target value. You can use predefined customized metrics. Setting the metric to network bytes will not help in this context since the instances have to be spread across different AZs for high availability. The optimized way of doing it, is by defining minimum and maximum instance capacities, as discussed above.

Configure ASG fast failover - This is a made-up option, given as a distractor.

Enable RDS Multi-AZ - This configuration will make your database highly available. But for the current scenario, you will need to have more than 1 instance in separate availability zones to keep the application highly available.

References:

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

https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-maintain-instance-levels.html

https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scaling-target-tracking.html

Question 4: **Incorrect**

Your company uses an Application Load Balancer to route incoming end-user traffic to applications hosted on Amazon EC2 instances. The applications capture incoming request information and store it in the Amazon Relational Database Service (RDS) running on Microsoft SQL Server DB engines.

As part of new compliance rules, you need to capture the client's IP address. How will you achieve this?

- (●) **Use the header X-Forwarded-From** **(Incorrect)**
- ( ) **You can get the Client IP addresses from server access logs**
- ( ) **You can get the Client IP addresses from Elastic Load Balancing logs**
- ( ) **Use the header X-Forwarded-For** **(Correct)**

Explanation
Correct option:

Use the header X-Forwarded-For - The X-Forwarded-For request header helps you identify the IP address of a client when you use an HTTP or HTTPS load balancer. Because load balancers intercept traffic between clients and servers, your server access logs contain only the IP address of the load balancer. To see the IP address of the client, use the X-Forwarded-For request header. Elastic Load Balancing stores the IP address of the client in the X-Forwarded-For request header and passes the header to your server.

Incorrect options:

You can get the Client IP addresses from server access logs - As discussed above, Load Balancers intercept traffic between clients and servers, so server access logs will contain only the IP address of the load balancer.

Use the header X-Forwarded-From - This is a made-up option and given as a distractor.

You can get the Client IP addresses from Elastic Load Balancing logs - Elastic Load Balancing logs requests sent to the load balancer, including requests that never made it to the targets. For example, if a client sends a malformed request, or there are no healthy targets to respond to the request, the request is still logged. So, this is not the right option if we wish to collect the IP addresses of the clients that have access to the

instances.

References:

https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/x-forwarded-headers.html

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

Question 5: **Correct**

A cybersecurity company is running a serverless backend with several compute-heavy workflows running on Lambda functions. The development team has noticed a performance lag after analyzing the performance metrics for the Lambda functions.

As a Developer Associate, which of the following options would you suggest as the BEST solution to address the compute-heavy workloads?

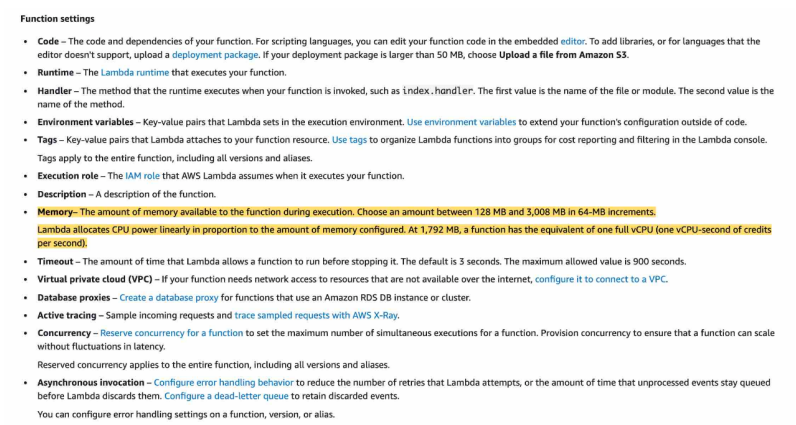
- ○ **Use reserved concurrency to account for the compute-heavy workflows**
- ○ **Invoke the Lambda functions asynchronously to process the compute-heavy workflows**
- ○ **Use provisioned concurrency to account for the compute-heavy workflows**
- ◉ **Increase the amount of memory available to the Lambda functions** (Correct)

Explanation
Correct option:

Increase the amount of memory available to the Lambda functions

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

In the AWS Lambda resource model, you choose the amount of memory you want for your function which allocates proportional CPU power and other resources. This means you will have access to more compute power when you choose one of the new larger settings. You can set your memory in 64MB increments from 128MB to 3008MB. You access these settings when you create a function or update its configuration. The settings are available using the AWS Management Console, AWS CLI, or SDKs.



Function settings

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

  Tags apply to the entire function, including all versions and aliases.
- **Execution role** – The IAM role that AWS Lambda assumes when it executes your function.
- **Description** – A description of the function.
- **Memory**– The amount of memory available to the function during execution. Choose an amount between 128 MB and 3,008 MB in 64-MB increments.

  Lambda allocates CPU power linearly in proportion to the amount of memory configured. At 1,792 MB, a function has the equivalent of one full vCPU (one vCPU-second of credits per second).
- **Timeout** – The amount of time that Lambda allows a function to run before stopping it. The default is 3 seconds. The maximum allowed value is 900 seconds.
- **Virtual private cloud (VPC)** – If your function needs network access to resources that are not available over the internet, configure it to connect to a VPC.
- **Database proxies** – Create a database proxy for functions that use an Amazon RDS DB instance or cluster.
- **Active tracing** – Sample incoming requests and trace sampled requests with AWS X-Ray.
- **Concurrency** – Reserve concurrency for a function to set the maximum number of simultaneous executions for a function. Provision concurrency to ensure that a function can scale without fluctuations in latency.

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

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

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

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

Incorrect options:

Invoke the Lambda functions asynchronously to process the compute-heavy workflows - When you invoke a function asynchronously, you don't wait for a response from the function code. You hand off the event to Lambda and Lambda handles the rest. You can configure how Lambda handles errors and can send invocation records to a downstream resource to chain together components of your application. The method of invocation has no bearing on the Lambda function's ability to process the compute-heavy workflows.

Use reserved concurrency to account for the compute-heavy workflows

Use provisioned concurrency to account for the compute-heavy workflows

Concurrency is the number of requests that your function is serving at any given time. When your function is invoked, Lambda allocates an instance of it to process the event. When the function code finishes running, it can handle another request. If the function is invoked again while a request is still being processed, another instance is allocated, which increases the function's concurrency. The type of concurrency has no bearing on the Lambda function's ability to process the compute-heavy workflows. So both these options are incorrect.

Reference:

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

Question 6: **Incorrect**

Your web application reads and writes data to your DynamoDB table. The table is provisioned with 400 Write Capacity Units (WCU's) shared across 4 partitions. One of the partitions receives 250 WCU/second while others receive much less. You receive the error 'ProvisionedThroughputExceededException'.

What is the likely cause of this error?

- ◯ **You have a hot partition** (Correct)
- ◯ **Configured IAM policy is wrong**
- ◯ **CloudWatch monitoring is lagging**
- ◉ **Write Capacity Units (WCU's) are applied across to all your DynamoDB tables and this needs reconfiguration** (Incorrect)

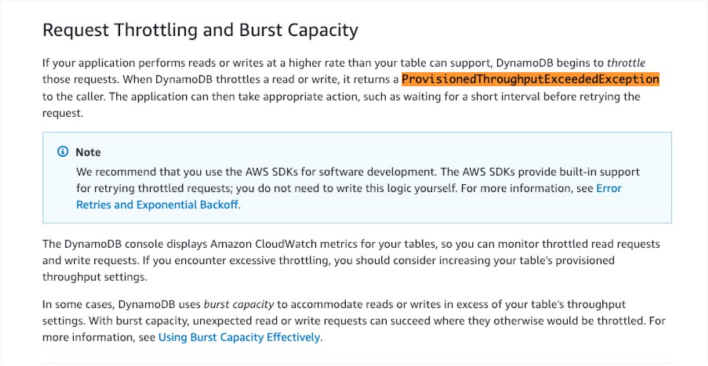
Explanation

Correct option:

You have a hot partition

It's not always possible to distribute read and write activity evenly. When data access is imbalanced, a "hot" partition can receive a higher volume of read and write traffic compared to other partitions. To better accommodate uneven access patterns, DynamoDB adaptive capacity enables your application to continue reading and writing to hot partitions without being throttled, provided that traffic does not exceed your table's total provisioned capacity or the partition maximum capacity.

ProvisionedThroughputExceededException explained:



Request Throttling and Burst Capacity

If your application performs reads or writes at a higher rate than your table can support, DynamoDB begins to *throttle* those requests. When DynamoDB throttles a read or write, it returns a `ProvisionedThroughputExceededException` to the caller. The application can then take appropriate action, such as waiting for a short interval before retrying the request.

Note

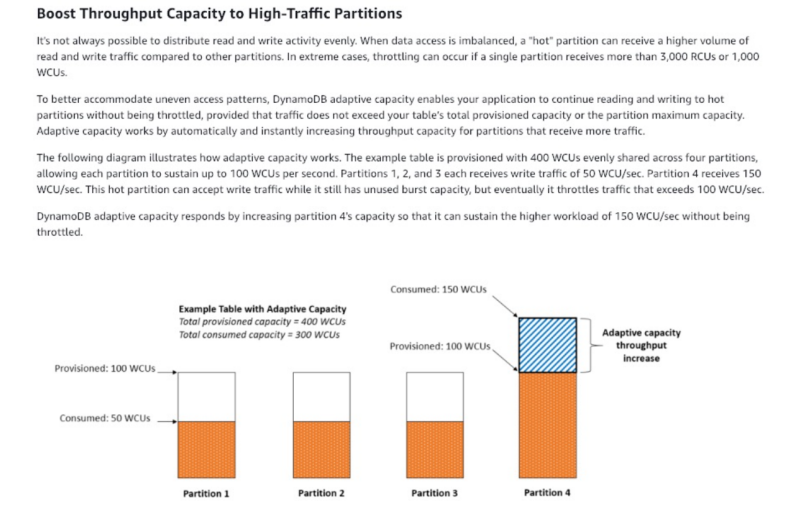
We recommend that you use the AWS SDKs for software development. The AWS SDKs provide built-in support for retrying throttled requests; you do not need to write this logic yourself. For more information, see Error Retries and Exponential Backoff.

The DynamoDB console displays Amazon CloudWatch metrics for your tables, so you can monitor throttled read requests and write requests. If you encounter excessive throttling, you should consider increasing your table's provisioned throughput settings.

In some cases, DynamoDB uses *burst capacity* to accommodate reads or writes in excess of your table's throughput settings. With burst capacity, unexpected read or write requests can succeed where they otherwise would be throttled. For more information, see Using Burst Capacity Effectively.

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

Hot partition explained:



via - https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/bp-partition-key-design.html

Incorrect options:

CloudWatch monitoring is lagging - The error is specific to DynamoDB itself and not to any connected service. CloudWatch is a fully managed service from AWS and does not result in throttling.

Configured IAM policy is wrong - The error is not associated with authorization but to exceeding something pre-configured value. So, it's clearly not a permissions issue.

Write-capacity units (WCU's) are applied across to all your DynamoDB tables and this needs reconfiguration - This statement is incorrect. Read Capacity Units and Write Capacity Units are specific to one table.

Reference:

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

Question 7: **Correct**

A company follows collaborative development practices. The engineering manager wants to isolate the development effort by setting up simulations of API components owned by various development teams.

Which API integration type is best suited for this requirement?

- ○ **HTTP**
- ◉ **MOCK** (Correct)
- ○ **HTTP_PROXY**
- ○ **AWS_PROXY**

Explanation

Correct option:

MOCK

This type of integration lets API Gateway return a response without sending the request further to the backend. This is useful for API testing because it can be used to test the integration setup without incurring charges for using the backend and to enable collaborative development of an API.

In collaborative development, a team can isolate their development effort by setting up simulations of API components owned by other teams by using the MOCK integrations. It is also used to return CORS-related headers to ensure that the API method permits CORS access. In fact, the API Gateway console integrates the OPTIONS method to support CORS with a mock integration.

Incorrect options:

AWS_PROXY - This type of integration lets an API method be integrated with the Lambda function invocation action with a flexible, versatile, and streamlined integration setup. This integration relies on direct interactions between the client and the integrated Lambda function.

HTTP_PROXY - The HTTP proxy integration allows a client to access the backend HTTP endpoints with a streamlined integration setup on single API method. You do not set the integration request or the integration response. API Gateway passes the incoming request from the client to the HTTP endpoint and passes the outgoing response from the HTTP endpoint to the client.

HTTP - This type of integration lets an API expose HTTP endpoints in the backend. With the HTTP integration, you must configure both the integration request and integration response. You must set up necessary data mappings from the method request to the integration request, and from the integration response to the method response.

Reference:

Question 8: **Incorrect**

You are a developer working with the AWS CLI to create Lambda functions that contain environment variables. Your functions will require over 50 environment variables consisting of sensitive information of database table names.

What is the total set size/number of environment variables you can create for AWS Lambda?

- (●) **The total size of all environment variables shouldn't exceed 8 KB. The maximum number of variables that can be created is 50** **(Incorrect)**
- ( ) **The total size of all environment variables shouldn't exceed 4 KB. There is no limit on the number of variables** **(Correct)**
- ( ) **The total size of all environment variables shouldn't exceed 8 KB. There is no limit on the number of variables**
- ( ) **The total size of all environment variables shouldn't exceed 4 KB. The maximum number of variables that can be created is 35**

Explanation

Correct option:

The total size of all environment variables shouldn't exceed 4 KB. There is no limit on the number of variables

An environment variable is a pair of strings that are stored in a function's version-specific configuration. The Lambda runtime makes environment variables available to your code and sets additional environment variables that contain information about the function and invocation request. The total size of all environment variables doesn't exceed 4 KB. There is no limit defined on the number of variables that can be used.

Incorrect options:

The total size of all environment variables shouldn't exceed 8 KB. The maximum number of variables that can be created is 50 - Incorrect option. The total size of environment variables cannot exceed 4 KB with no restriction on the number of variables.

The total size of all environment variables shouldn't exceed 8 KB. There is no limit on the number of variables - Incorrect option. The total size of environment variables cannot exceed 4 KB with no restriction on the number of variables.

The total size of all environment variables shouldn't exceed 4 KB. The maximum number of variables that can be created is 35 - Incorrect option. The total size of environment variables cannot exceed 4 KB with no restriction on the number of variables.

Question 9: **Incorrect**

An IT company has its serverless stack integrated with AWS X-Ray. The developer at the company has noticed a high volume of data going into X-Ray and the AWS monthly usage charges have skyrocketed as a result. The developer has requested changes to mitigate the issue.

As a Developer Associate, which of the following solutions would you recommend to obtain tracing trends while reducing costs with minimal disruption?

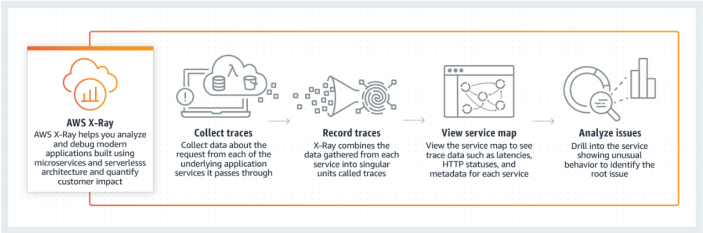
- ◯ **Custom configuration for the X-Ray agents**
- ◯ **Enable X-Ray sampling** **(Correct)**
- ◯ **Implement a network sampling rule**
- ◉ **Use Filter Expressions in the X-Ray console** **(Incorrect)**

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:



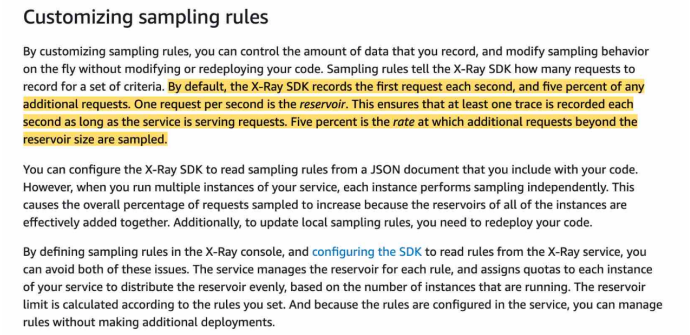
via - https://aws.amazon.com/xray/

Enable X-Ray 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. X-Ray sampling is enabled directly from the AWS console, hence your application code does not need to change.

You can also customize the X-Ray sampling rules:



Customizing sampling rules

By customizing sampling rules, you can control the amount of data that you record, and modify sampling behavior on the fly without modifying or redeploying your code. Sampling rules tell the X-Ray SDK how many requests to record for a set of criteria. By default, the X-Ray SDK records the first request each second, and five percent of any additional requests. One request per second is the *reservoir*. This ensures that at least one trace is recorded each second as long as the service is serving requests. Five percent is the *rate* at which additional requests beyond the reservoir size are sampled.

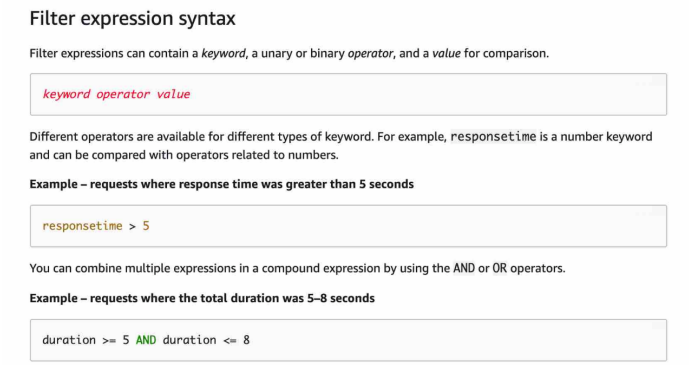
You can configure the X-Ray SDK to read sampling rules from a JSON document that you include with your code. However, when you run multiple instances of your service, each instance performs sampling independently. This causes the overall percentage of requests sampled to increase because the reservoirs of all of the instances are effectively added together. Additionally, to update local sampling rules, you need to redeploy your code.

By defining sampling rules in the X-Ray console, and configuring the SDK to read rules from the X-Ray service, you can avoid both of these issues. The service manages the reservoir for each rule, and assigns quotas to each instance of your service to distribute the reservoir evenly, based on the number of instances that are running. The reservoir limit is calculated according to the rules you set. And because the rules are configured in the service, you can manage rules without making additional deployments.

via - https://docs.aws.amazon.com/xray/latest/devguide/xray-console-sampling.html

Incorrect options:

Use Filter Expressions in the X-Ray console - When you choose a time period of traces to view in the X-Ray console, you might get more results than the console can display. You can narrow the results to just the traces that you want to find by using a filter expression. This option is not correct because it does not reduce the volume of data sent into the X-Ray console.



Filter expression syntax

Filter expressions can contain a *keyword*, a unary or binary *operator*, and a *value* for comparison.

```
keyword operator value
```

Different operators are available for different types of keyword. For example, `responsetime` is a number keyword and can be compared with operators related to numbers.

Example – requests where response time was greater than 5 seconds

```
responsetime > 5
```

You can combine multiple expressions in a compound expression by using the `AND` or `OR` operators.

Example – requests where the total duration was 5–8 seconds

```
duration >= 5 AND duration <= 8
```

via - https://docs.aws.amazon.com/xray/latest/devguide/xray-console-filters.html

Custom configuration for the X-Ray agents - You cannot do a custom configuration, instead you can do custom sampling rules. So this option is incorrect.

Implement a network sampling rule - This option has been added as a distractor.

References:

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

Question 10: **Incorrect**

A development team has deployed a REST API in Amazon API Gateway to two different stages - a test stage and a prod stage. The test stage is used as a test build and the prod stage as a stable build. After the updates have passed the test, the team wishes to promote the test stage to the prod stage.

Which of the following represents the optimal solution for this use-case?

- (●) **Delete the existing prod stage. Create a new stage with the same name (prod) and deploy the tested version on this stage** **(Incorrect)**
- ( ) **Update stage variable value from the stage name of test to that of prod** **(Correct)**
- ( ) **Deploy the API without choosing a stage. This way, the working deployment will be updated in all stages**
- ( ) **API performance is optimized in a different way for prod environments. Hence, promoting test to prod is not correct. The promotion should be done by redeploying the API to the prod stage**

Explanation

Correct option:

Update stage variable value from the stage name of test to that of prod

After creating your API, you must deploy it to make it callable by your users. To deploy an API, you create an API deployment and associate it with a stage. A stage is a logical reference to a lifecycle state of your API (for example, dev, prod, beta, v2). API stages are identified by the API ID and stage name. They're included in the URL that you use to invoke the API. Each stage is a named reference to a deployment of the API and is made available for client applications to call.

Stages enable robust version control of your API. In our current use-case, after the updates pass the test, you can promote the test stage to the prod stage. The promotion can be done by redeploying the API to the prod stage or updating a stage variable value from the stage name of test to that of prod.

Incorrect options:

Deploy the API without choosing a stage. This way, the working deployment will be updated in all stages - An API can only be deployed to a stage. Hence, it is not possible

to deploy an API without choosing a stage.

*Delete the existing prod stage. Create a new stage with the same name (prod) and deploy the tested version on this stage** - This is possible, but not an optimal way of deploying a change. Also, as prod refers to real production system, this option will result in downtime.

API performance is optimized in a different way for prod environments. Hence, promoting test to prod is not correct. The promotion should be done by redeploying the API to the prod stage - For each stage, you can optimize API performance by adjusting the default account-level request throttling limits and enabling API caching. And these settings can be changed/updated at any time.

Reference:

https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-deploy-api.html

Question 11: **Correct**

Your application is deployed automatically using AWS Elastic Beanstalk. Your YAML configuration files are stored in the folder .ebextensions and new files are added or updated often. The DevOps team does not want to re-deploy the application every time there are configuration changes, instead, they would rather manage configuration externally, securely, and have it load dynamically into the application at runtime.

What option allows you to do this?

- ○ **Use S3**
- ○ **Use Environment variables**
- ○ **Use Stage Variables**
- ◉ **Use SSM Parameter Store** **(Correct)**

Explanation
Correct option:

Use SSM Parameter Store

AWS Systems Manager Parameter Store provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, and license codes as parameter values. For the given use-

case, as the DevOps team does not want to re-deploy the application every time there are configuration changes, so they can use the SSM Parameter Store to store the configuration externally.

Incorrect options:

Use Environment variables - Environment variables provide another way to specify configuration options and credentials, and can be useful for scripting or temporarily setting a named profile as the default. Your application is not running AWS CLI. Since the use-case requires the configuration to be stored securely, so using Environment variables is ruled out, as these are not encrypted at rest and these are visible in clear text in the AWS Console as well as in the response of some actions of the Elastic BeanStalk API.

Use Stage Variables - You can use stage variables for managing multiple release stages for API Gateway, this is not what you are looking for here.

Use S3 - S3 offers the same benefit as the SSM Parameter Store where there are no servers to manage. With S3 you have to set encryption and choose other security options and there are more chances of misconfiguring security if you share your S3 bucket with other objects. You would have to create a custom setup to come close to the parameter store. Use Parameter Store and let AWS handle the rest.

Reference:

https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-paramstore.html

Question 12: **Incorrect**

A developer is designing an AWS CloudFormation template for deploying Amazon EC2 instances in numerous AWS accounts. The developer needs to select EC2 instances from a list of pre-approved instance types.

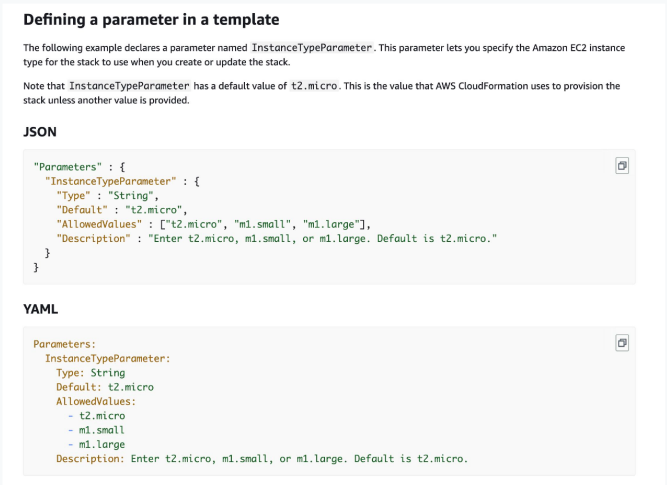
What measures could the developer take to integrate the list of authorized instance types into the CloudFormation template?

- ◉ **Configure separate parameters for each EC2 instance type in the CloudFormation template** **(Incorrect)**
- ○ **Configure a parameter with the list of EC2 instance types as AllowedValues in the CloudFormation template** **(Correct)**
- ○ **Configure a pseudo parameter with the list of EC2 instance types as AllowedValues in the CloudFormation template**
- ○ **Configure a mapping having a list of EC2 instance types as parameters in the CloudFormation template**

Explanation
Correct option:

Configure a parameter with the list of EC2 instance types as AllowedValues in the CloudFormation template

You can use the Parameters section to customize your templates. Parameters enable you to input custom values to your template each time you create or update a stack.



Defining a parameter in a template

The following example declares a parameter named `InstanceTypeParameter`. This parameter lets you specify the Amazon EC2 instance type for the stack to use when you create or update the stack.

Note that `InstanceTypeParameter` has a default value of `t2.micro`. This is the value that AWS CloudFormation uses to provision the stack unless another value is provided.

JSON

```
"Parameters" : {
  "InstanceTypeParameter" : {
    "Type" : "String",
    "Default" : "t2.micro",
    "AllowedValues" : ["t2.micro", "m1.small", "m1.large"],
    "Description" : "Enter t2.micro, m1.small, or m1.large. Default is t2.micro."
  }
}
```

YAML

```
Parameters:
  InstanceTypeParameter:
    Type: String
    Default: t2.micro
    AllowedValues:
      - t2.micro
      - m1.small
      - m1.large
    Description: Enter t2.micro, m1.small, or m1.large. Default is t2.micro.
```

via - https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html

AllowedValues refers to an array containing the list of values allowed for the parameter. When applied to a parameter of type String, the parameter value must be one of the allowed values. When applied to a parameter of type CommaDelimitedList, each value in the list must be one of the specified allowed values.

Incorrect options:

Configure separate parameters for each EC2 instance type in the CloudFormation template - Creating separate parameters for each instance type is semantically incorrect as the underlying value will point to the same resource but have multiple inputs.

Configure a mapping having a list of EC2 instance types as parameters in the CloudFormation template - The Mappings section matches a key to a corresponding set of named values. For example, if you want to set values based on a region, you can create a mapping that uses the region name as a key and contains the values you want to specify for each specific region. You use the Fn::FindInMap intrinsic function to retrieve values in a map. A mapping is not a list, rather, it consists of key value pairs. You can't include parameters, pseudo parameters, or intrinsic functions in the Mappings section. So, this option is incorrect.

Configure a pseudo parameter with the list of EC2 instance types as AllowedValues in the CloudFormation template - Pseudo parameters are parameters that are predefined by AWS CloudFormation. You don't declare them in your template. So, this

option is incorrect.

References:

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/mappings-section-structure.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/pseudo-parameter-reference.html

Question 13: **Incorrect**

A company wants to automate and orchestrate a multi-source high-volume flow of data in a scalable data management solution built using AWS services. The solution must ensure that the business rules and transformations run in sequence, handle reprocessing of data in case of errors, and require minimal maintenance.

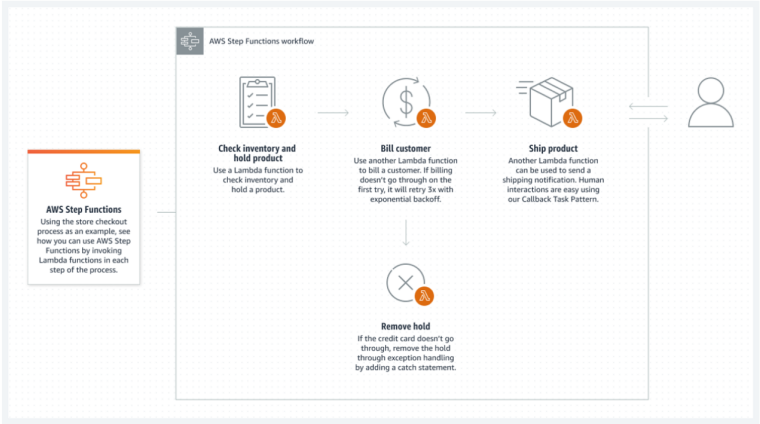
Which AWS service should the company use to manage and automate the orchestration of the data flows?

- (●) **AWS Glue** (Incorrect)
- ( ) **AWS Batch**
- ( ) **Amazon Kinesis Data Streams**
- ( ) **AWS Step Functions** (Correct)

Explanation
Correct option:

AWS Step Functions

AWS Step Functions is a visual workflow service that helps developers use AWS services to build distributed applications, automate processes, orchestrate microservices, and create data and machine learning (ML) pipelines.



via - https://aws.amazon.com/step-functions/

Incorrect options:

Amazon Kinesis Data Streams - Amazon Kinesis Data Streams is a serverless streaming data service that makes it easy to capture, process, and store data streams at any scale.

AWS Glue - AWS Glue is a serverless data integration service that makes it easier to discover, prepare, move, and integrate data from multiple sources for analytics, machine learning (ML), and application development.

AWS Batch - AWS Batch is a set of batch management capabilities that enables developers, scientists, and engineers to easily and efficiently run hundreds of thousands of batch computing jobs on AWS. AWS Batch dynamically provisions the optimal quantity and type of compute resources (e.g., CPU or memory optimized compute resources) based on the volume and specific resource requirements of the batch jobs submitted.

References:

https://aws.amazon.com/step-functions/

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

https://aws.amazon.com/glue/

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

Question 14: **Incorrect**

A team is checking the viability of using AWS Step Functions for creating a banking workflow for loan approvals. The web application will also have human approval as one of the steps in the workflow.

As a developer associate, which of the following would you identify as the key characteristics for AWS Step Function? (Select two)

- [ ] **Standard Workflows on AWS Step Functions are suitable for long-running, durable, and auditable workflows that do not support any human approval steps**
- [ ] **You should use Express Workflows for workloads with high event rates and short duration** **(Correct)**
- [x] **Express Workflows have a maximum duration of five minutes and Standard workflows have a maximum duration of 180 days or 6 months** **(Incorrect)**
- [ ] **Standard Workflows on AWS Step Functions are suitable for long-running, durable, and auditable workflows that can also support any human approval steps** **(Correct)**
- [x] **Both Standard and Express Workflows support all service integrations, activities, and design patterns** **(Incorrect)**

Explanation
Correct options:

Standard Workflows on AWS Step Functions are suitable for long-running, durable, and auditable workflows that can also support any human approval steps - Standard Workflows on AWS Step Functions are more suitable for long-running, durable, and auditable workflows where repeating workflow steps is expensive (e.g., restarting a long-running media transcode) or harmful (e.g., charging a credit card twice). Example workloads include training and deploying machine learning models, report generation, billing, credit card processing, and ordering and fulfillment processes. Step functions also support any human approval steps.

*You should use Express Workflows for workloads with high event rates and short duration** - You should use Express Workflows for workloads with high event rates and short durations. Express Workflows support event rates of more than 100,000 per second.

Incorrect options:

Standard Workflows on AWS Step Functions are suitable for long-running, durable, and auditable workflows that do not support any human approval steps - As Step functions support any human approval steps, so this option is incorrect.

Express Workflows have a maximum duration of five minutes and Standard workflows have a maximum duration of 180 days or 6 months - Express Workflows have a maximum duration of five minutes and Standard workflows have a maximum duration of one year.

Both Standard and Express Workflows support all service integrations, activities, and design patterns - Standard Workflows support all service integrations, activities, and design patterns. Express Workflows do not support activities, job-run (.sync), and Callback patterns.

Reference:

https://aws.amazon.com/step-functions/features/

https://aws.amazon.com/blogs/compute/implementing-serverless-manual-approval-steps-in-aws-step-functions-and-amazon-api-gateway/

Question 15: **Incorrect**

You company runs business logic on smaller software components that perform various functions. Some functions process information in a few seconds while others seem to take a long time to complete. Your manager asked you to decouple components that take a long time to ensure software applications stay responsive under load. You decide to configure Amazon Simple Queue Service (SQS) to work with your Elastic Beanstalk configuration.

Which of the following Elastic Beanstalk environment should you choose to meet this requirement?

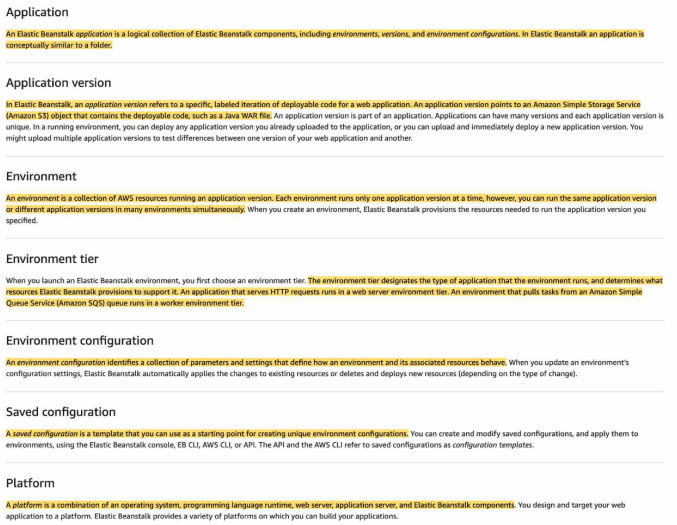
- ○ **Single Instance with Elastic IP**
- ○ **Single Instance Worker node**
- ○ **Dedicated worker environment** **(Correct)**
- ◉ **Load-balancing, Autoscaling environment** **(Incorrect)**

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*.

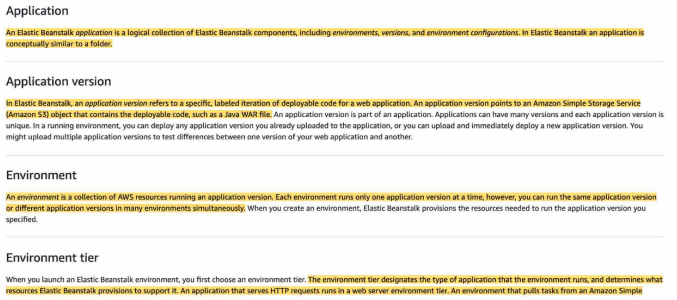
Platform

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

Dedicated worker environment - 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 emails, 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.



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

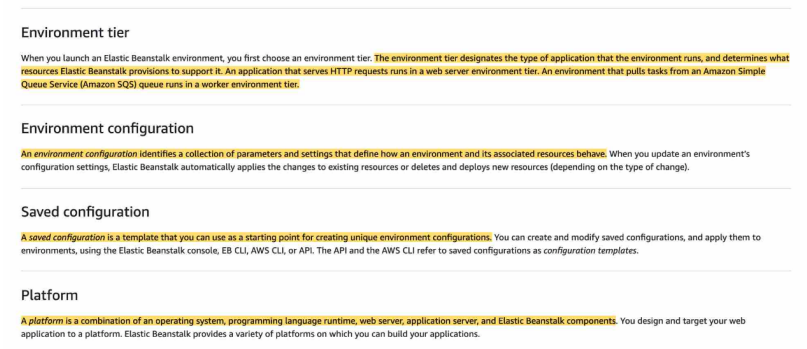
In Elastic Beanstalk, an *application version* refers to a specific, labeled iteration of deployable code for a web application. An application version points to an Amazon Simple Storage Service (Amazon S3) object that contains the deployable code, such as a Java WAR file. An application version is part 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



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/using-features-managing-env-tiers.html

Incorrect options:

Single Instance Worker node - Worker machines in Kubernetes are called nodes. Amazon EKS worker nodes are standard Amazon EC2 instances. EKS worker nodes are not to be confused with the Elastic Beanstalk worker environment. Since we are talking about the Elastic Beanstalk environment, this is not the correct choice.

Load-balancing, Autoscaling environment - A load-balancing and autoscaling environment uses the Elastic Load Balancing and Amazon EC2 Auto Scaling services to provision the Amazon EC2 instances that are required for your deployed application. Amazon EC2 Auto Scaling automatically starts additional instances to accommodate increasing load on your application. If your application requires scalability with the option of running in multiple Availability Zones, use a load-balancing, autoscaling environment. This is not the right environment for the given use-case since it will add costs to the overall solution.

Single Instance with Elastic IP - A single-instance environment contains one Amazon EC2 instance with an Elastic IP address. A single-instance environment doesn't have a load balancer, which can help you reduce costs compared to a load-balancing, autoscaling environment. This is not a highly available architecture, because if that one instance goes down then your application is down. This is not recommended for production environments.

Reference:

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

Question 16: **Correct**

As part of employee skills upgrade, the developers of your team have been delegated few responsibilities of DevOps engineers. Developers now have full control over modeling the entire software delivery process, from coding to deployment. As the team lead, you are now responsible for any manual approvals needed in the process.

Which of the following approaches supports the given workflow?

- ○ **Create multiple CodePipelines for each environment and link them using AWS Lambda**
- ◉ **Create one CodePipeline for your entire flow and add a manual approval step** **(Correct)**
- ○ **Create deeply integrated AWS CodePipelines for each environment**
- ○ **Use CodePipeline with Amazon Virtual Private Cloud**

Explanation

Correct option:

Create one CodePipeline for your entire flow and add a manual approval step - You can add an approval action to a stage in a CodePipeline pipeline at the point where you want the pipeline to stop so someone can manually approve or reject the action. Approval actions can't be added to Source stages. Source stages can contain only source actions.

Incorrect options:

Create multiple CodePipelines for each environment and link them using AWS Lambda - You can create Lambda functions and add them as actions in your pipelines but the approval step is confined to a workflow process and cannot be outsourced to any other AWS service.

Create deeply integrated AWS CodePipelines for each environment - You can use an AWS CloudFormation template in conjunction with AWS CodePipeline and AWS CodeCommit to create a test environment that deploys to your production environment when the changes to your application are approved, helping you automate a continuous delivery workflow. This is a possible answer but not an optimized way of achieving what the client needs.

Use CodePipeline with Amazon Virtual Private Cloud - AWS CodePipeline supports Amazon Virtual Private Cloud (Amazon VPC) endpoints powered by AWS PrivateLink. This means you can connect directly to CodePipeline through a private endpoint in your VPC, keeping all traffic inside your VPC and the AWS network. This is a robust security feature but is of no value for our current use-case.

References:

https://docs.aws.amazon.com/codepipeline/latest/userguide/approvals-action-add.html

https://docs.aws.amazon.com/codepipeline/latest/userguide/vpc-support.html

Question 17: **Correct**

A developer wants to integrate user-specific file upload and download features in an application that uses both Amazon Cognito user pools and Cognito identity pools for secure access with Amazon S3. The developer also wants to ensure that only authorized users can access their own files and that the files are securely saved and retrieved. The files are 5 KB to 500 MB in size.

What do you recommend as the most efficient solution?

- ○ **Use CloudFront Lambda@Edge to validate that the given file is uploaded to S3 and downloaded from S3 only by the authorized user**
- ○ **Integrate Amazon API Gateway with a Lambda function that validates that the given file is uploaded to S3 and downloaded from S3 only by the authorized user**
- ○ **Use S3 Event Notifications to trigger a Lambda function that validates that the given file is uploaded and downloaded only by the authorized user**
- ◉ **Leverage an IAM policy with the Amazon Cognito identity prefix to restrict users to use their own folders in Amazon S3** **(Correct)**

Explanation
Correct option:

Leverage an IAM policy with the Amazon Cognito identity prefix to restrict users to use their own folders in Amazon S3

Amazon Cognito identity pools (federated identities) enable you to create unique identities for your users and federate them with identity providers. With an identity pool, you can obtain temporary, limited-privilege AWS credentials to access other AWS services. Amazon Cognito identity pools support the following identity providers:

Public providers: Login with Amazon (identity pools), Facebook (identity pools), Google (identity pools), Sign in with Apple (identity pools).

Amazon Cognito user pools

OpenID Connect providers (identity pools)

SAML identity providers (identity pools)

Developer authenticated identities (identity pools)

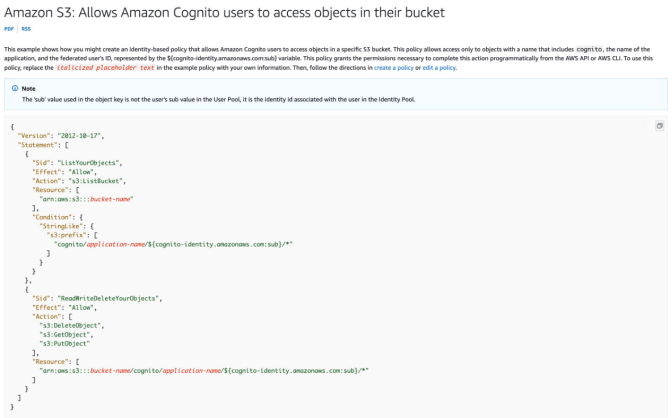
You can create an identity-based policy that allows Amazon Cognito users to access objects in a specific S3 bucket. This policy allows access only to objects with a name that includes Cognito, the name of the application, and the federated user's ID, represented by the ${cognito-identity.amazonaws.com:sub} variable.


Amazon S3: Allows Amazon Cognito users to access objects in their bucket

POF | RSS

This example shows how you might create an identity-based policy that allows Amazon Cognito users to access objects in a specific S3 bucket. This policy allows access only to objects with a name that includes cognito, the name of the application, and the federated user's ID, represented by the ${cognito-identity.amazonaws.com:sub} variable. This policy grants the permissions necessary to complete this action programmatically from the AWS API or AWS CLI. To use this policy, replace the italicized placeholder text in the example policy with your own information. Then, follow the directions in create a policy or edit a policy.

You can create an identity-based policy that allows Amazon Cognito users to access objects in a specific S3 bucket. This policy allows access only to objects with a name that includes Cognito, the name of the application, and the federated user's ID, represented by the ${cognito-identity.amazonaws.com:sub} variable.



via - https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_examples_s3_cognito-bucket.html

Incorrect options:

Use S3 Event Notifications to trigger a Lambda function that validates that the given file is uploaded and downloaded only by the authorized user - While it is certainly possible to build this solution, however, it is not the most optimal solution as it does not prevent an invalid upload of a file into another user's designated folder. So this option is incorrect.

Integrate Amazon API Gateway with a Lambda function that validates that the given file is uploaded to S3 and downloaded from S3 only by the authorized user - Again, it is certainly possible to build this solution, however, it is not the most optimal solution as it does not prevent an invalid upload of a file into another user's designated folder. So this option is incorrect.

Use CloudFront Lambda@Edge to validate that the given file is uploaded to S3 and downloaded from S3 only by the authorized user - This option assumes that the solution comprises a CloudFront distribution. This introduces inefficiency in the solution, as one needs to pay for CloudFront/Lambda@Edge and adds unnecessary hops in the data flow for both uploads and downloads.

References:

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_examples_s3_cognito-bucket.html

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

Question 18: **Correct**

A media company uses Amazon Simple Queue Service (SQS) queue to manage their transactions. With changing business needs, the payload size of the messages is increasing. The Team Lead of the project is worried about the 256 KB message size limit that SQS has.

What can be done to make the queue accept messages of a larger size?

- (●) **Use the SQS Extended Client** **(Correct)**
- ( ) **Get a service limit increase from AWS**
- ( ) **Use the MultiPart API**
- ( ) **Use gzip compression**

Explanation
Correct option:

Use the SQS Extended Client - To manage large Amazon Simple Queue Service (Amazon SQS) messages, you can use Amazon Simple Storage Service (Amazon S3) and the Amazon SQS Extended Client Library for Java. This is especially useful for storing and consuming messages up to 2 GB. Unless your application requires repeatedly creating queues and leaving them inactive or storing large amounts of data in your queues, consider using Amazon S3 for storing your data.

Incorrect options:

Use the MultiPart API - This is an incorrect statement. There is no multi-part API for Amazon Simple Queue Service.

Get a service limit increase from AWS - While it is possible to get service limits extended for certain AWS services, AWS already offers Extended Client to deal with queues that have larger messages.

Use gzip compression - You can compress the messages before sending them to the queue. The messages also need to be encoded after this to cater to SQS message standards. This adds bulk to the messages and will not be an optimal solution for the current scenario.

Reference:

https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-s3-messages.html

Question 19: **Incorrect**

A high-frequency stock trading firm is migrating their messaging queues from self-managed message-oriented middleware systems to Amazon SQS. The development team at the company wants to minimize the costs of using SQS.

As a Developer Associate, which of the following options would you recommend to address the given use-case?

- ○ **Use SQS long polling to retrieve messages from your Amazon SQS queues** (Correct)
- ○ **Use SQS visibility timeout to retrieve messages from your Amazon SQS queues**
- ○ **Use SQS message timer to retrieve messages from your Amazon SQS queues**
- ◉ **Use SQS short polling to retrieve messages from your Amazon SQS queues** (Incorrect)

Explanation

Correct option:

Use SQS long polling to retrieve messages from your Amazon SQS queues

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

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

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

Exam Alert:

Please review the differences between Short Polling vs Long Polling:


Amazon SQS short and long polling

PDF | Kindle | RSS

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

Amazon SQS short and long polling

PDF | Kindle | RSS

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

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

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

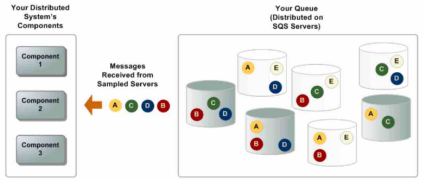
The following sections explain the details of short polling and long polling.

Topics

Consuming messages using short polling

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

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



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

Consuming message using long polling

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

Long polling offers the following benefits:

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

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

- Return messages as soon as they become available.

Differences between long and short polling

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

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

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

Incorrect options:

Use SQS short polling to retrieve messages from your Amazon SQS queues - With short polling, Amazon SQS sends the response right away, even if the query found no messages. You end up paying more because of the increased number of empty receives.

Use SQS visibility timeout to retrieve messages from your Amazon SQS queues - Visibility timeout is a period during which Amazon SQS prevents other consumers from receiving and processing a given message. The default visibility timeout for a message is 30 seconds. The minimum is 0 seconds. The maximum is 12 hours. You cannot use visibility timeout to retrieve messages from your Amazon SQS queues. This option has been added as a distractor.

Use SQS message timer to retrieve messages from your Amazon SQS queues - You can use message timers to set an initial invisibility period for a message added to a queue. So, if you send a message with a 60-second timer, the message isn't visible to consumers for its first 60 seconds in the queue. The default (minimum) delay for a message is 0 seconds. The maximum is 15 minutes. You cannot use message timer to retrieve messages from your Amazon SQS queues. This option has been added as a distractor.

Reference:

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

Question 20: **Incorrect**

An organization is moving its on-premises resources to the cloud. Source code will be moved to AWS CodeCommit and AWS CodeBuild will be used for compiling the source code using Apache Maven as a build tool. The organization wants the build environment should allow for scaling and running builds in parallel.

Which of the following options should the organization choose for their requirement?

- ○ **Choose a high-performance instance type for your CodeBuild instances**
- ◉ **Enable CodeBuild Auto Scaling** **(Incorrect)**
- ○ **Run CodeBuild in an Auto Scaling group**
- ○ **CodeBuild scales automatically, the organization does not have to do anything for scaling or for parallel builds** **(Correct)**

Explanation
Correct option:

CodeBuild scales automatically, the organization does not have to do anything for scaling or for parallel builds - AWS CodeBuild is a fully managed build service in the cloud. CodeBuild compiles your source code, runs unit tests, and produces artifacts that are ready to deploy. CodeBuild eliminates the need to provision, manage, and scale your own build servers. It provides prepackaged build environments for popular programming languages and build tools such as Apache Maven, Gradle, and more. You can also customize build environments in CodeBuild to use your own build tools. CodeBuild scales automatically to meet peak build requests.

Incorrect options:

Choose a high-performance instance type for your CodeBuild instances - For the current requirement, this is will not make any difference.

Run CodeBuild in an Auto Scaling Group - AWS CodeBuild is a managed service and scales automatically, does not need Auto Scaling Group to scale it up.

Enable CodeBuild Auto Scaling - This has been added as a distractor. CodeBuild scales automatically to meet peak build requests.

References:

https://docs.aws.amazon.com/codebuild/latest/userguide/welcome.html

https://docs.aws.amazon.com/codebuild/latest/userguide/build-env-ref-compute-types.html

Question 21: **Correct**

Your company is planning to move away from reserving EC2 instances and would like to adopt a more agile form of serverless architecture.

Which of the following is the simplest and the least effort way of deploying the Docker containers on this serverless architecture?

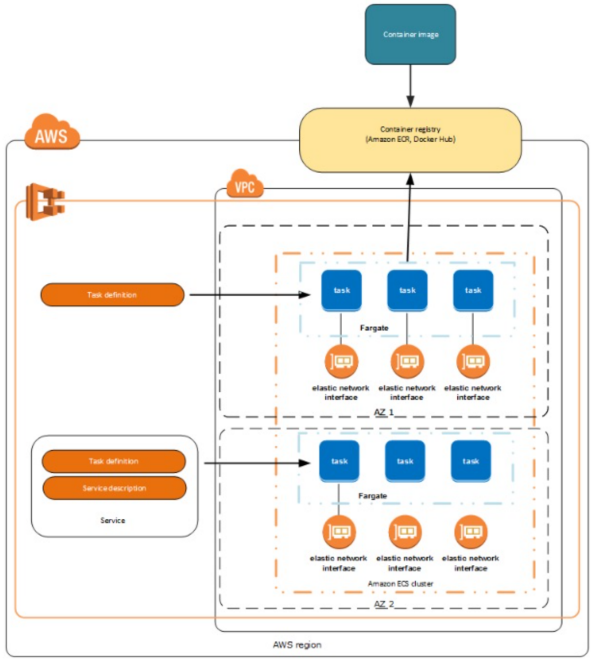
- ○ **Amazon Elastic Container Service (Amazon ECS) on EC2**
- ◉ **Amazon Elastic Container Service (Amazon ECS) on Fargate** **(Correct)**
- ○ **Amazon Elastic Kubernetes Service (Amazon EKS) on Fargate**
- ○ **AWS Elastic Beanstalk**

Explanation
Correct option:

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

AWS Fargate is a serverless compute engine for containers that works with both Amazon Elastic Container Service (ECS) and Amazon Elastic Kubernetes Service (EKS). Fargate makes it easy for you to focus on building your applications. Fargate removes the need to provision and manage servers, lets you specify and pay for resources per application, and improves security through application isolation by design.

ECS Fargate Overview:



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

Incorrect options:

Amazon Elastic Container Service (Amazon ECS) on EC2 - Amazon Elastic Container Service (Amazon ECS) is a highly scalable, fast, container management service that makes it easy to run, stop, and manage Docker containers on a cluster. ECS uses EC2 instances and hence cannot be called a serverless solution.

Amazon Elastic Kubernetes Service (Amazon EKS) on Fargate - Amazon Elastic Kubernetes Service (Amazon EKS) is a fully managed Kubernetes service. You can choose to run your EKS clusters using AWS Fargate, which is a serverless compute for containers. Since the use-case talks about the simplest and the least effort way to deploy Docker containers, EKS is not the best fit as you can use ECS Fargate to build a much easier solution. EKS is better suited to run Kubernetes on AWS without needing to install and operate your own Kubernetes control plane or worker nodes.

AWS Elastic Beanstalk - AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services. You can simply upload your code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring. At the same time, you retain full control over the AWS resources powering your application and can access the underlying resources at any time. Beanstalk uses EC2 instances for its deployment, hence cannot be called a serverless architecture.

References:

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

https://aws.amazon.com/eks/

https://aws.amazon.com/elasticbeanstalk/

Question 22: **Correct**

Your team lead has requested code review of your code for Lambda functions. Your code is written in Python and makes use of the Amazon Simple Storage Service (S3) to upload logs to an S3 bucket. After the review, your team lead has recommended reuse of execution context to improve the Lambda performance.

Which of the following actions will help you implement the recommendation?

- ◯ **Assign more RAM to the function**
- ◯ **Enable X-Ray integration**
- ◉ **Move the Amazon S3 client initialization, out of your function handler** **(Correct)**
- ◯ **Use environment variables to pass operational parameters**

Explanation
Correct option:

Move the Amazon S3 client initialization, out of your function handler - AWS best practices for Lambda suggest taking advantage of execution context reuse to improve the performance of your functions. Initialize SDK clients and database connections outside of the function handler, and cache static assets locally in the /tmp directory. Subsequent invocations processed by the same instance of your function can reuse these resources. This saves execution time and cost. To avoid potential data leaks across invocations, don't use the execution context to store user data, events, or other information with security implications.

Incorrect options:

Use environment variables to pass operational parameters - This is one of the suggested best practices for Lambda. By using environment variables to pass operational parameters you can avoid hard-coding useful information. But, this is not the right answer for the current use-case, since it talks about reusing context.

Assign more RAM to the function - Increasing RAM will help speed up the process. But, in the current question, the reviewer has specifically mentioned about reusing context. Hence, this is not the right answer.

Enable X-Ray integration - You can use AWS X-Ray to visualize the components of your application, identify performance bottlenecks, and troubleshoot requests that resulted in an error. Your Lambda functions send trace data to X-Ray, and X-Ray processes the data to generate a service map and searchable trace summaries. This is a useful tool for troubleshooting. But, for the current use-case, we already know the bottleneck that needs to be fixed and that is the context reuse.

References:

https://docs.aws.amazon.com/lambda/latest/dg/best-practices.html

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

Question 23: **Incorrect**

A company uses Amazon RDS as its database. For improved user experience, it has been decided that a highly reliable fully-managed caching layer has to be configured in front of RDS.

Which of the following is the right choice, keeping in mind that cache content regeneration is a costly activity?

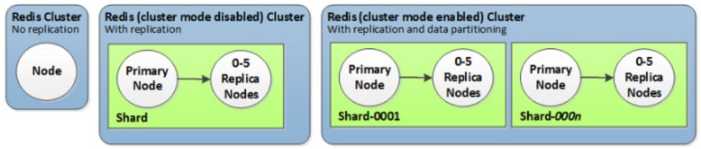
- ○ **Migrate the database to Amazon Redshift**
- ○ **Implement Amazon ElastiCache Redis in Cluster Mode** (Correct)
- ○ **Install Redis on an Amazon EC2 instance**
- ◉ **Implement Amazon ElastiCache Memcached** (Incorrect)

Explanation
Correct option:

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

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

Redis Cluster config:



Question 23: **Incorrect**

A company uses Amazon RDS as its database. For improved user experience, it has been decided that a highly reliable fully-managed caching layer has to be configured in front of RDS.

Which of the following is the right choice, keeping in mind that cache content regeneration is a costly activity?

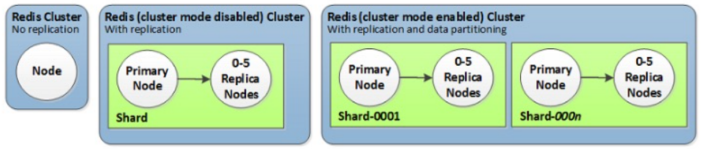
- ○ **Migrate the database to Amazon Redshift**
- ○ **Implement Amazon ElastiCache Redis in Cluster Mode** (Correct)
- ○ **Install Redis on an Amazon EC2 instance**
- ◉ **Implement Amazon ElastiCache Memcached** (Incorrect)

Explanation
Correct option:

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

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

Redis Cluster config:

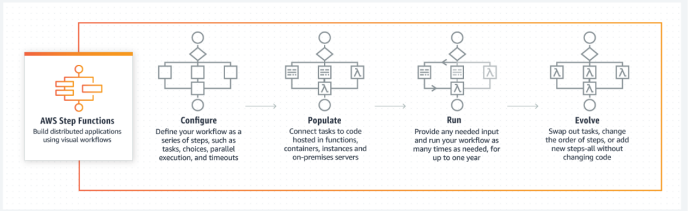


Explanation
Correct option:

Use AWS Step Functions state machines to orchestrate the workflow

AWS Step Functions is a web service that enables you to coordinate the components of distributed applications and microservices using visual workflows. You build applications from individual components that each perform a discrete function, or task, allowing you to scale and change applications quickly.

How Step Functions Work:



via - https://aws.amazon.com/step-functions/

The following are key features of AWS Step Functions:

Step Functions are based on the concepts of tasks and state machines. You define state machines using the JSON-based Amazon States Language. A state machine is defined by the states it contains and the relationships between them. States are elements in your state machine. Individual states can make decisions based on their input, perform actions, and pass output to other states. In this way, a state machine can orchestrate workflows.

Please see this note for a simple example of a State Machine:

A state machine is defined by the states it contains and the relationships between them.

The following is an example.

```
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass state",
  "StartAt": "HelloWorld",
  "States": {
    "HelloWorld": {
      "Type": "Pass",
      "Result": "Hello World!",
      "End": true
    }
  }
}
```

When an execution of this state machine is launched, the system begins with the state referenced in the `StartAt` field (`"HelloWorld"`). If this state has an `"End": true` field, the execution stops and returns a result. Otherwise, the system looks for a `"Next":` field and continues with that state next. This process repeats until the system reaches a terminal state (a state with `"Type": "Succeed"`, `"Type": "Fail"`, or `"End": true`), or a runtime error occurs.

The following rules apply to states within a state machine:

- States can occur in any order within the enclosing block, but the order in which they're listed doesn't affect the order in which they're run. The contents of the states determines this order.
- Within a state machine, there can be only one state that's designated as the `start` state, designated by the value of the `StartAt` field in the top-level structure. This state is the one that is executed first when the execution starts.
- Any state for which the `End` field is `true` is considered an `end` (or `terminal`) state. Depending on your state machine logic—for example, if your state machine has multiple branches of execution—you might have more than one `end` state.
- If your state machine consists of only one state, it can be both the `start` state and the `end` state.

via - https://docs.aws.amazon.com/step-functions/latest/dg/amazon-states-language-state-machine-structure.html

Incorrect options:

Use AWS Step Functions activities to orchestrate the workflow - In AWS Step Functions, activities are a way to associate code running somewhere (known as an activity worker) with a specific task in a state machine. When a Step Function reaches an activity task state, the workflow waits for an activity worker to poll for a task. For example, an activity worker can be an application running on an Amazon EC2 instance or an AWS Lambda function. AWS Step Functions activities cannot orchestrate a workflow.

Use AWS Glue to orchestrate the workflow - AWS Glue is a fully managed extract, transform, and load (ETL) service that makes it easy for customers to prepare and load their data for analytics. AWS Glue cannot orchestrate a workflow.

Use AWS Batch to orchestrate the workflow - AWS Batch runs batch computing jobs on the AWS Cloud. AWS Batch dynamically provisions the optimal quantity and type of compute resources (e.g., CPU or memory optimized instances) based on the volume and specific resource requirements of the batch jobs submitted. AWS Batch cannot orchestrate a workflow.

Reference:

https://aws.amazon.com/step-functions/

https://docs.aws.amazon.com/step-functions/latest/dg/amazon-states-language-state-machine-structure.html

Question 25: **Correct**

You team maintains a public API Gateway that is accessed by clients from another domain. Usage has been consistent for the last few months but recently it has more than doubled. As a result, your costs have gone up and would like to prevent other unauthorized domains from accessing your API.

Which of the following actions should you take?

- ○ **Use Mapping Templates**
- ◉ **Restrict access by using CORS** **(Correct)**
- ○ **Use Account-level throttling**
- ○ **Assign a Security Group to your API Gateway**

Explanation
Correct option:

Restrict access by using CORS - Cross-origin resource sharing (CORS) defines a way for client web applications that are loaded in one domain to interact with resources in a different domain. When your API's resources receive requests from a domain other than the API's own domain and you want to restrict servicing these requests, you must disable cross-origin resource sharing (CORS) for selected methods on the resource.

Incorrect options:

Use Account-level throttling - To prevent your API from being overwhelmed by too many requests, Amazon API Gateway throttles requests to your API. By default, API Gateway limits the steady-state request rate to 10,000 requests per second (rps). It limits the burst (that is, the maximum bucket size) to 5,000 requests across all APIs within an AWS account. This is Account-level throttling. As you see, this is about limit on the number of requests and is not a suitable answer for the current scenario.

Use Mapping Templates - A mapping template is a script expressed in Velocity Template Language (VTL) and applied to the payload using JSONPath expressions. Mapping templates help format/structure the data in a way that it is easily readable, unlike a server response that might always be easy to ready. Mapping Templates have nothing to do with access and are not useful for the current scenario.

Assign a Security Group to your API Gateway - API Gateway does not use security groups but uses resource policies, which are JSON policy documents that you attach to an API to control whether a specified principal (typically an IAM user or role) can invoke the API. You can restrict IP address using this, the downside being, an IP address can be changed by the accessing user. So, this is not an optimal solution for the current use case.

References:

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

https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-protect.html

https://docs.aws.amazon.com/apigateway/latest/developerguide/rest-api-data-transformations.html

Question 26: **Correct**

You have migrated an on-premise SQL Server database to an Amazon Relational Database Service (RDS) database attached to a VPC inside a private subnet. Also, the related Java application, hosted on-premise, has been moved to an Amazon Lambda function.

Which of the following should you implement to connect AWS Lambda function to its RDS instance?

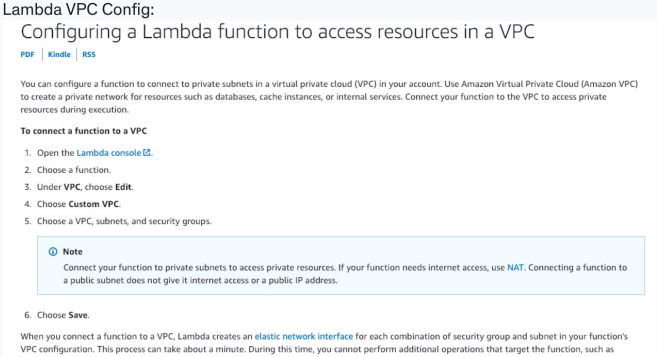
- ○ **Use Environment variables to pass in the RDS connection string**
- ◉ **Configure Lambda to connect to VPC with private subnet and Security Group needed to access RDS** **(Correct)**
- ○ **Use Lambda layers to connect to the internet and RDS separately**
- ○ **Configure lambda to connect to the public subnet that will give internet access and use Security Group to access RDS inside the private subnet**

Explanation

Correct option:

Configure Lambda to connect to VPC with private subnet and Security Group needed to access RDS - You can configure a Lambda function to connect to private subnets in a virtual private cloud (VPC) in your account. Use Amazon Virtual Private Cloud (Amazon VPC) to create a private network for resources such as databases, cache instances, or internal services. Connect your lambda function to the VPC to access private resources during execution. When you connect a function to a VPC, Lambda creates an elastic network interface for each combination of the security group and subnet in your function's VPC configuration. This is the right way of giving RDS access to Lambda.

Lambda VPC Config:



Configuring a Lambda function to access resources in a VPC

PDF | Kindle | RSS

You can configure a function to connect to private subnets in a virtual private cloud (VPC) in your account. Use Amazon Virtual Private Cloud (Amazon VPC) to create a private network for resources such as databases, cache instances, or internal services. Connect your function to the VPC to access private resources during execution.

To connect a function to a VPC

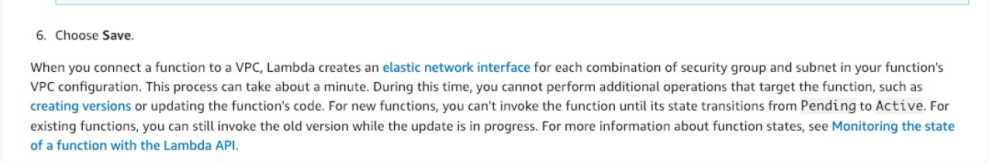
1. Open the Lambda console.
2. Choose a function.
3. Under **VPC**, choose **Edit**.
4. Choose **Custom VPC**.
5. Choose a VPC, subnets, and security groups.

Note

Connect your function to private subnets to access private resources. If your function needs internet access, use NAT. Connecting a function to a public subnet does not give it internet access or a public IP address.

6. Choose **Save**.

When you connect a function to a VPC, Lambda creates an elastic network interface for each combination of security group and subnet in your function's VPC configuration. This process can take about a minute. During this time, you cannot perform additional operations that target the function, such as



6. Choose **Save**.

When you connect a function to a VPC, Lambda creates an elastic network interface for each combination of security group and subnet in your function's VPC configuration. This process can take about a minute. During this time, you cannot perform additional operations that target the function, such as creating versions or updating the function's code. For new functions, you can't invoke the function until its state transitions from `Pending` to `Active`. For existing functions, you can still invoke the old version while the update is in progress. For more information about function states, see Monitoring the state of a function with the Lambda API.

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

Incorrect options:

Use Lambda layers to connect to the internet and RDS separately - You can configure your Lambda function to pull in additional code and content in the form of layers. A layer is a ZIP archive that contains libraries, a custom runtime, or other dependencies. Layers will not help in configuring access to RDS instance and hence is an incorrect choice.

Configure lambda to connect to the public subnet that will give internet access and use the Security Group to access RDS inside the private subnet - This is an incorrect statement. Connecting a Lambda function to a public subnet does not give it internet access or a public IP address. To grant internet access to your function, its associated VPC must have a NAT gateway (or NAT instance) in a public subnet.

Use Environment variables to pass in the RDS connection string - You can use environment variables to store secrets securely and adjust your function's behavior without updating code. You can use environment variables to exchange data with RDS, but you will still need access to RDS, which is not possible with just environment variables.

References:

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

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

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

https://aws.amazon.com/premiumsupport/knowledge-center/internet-access-lambda-function/

Question 27: **Correct**

As part of their on-boarding, the employees at an IT company need to upload their profile photos in a private S3 bucket. The company wants to build an in-house web application hosted on an EC2 instance that should display the profile photos in a secure way when the employees mark their attendance.

As a Developer Associate, which of the following solutions would you suggest to address this use-case?

- (●) **Save the S3 key for each user's profile photo in a DynamoDB table and use a lambda function to dynamically generate a pre-signed URL. Reference this URL for display via the web application** (Correct)
- ( ) **Make the S3 bucket public so that the application can reference the image URL for display**
- ( ) **Keep each user's profile image encoded in base64 format in a DynamoDB table and reference it from the application for display**
- ( ) **Keep each user's profile image encoded in base64 format in an RDS table and reference it from the application for display**

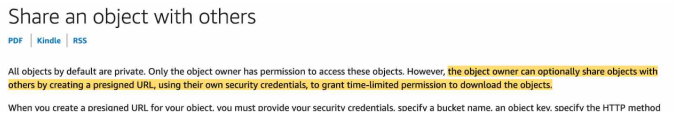
Explanation
Correct option:

"Save the S3 key for each user's profile photo in a DynamoDB table and use a lambda function to dynamically generate a pre-signed URL. Reference this URL for display via the web application"

On Amazon S3, all objects by default are private. Only the object owner has permission to access these objects. However, the object owner can optionally share objects with others by creating a pre-signed URL, using their own security credentials, to grant time-limited permission to download the objects.

You can also use an IAM instance profile to create a pre-signed URL. When you create a pre-signed URL for your object, you must provide your security credentials, specify a bucket name, an object key, specify the HTTP method (GET to download the object), and expiration date and time. The pre-signed URLs are valid only for the specified duration. So for the given use-case, the object key can be retrieved from the DynamoDB table, and then the application can generate the pre-signed URL using the IAM instance profile.

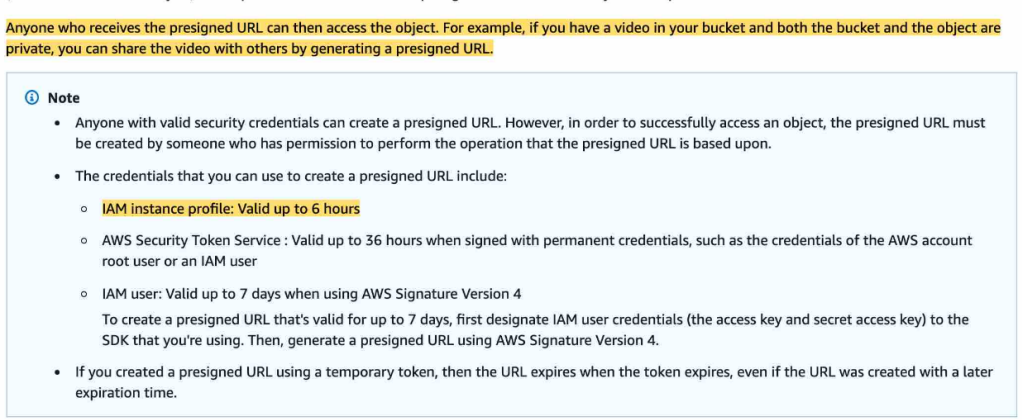
Please see this note for more details:


Share an object with others

PDF | Kindle | RSS

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

When you create a presigned URL for your object, you must provide your security credentials, specify a bucket name, an object key, specify the HTTP method



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

Incorrect options:

"Make the S3 bucket public so that the application can reference the image URL for display" - Making the S3 bucket public would violate the security and privacy requirements for the use-case, so this option is incorrect.

"Keep each user's profile image encoded in base64 format in a DynamoDB table and reference it from the application for display"

"Keep each user's profile image encoded in base64 format in an RDS table and reference it from the application for display"

It's a bad practice to keep the raw image data in the database itself. Also, it would not be possible to create a secure access URL for the image without a significant development effort. Hence both these options are incorrect.

Reference:

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

Question 28: Incorrect

You are running a cloud file storage website with an Internet-facing Application Load Balancer, which routes requests from users over the internet to 10 registered Amazon EC2 instances. Users are complaining that your website always asks them to re-authenticate when they switch pages. You are puzzled because this behavior is not seen in your local machine or dev environment.

What could be the reason?

- ○ **The Load Balancer does not have stickiness enabled** (Correct)
- ○ **The Load Balancer does not have TLS enabled**
- ◉ **The EC2 instances are logging out the users because the instances never have access to the client IPs because of the Load Balancer** (Incorrect)
- ○ **Application Load Balancer is in slow-start mode, which gives ALB a little more time to read and write session data**

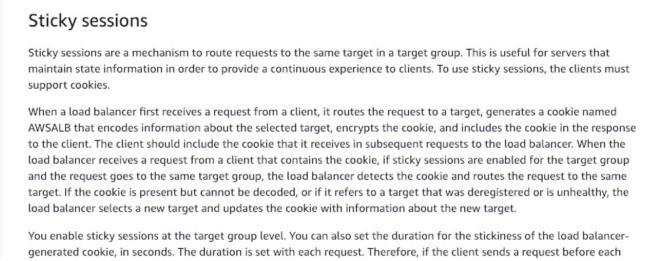
Explanation

Correct option:

The Load Balancer does not have stickiness enabled - Sticky sessions are a mechanism to route requests to the same target in a target group. This is useful for servers that maintain state information to provide a continuous experience to clients. To use sticky sessions, the clients must support cookies.

When a load balancer first receives a request from a client, it routes the request to a target, generates a cookie named AWSALB that encodes information about the selected target, encrypts the cookie, and includes the cookie in the response to the client. The client should include the cookie that it receives in subsequent requests to the load balancer. When the load balancer receives a request from a client that contains the cookie, if sticky sessions are enabled for the target group and the request goes to the same target group, the load balancer detects the cookie and routes the request to the same target.

More info here:

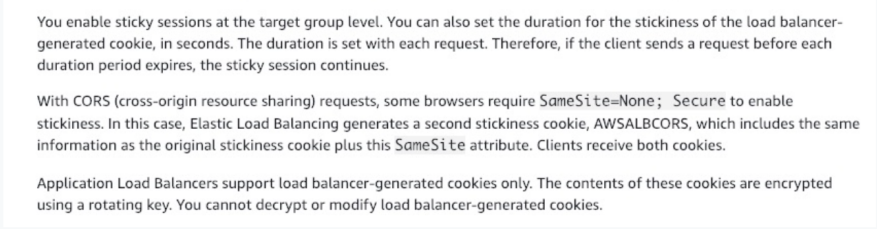


Sticky sessions

Sticky sessions are a mechanism to route requests to the same target in a target group. This is useful for servers that maintain state information in order to provide a continuous experience to clients. To use sticky sessions, the clients must support cookies.

When a load balancer first receives a request from a client, it routes the request to a target, generates a cookie named AWSALB that encodes information about the selected target, encrypts the cookie, and includes the cookie in the response to the client. The client should include the cookie that it receives in subsequent requests to the load balancer. When the load balancer receives a request from a client that contains the cookie, if sticky sessions are enabled for the target group and the request goes to the same target group, the load balancer detects the cookie and routes the request to the same target. If the cookie is present but cannot be decoded, or if it refers to a target that was deregistered or is unhealthy, the load balancer selects a new target and updates the cookie with information about the new target.

You enable sticky sessions at the target group level. You can also set the duration for the stickiness of the load balancer-generated cookie, in seconds. The duration is set with each request. Therefore, if the client sends a request before each



You enable sticky sessions at the target group level. You can also set the duration for the stickiness of the load balancer-generated cookie, in seconds. The duration is set with each request. Therefore, if the client sends a request before each duration period expires, the sticky session continues.

With CORS (cross-origin resource sharing) requests, some browsers require `SameSite=None; Secure` to enable stickiness. In this case, Elastic Load Balancing generates a second stickiness cookie, AWSALBCORS, which includes the same information as the original stickiness cookie plus this `SameSite` attribute. Clients receive both cookies.

Application Load Balancers support load balancer-generated cookies only. The contents of these cookies are encrypted using a rotating key. You cannot decrypt or modify load balancer-generated cookies.

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

Incorrect options:

Application Load Balancer is in slow-start mode, which gives ALB a little more time to read and write session data - This is an invalid statement. The load balancer serves as a single point of contact for clients and distributes incoming traffic across its healthy registered targets. By default, a target starts to receive its full share of requests as soon as it is registered with a target group and passes an initial health check. Using slow start mode gives targets time to warm up before the load balancer sends them a full share of requests. This does not help in session management.

The EC2 instances are logging out the users because the instances never have access to the client IPs because of the Load Balancer - This is an incorrect statement. Elastic Load Balancing stores the IP address of the client in the X-Forwarded-For request header and passes the header to the server. If needed, the server can read IP addresses from this data.

The Load Balancer does not have TLS enabled - To use an HTTPS listener, you must deploy at least one SSL/TLS server certificate on your load balancer. The load balancer uses a server certificate to terminate the front-end connection and then decrypt requests from clients before sending them to the targets. This does not help in session management.

References:

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

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

Question 29: **Correct**

A banking application needs to send real-time alerts and notifications based on any updates from the backend services. The company wants to avoid implementing complex polling mechanisms for these notifications.

Which of the following types of APIs supported by the Amazon API Gateway is the right fit?

- ○ **REST or HTTP APIs**
- ○ **REST APIs**
- ◉ **WebSocket APIs** **(Correct)**
- ○ **HTTP APIs**

Explanation
Correct option:

WebSocket APIs

In a WebSocket API, the client and the server can both send messages to each other at any time. Backend servers can easily push data to connected users and devices, avoiding the need to implement complex polling mechanisms.

For example, you could build a serverless application using an API Gateway WebSocket API and AWS Lambda to send and receive messages to and from individual users or groups of users in a chat room. Or you could invoke backend services such as AWS Lambda, Amazon Kinesis, or an HTTP endpoint based on message content.

You can use API Gateway WebSocket APIs to build secure, real-time communication applications without having to provision or manage any servers to manage connections or large-scale data exchanges. Targeted use cases include real-time applications such as the following:

1. Chat applications
2. Real-time dashboards such as stock tickers
3. Real-time alerts and notifications

API Gateway provides WebSocket API management functionality such as the following:

1. Monitoring and throttling of connections and messages
2. Using AWS X-Ray to trace messages as they travel through the APIs to backend services
3. Easy integration with HTTP/HTTPS endpoints

Incorrect options:

Incorrect options:

REST or HTTP APIs

REST APIs - An API Gateway REST API is made up of resources and methods. A resource is a logical entity that an app can access through a resource path. A method corresponds to a REST API request that is submitted by the user of your API and the response returned to the user.

For example, /incomes could be the path of a resource representing the income of the app user. A resource can have one or more operations that are defined by appropriate HTTP verbs such as GET, POST, PUT, PATCH, and DELETE. A combination of a resource path and an operation identifies a method of the API. For example, a POST /incomes method could add an income earned by the caller, and a GET /expenses method could query the reported expenses incurred by the caller.

HTTP APIs - HTTP APIs enable you to create RESTful APIs with lower latency and lower cost than REST APIs. You can use HTTP APIs to send requests to AWS Lambda functions or to any publicly routable HTTP endpoint.

For example, you can create an HTTP API that integrates with a Lambda function on the backend. When a client calls your API, API Gateway sends the request to the Lambda function and returns the function's response to the client.

Server push mechanism is not possible in REST and HTTP APIs.

Reference:

https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-overview-developer-experience.html

Question 30: **Correct**

A developer is working on an AWS Lambda function that reads data from Amazon S3 objects and writes the data to an Amazon DynamoDB table. Although the function triggers successfully from an S3 event notification upon object creation, it encounters a failure while attempting to write data to the DynamoDB table.

What is the probable reason for the failure?

- ○ **The Lambda function's reserved concurrency limit has been exceeded**
- ○ **DynamoDB table does not have a Gateway VPC Endpoint, which is required by the Lambda function for a successful write**
- ◉ **The Lambda function does not have IAM permissions to write to DynamoDB** (Correct)
- ○ **The Lambda function's provisioned concurrency limit has been exceeded**

Explanation
Correct option:

The Lambda function does not have IAM permissions to write to DynamoDB

You need to use an identity-based policy that allows read and write access to a specific Amazon DynamoDB table. To use this policy, attach the policy to a Lambda service role. A service role is a role that you create in your account to allow a service to perform actions on your behalf. That service role must include AWS Lambda as the principal in the trust policy. The role is then used to grant a Lambda function access to a DynamoDB table. By using an IAM policy and role to control access, you don't need to embed credentials in code and can tightly control which services the Lambda function can access.

Incorrect options:

The Lambda function's provisioned concurrency limit has been exceeded

The Lambda function's reserved concurrency limit has been exceeded

Reserved concurrency – 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.

Provisioned concurrency – Provisioned concurrency initializes a requested number of execution environments so that they are prepared to respond immediately to your function's invocations. Note that configuring provisioned concurrency incurs charges to your AWS account.

Neither reserved concurrency nor provisioned concurrency has any relevance to the given use case. Both options have been added as distractors.

DynamoDB table does not have a Gateway VPC Endpoint, which is required by the Lambda function for a successful write - Gateway endpoints provide reliable connectivity to Amazon S3 and DynamoDB without requiring an internet gateway or a NAT device for your VPC. Gateway endpoints do not enable AWS PrivateLink. This option acts as a distractor since the Lambda function is not provisioned within a VPC by default, so there is no need of a Gateway VPC Endpoint to access DynamoDB.

References:

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_examples_lambda-access-dynamodb.html

https://aws.amazon.com/blogs/security/how-to-create-an-aws-iam-policy-to-grant-aws-lambda-access-to-an-amazon-dynamodb-table/

Question 31: **Correct**

To meet compliance guidelines, a company needs to ensure replication of any data stored in its S3 buckets.

Which of the following characteristics are correct while configuring an S3 bucket for replication? (Select two)

- [x] **S3 lifecycle actions are not replicated with S3 replication** (Correct)
- [ ] **Once replication is enabled on a bucket, all old and new objects will be replicated**
- [ ] **Replicated objects do not retain metadata**
- [ ] **Object tags cannot be replicated across AWS Regions using Cross-Region Replication**
- [x] **Same-Region Replication (SRR) and Cross-Region Replication (CRR) can be configured at the S3 bucket level, a shared prefix level, or an object level using S3 object tags** (Correct)

Explanation

Correct options:

Same-Region Replication (SRR) and Cross-Region Replication (CRR) can be configured at the S3 bucket level, a shared prefix level, or an object level using S3 object tags - Amazon S3 Replication (CRR and SRR) is configured at the S3 bucket level, a shared prefix level, or an object level using S3 object tags. You add a replication configuration on your source bucket by specifying a destination bucket in the same or different AWS region for replication.

S3 lifecycle actions are not replicated with S3 replication - With S3 Replication (CRR and SRR), you can establish replication rules to make copies of your objects into another storage class, in the same or a different region. Lifecycle actions are not replicated, and if you want the same lifecycle configuration applied to both source and destination buckets, enable the same lifecycle configuration on both.

Incorrect options:

Object tags cannot be replicated across AWS Regions using Cross-Region Replication - Object tags can be replicated across AWS Regions using Cross-Region Replication. For customers with Cross-Region Replication already enabled, new permissions are required for tags to replicate.

Once replication is enabled on a bucket, all old and new objects will be replicated -

Once replication is enabled on a bucket, all old and new objects will be replicated - Replication only replicates the objects added to the bucket after replication is enabled on the bucket. Any objects present in the bucket before enabling replication are not replicated.

Replicated objects do not retain metadata - You can use replication to make copies of your objects that retain all metadata, such as the original object creation time and version IDs. This capability is important if you need to ensure that your replica is identical to the source object.

Reference:

https://docs.amazonaws.cn/en_us/AmazonS3/latest/userguide/replication.html

Question 32: **Incorrect**

A development team has created a new IAM user that has `s3:putObject` permission to write to an S3 bucket. This S3 bucket uses server-side encryption with AWS KMS managed keys (SSE-KMS) as the default encryption. Using the access key ID and the secret access key of the IAM user, the application received an access denied error when calling the `PutObject` API.

As a Developer Associate, how would you resolve this issue?

- ○ **Correct the ACL of the S3 bucket to allow the IAM user to upload encrypted objects**
- ◉ **Correct the policy of the IAM user to allow the `s3:Encrypt` action** **(Incorrect)**
- ○ **Correct the bucket policy of the S3 bucket to allow the IAM user to upload encrypted objects**
- ○ **Correct the policy of the IAM user to allow the `kms:GenerateDataKey` action** **(Correct)**

Explanation
Correct option:

Correct the policy of the IAM user to allow the `kms:GenerateDataKey` action - You can protect data at rest in Amazon S3 by using three different modes of server-side encryption: SSE-S3, SSE-C, or SSE-KMS. SSE-KMS requires that AWS manage the data key but you manage the customer master key (CMK) in AWS KMS. You can choose a customer managed CMK or the AWS managed CMK for Amazon S3 in your account. If you choose to encrypt your data using the standard features, AWS KMS and Amazon S3

you choose to encrypt your data using the standard features, AWS KMS and Amazon S3 perform the following actions:

1. Amazon S3 requests a plaintext data key and a copy of the key encrypted under the specified CMK.
2. AWS KMS generates a data key, encrypts it under the CMK, and sends both the plaintext data key and the encrypted data key to Amazon S3.
3. Amazon S3 encrypts the data using the data key and removes the plaintext key from memory as soon as possible after use.
4. Amazon S3 stores the encrypted data key as metadata with the encrypted data.

The error message indicates that your IAM user or role needs permission for the `kms:GenerateDataKey` action. This permission is required for buckets that use default encryption with a custom AWS KMS key.

In the JSON policy documents, look for policies related to AWS KMS access. Review statements with "Effect": "Allow" to check if the user or role has permissions for the kms:GenerateDataKey action on the bucket's AWS KMS key. If this permission is missing, then add the permission to the appropriate policy.

In the JSON policy documents, look for statements with "Effect": "Deny". Then, confirm that those statements don't deny the s3:PutObject action on the bucket. The statements must also not deny the IAM user or role access to the kms:GenerateDataKey action on the key used to encrypt the bucket. Additionally, make sure the necessary KMS and S3 permissions are not restricted using a VPC endpoint policy, service control policy, permissions boundary, or session policy.

Incorrect options:

Correct the policy of the IAM user to allow the `s3:Encrypt` action - This is an invalid action given only as a distractor.

Correct the bucket policy of the S3 bucket to allow the IAM user to upload encrypted objects - The user already has access to the bucket. What the user lacks is access to generate a KMS key, which is mandatory when a bucket is enabled for default encryption.

Correct the ACL of the S3 bucket to allow the IAM user to upload encrypted objects - Amazon S3 access control lists (ACLs) enable you to manage access to buckets and objects. Each bucket and object has an ACL attached to it as a subresource. It defines which AWS accounts or groups are granted access and the type of access. ACL is another way of giving access to S3 bucket objects. Permissions to use KMS keys will still be needed.

References:

https://aws.amazon.com/premiumsupport/knowledge-center/s3-access-denied-error-kms/

https://docs.aws.amazon.com/kms/latest/developerguide/services-s3.html

Question 33: **Correct**

You have a three-tier web application consisting of a web layer using AngularJS, an application layer using an AWS API Gateway and a data layer in an Amazon Relational Database Service (RDS) database. Your web application allows visitors to look up popular movies from the past. The company is looking at reducing the number of calls made to endpoint and improve latency to the API.

What can you do to improve performance?

- ○ **Use Stage Variables**
- ◉ **Enable API Gateway Caching** **(Correct)**
- ○ **Use Amazon Kinesis Data Streams to stream incoming data and reduce the burden on Gateway APIs**
- ○ **Use Mapping Templates**

Explanation

Correct option:

Enable API Gateway Caching - You can enable API caching in Amazon API Gateway to cache your endpoint's responses. With caching, you can reduce the number of calls made to your endpoint and also improve the latency of requests to your API. When you enable caching for a stage, API Gateway caches responses from your endpoint for a specified time-to-live (TTL) period, in seconds. API Gateway then responds to the request by looking up the endpoint response from the cache instead of making a request to your endpoint. The default TTL value for API caching is 300 seconds. The maximum TTL value is 3600 seconds. TTL=0 means caching is disabled.

Incorrect options:

Use Mapping Templates - A mapping template is a script expressed in Velocity Template Language (VTL) and applied to the payload using JSONPath expressions. Mapping templates help format/structure the data in a way that it is easily readable, unlike a server response that might always be easy to ready. Mapping Templates do not help in latency issues of the APIs.

Use Stage Variables - Stage variables act like environment variables and can be used to change the behavior of your API Gateway methods for each deployment stage; for example, making it possible to reach a different back end depending on which stage the API is running on. Stage variables do not help in latency issues.

Use Amazon Kinesis Data Streams to stream incoming data and reduce the burden on Gateway APIs - Amazon Kinesis Data Streams (KDS) is a massively scalable and durable real-time data streaming service. KDS can continuously capture gigabytes of data per second from hundreds of thousands of sources such as website clickstreams, database event streams, financial transactions, social media feeds, IT logs, and location-

data per second from hundreds of thousands of sources such as website clickstreams, database event streams, financial transactions, social media feeds, IT logs, and location-tracking events.

Reference:

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

Question 34: **Incorrect**

A digital marketing company has its website hosted on an Amazon S3 bucket A. The development team notices that the web fonts that are hosted on another S3 bucket B are not loading on the website.

Which of the following solutions can be used to address this issue?

- ○ **Configure CORS on the bucket A that is hosting the website to allow any origin to respond to requests**
- ○ **Enable versioning on both the buckets to facilitate correct functioning of the website**
- ◉ **Update bucket policies on both bucket A and bucket B to allow successful loading of the web fonts on the website** **(Incorrect)**
- ○ **Configure CORS on the bucket B that is hosting the web fonts to allow Bucket A origin to make the requests** **(Correct)**

Explanation
Correct option:

Configure CORS on the bucket B that is hosting the web fonts to allow Bucket A origin to make the requests

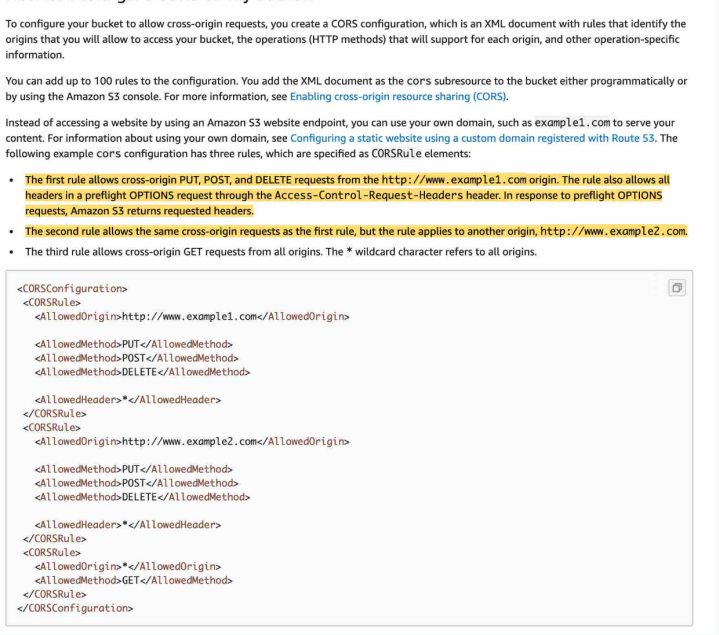
Cross-origin resource sharing (CORS) defines a way for client web applications that are loaded in one domain to interact with resources in a different domain.

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

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

Please see this note for more details:

How do I configure CORS on my bucket?



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

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

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

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

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

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

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

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

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

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

Incorrect options:

Enable versioning on both the buckets to facilitate the correct functioning of the website - This option is a distractor and versioning will not help to address the web fonts loading issue on the website.

Update bucket policies on both bucket A and bucket B to allow successful loading of the web fonts on the website - It's not the bucket policies but the CORS configuration on bucket B that needs to be updated to allow web fonts to be loaded on the website. Updating bucket policies will not help to address the web fonts loading issue on the website.

Configure CORS on the bucket A that is hosting the website to allow any origin to respond to requests - The CORS configuration needs to be updated on bucket B to allow web fonts to be loaded on the website hosted on bucket A. So this option in incorrect.

Reference:

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

Question 35: **Incorrect**

AWS CloudFormation helps model and provision all the cloud infrastructure resources needed for your business.

Which of the following services rely on CloudFormation to provision resources (Select two)?

- [ ] **AWS CodeBuild**
- [ ] **AWS Serverless Application Model (AWS SAM)** (Correct)
- [ ] **AWS Elastic Beanstalk** (Correct)
- [x] **AWS Lambda** (Incorrect)
- [x] **AWS Autoscaling** (Incorrect)

Explanation

Correct option:

AWS Elastic Beanstalk - AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS. Elastic Beanstalk uses AWS CloudFormation to launch the resources in your environment and propagate configuration changes.

AWS Serverless Application Model (AWS SAM) - You use the AWS SAM specification to define your serverless application. AWS SAM templates are an extension of AWS CloudFormation templates, with some additional components that make them easier to work with. AWS SAM needs CloudFormation templates as a basis for its configuration.

Incorrect options:

AWS Lambda - AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume. Hence, Lamda does not need CloudFormation to run its services.

AWS Autoscaling - AWS Auto Scaling monitors your applications and automatically adjusts the capacity to maintain steady, predictable performance at the lowest possible cost. Using AWS Auto Scaling, it's easy to setup application scaling for multiple resources across multiple services in minutes. Auto Scaling used CloudFormation but is not a mandatory requirement.

AWS CodeBuild - AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy. With CodeBuild, you don't need to provision, manage, and scale your own build servers. AWS CodePipeline uses AWS CloudFormation as a deployment action but is not a mandatory service.

Question 36: **Correct**

An intern at an IT company is getting started with AWS Cloud and wants to understand the following Amazon S3 bucket access policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ListAllS3Buckets",
            "Effect": "Allow",
            "Action": ["s3:ListAllMyBuckets"],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowBucketLevelActions",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowBucketObjectActions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:s3:::*/*"
        },
        {
            "Sid": "RequireMFAForProductionBucket",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::Production/*",
                "arn:aws:s3:::Production"
            ],
            "Condition": {
                "NumericGreaterThanIfExists": {"aws:MultiFactorAuthAge":
"1800"}
            }
        }
    ]
}
```

As a Developer Associate, can you help him identify what the policy is for?

- ○ **Allows IAM users to access their own home directory in Amazon S3, programmatically and in the console**
- ○ **Allows full S3 access to an Amazon Cognito user, but explicitly denies access to the Production bucket if the Cognito user is not authenticated**
- ◉ **Allows full S3 access, but explicitly denies access to the Production bucket if the user has not signed in using MFA within the last thirty minutes** (Correct)
- ○ **Allows a user to manage a single Amazon S3 bucket and denies every other AWS action and resource if the user is not signed in using MFA within last thirty minutes**

Explanation
Correct option:

Allows full S3 access, but explicitly denies access to the Production bucket if the user has not signed in using MFA within the last thirty minutes - This example shows how you might create a policy that allows an Amazon S3 user to access any bucket, including updating, adding, and deleting objects. However, it explicitly denies access to the Production bucket if the user has not signed in using multi-factor authentication (MFA) within the last thirty minutes. This policy grants the permissions necessary to perform this action in the console or programmatically using the AWS CLI or AWS API.

This policy never allows programmatic access to the Production bucket using long-term user access keys. This is accomplished using the aws:MultiFactorAuthAge condition key with the NumericGreaterThanIfExists condition operator. This policy condition returns true if MFA is not present or if the age of the MFA is greater than 30 minutes. In those situations, access is denied. To access the Production bucket programmatically, the S3 user must use temporary credentials that were generated in the last 30 minutes using the GetSessionToken API operation.

Incorrect options:

Allows a user to manage a single Amazon S3 bucket and denies every other AWS action and resource if the user is not signed in using MFA within last thirty minutes

Allows full S3 access to an Amazon Cognito user, but explicitly denies access to the Production bucket if the Cognito user is not authenticated

Allows IAM users to access their own home directory in Amazon S3, programmatically and in the console

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

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_examples_s3_full-access-except-production.html

Question 37: **Incorrect**

A company has more than 100 million members worldwide enjoying 125 million hours of TV shows and movies each day. The company uses AWS for nearly all its computing and storage needs, which use more than 10,000 server instances on AWS. This results in an extremely complex and dynamic networking environment where applications are constantly communicating inside AWS and across the Internet. Monitoring and optimizing its network is critical for the company.

The company needs a solution for ingesting and analyzing the multiple terabytes of real-time data its network generates daily in the form of flow logs. Which technology/service should the company use to ingest this data economically and has the flexibility to direct this data to other downstream systems?

- ○ **Amazon Kinesis Data Streams** (Correct)
- ○ **AWS Glue**
- ◉ **Amazon Kinesis Firehose** (Incorrect)
- ○ **Amazon Simple Queue Service (SQS)**

Explanation
Correct option:

Amazon Kinesis Data Streams

Amazon Kinesis Data Streams (KDS) is a massively scalable and durable real-time data streaming service. KDS can continuously capture gigabytes of data per second from hundreds of thousands of sources such as website clickstreams, database event streams, financial transactions, social media feeds, IT logs, and location-tracking events. The data collected is available in milliseconds to enable real-time analytics use cases such as real-time dashboards, real-time anomaly detection, dynamic pricing, and more.

Kinesis Data Streams enables real-time processing of streaming big data. It provides ordering of records, as well as the ability to read and/or replay records in the same order to multiple Amazon Kinesis Applications. The Amazon Kinesis Client Library (KCL) delivers all records for a given partition key to the same record processor, making it easier to build multiple applications reading from the same Amazon Kinesis data stream

Question 38: **Incorrect**

A telecommunications company that provides internet service for mobile device users maintains over 100 c4.large instances in the us-east-1 region. The EC2 instances run complex algorithms. The manager would like to track CPU utilization of the EC2 instances as frequently as every 10 seconds.

Which of the following represents the BEST solution for the given use-case?

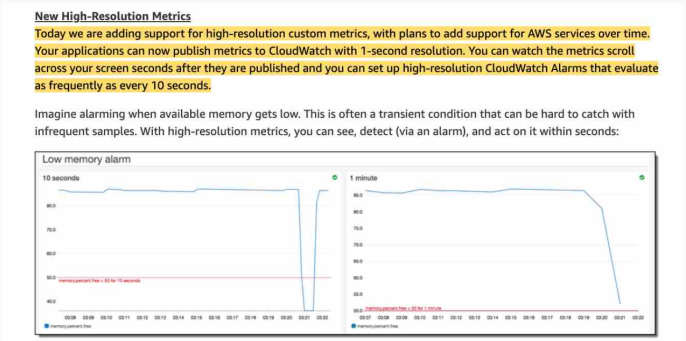
- ○ **Create a high-resolution custom metric and push the data using a script triggered every 10 seconds** **(Correct)**
- ○ **Simply get it from the CloudWatch Metrics**
- ○ **Open a support ticket with AWS**
- ◉ **Enable EC2 detailed monitoring** **(Incorrect)**

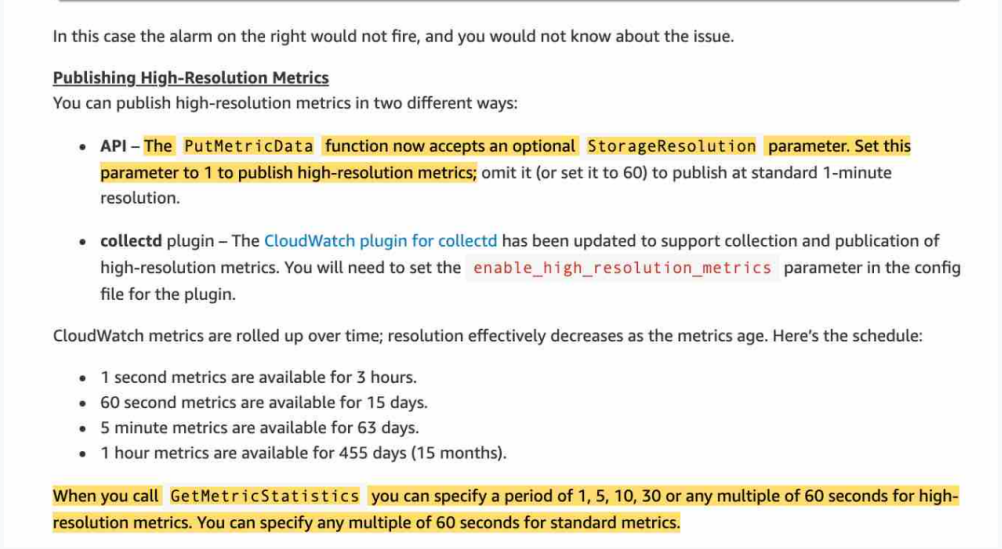
Explanation

Correct option:

Create a high-resolution custom metric and push the data using a script triggered every 10 seconds

Using high-resolution custom metric, your applications can publish metrics to CloudWatch with 1-second resolution. You can watch the metrics scroll across your screen seconds after they are published and you can set up high-resolution CloudWatch Alarms that evaluate as frequently as every 10 seconds. You can alert with High-Resolution Alarms, as frequently as 10-second periods. High-Resolution Alarms allow you to react and take actions faster and support the same actions available today with standard 1-minute alarms.





In this case the alarm on the right would not fire, and you would not know about the issue.

Publishing High-Resolution Metrics
You can publish high-resolution metrics in two different ways:

- **API** – The `PutMetricData` function now accepts an optional `StorageResolution` parameter. Set this parameter to 1 to publish high-resolution metrics; omit it (or set it to 60) to publish at standard 1-minute resolution.
- **collectd** plugin – The CloudWatch plugin for collectd has been updated to support collection and publication of high-resolution metrics. You will need to set the `enable_high_resolution_metrics` parameter in the config file for the plugin.

CloudWatch metrics are rolled up over time; resolution effectively decreases as the metrics age. Here's the schedule:

- 1 second metrics are available for 3 hours.
- 60 second metrics are available for 15 days.
- 5 minute metrics are available for 63 days.
- 1 hour metrics are available for 455 days (15 months).

When you call `GetMetricStatistics` you can specify a period of 1, 5, 10, 30 or any multiple of 60 seconds for high-resolution metrics. You can specify any multiple of 60 seconds for standard metrics.

via - https://aws.amazon.com/blogs/aws/new-high-resolution-custom-metrics-and-alarms-for-amazon-cloudwatch/

Incorrect options:

Enable EC2 detailed monitoring - As part of basic monitoring, Amazon EC2 sends metric data to CloudWatch in 5-minute periods. To send metric data for your instance to CloudWatch in 1-minute periods, you can enable detailed monitoring on the instance, however, this comes at an additional cost.

Simply get it from the CloudWatch Metrics - You can get data from metrics. The basic monitoring data is available automatically in a 5-minute interval and detailed monitoring data is available in a 1-minute interval.

Open a support ticket with AWS - This option has been added as a distractor.

Reference:

https://aws.amazon.com/blogs/aws/new-high-resolution-custom-metrics-and-alarms-for-amazon-cloudwatch/

Question 39: **Incorrect**

The Development team at a media company is working on securing their databases.

Which of the following AWS database engines can be configured with IAM Database Authentication? (Select two)

- [x] **RDS Oracle** **(Incorrect)**
- [ ] **RDS PostGreSQL** **(Correct)**
- [ ] **RDS MySQL** **(Correct)**
- [ ] **RDS Db2**
- [x] **RDS SQL Server** **(Incorrect)**

Explanation

Correct options:

You can authenticate to your DB instance using AWS Identity and Access Management (IAM) database authentication. With this authentication method, you don't need to use a password when you connect to a DB instance. Instead, you use an authentication token. An authentication token is a unique string of characters that Amazon RDS generates on request. Each token has a lifetime of 15 minutes. You don't need to store user credentials in the database, because authentication is managed externally using IAM.

RDS MySQL - IAM database authentication works with MySQL and PostgreSQL engines for Aurora as well as MySQL, MariaDB and RDS PostgreSQL engines for RDS.

RDS PostGreSQL - IAM database authentication works with MySQL and PostgreSQL engines for Aurora as well as MySQL, MariaDB and RDS PostgreSQL engines for RDS.

Incorrect options:

RDS Oracle

RDS SQL Server

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

RDS Db2 - This option has been added as a distractor. Db2 is a family of data management products, including database servers, developed by IBM. RDS does not support Db2 database engine.

Reference:

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.IAMDBAuth.html

Question 40: **Incorrect**

A developer wants to securely store an access token that allows a transaction-processing application running on Amazon EC2 instances to authenticate and send a chat message (via the chat API) to the company's support team when an invalid transaction is detected. While minimizing management overhead, the chat API access token must be encrypted both at rest and in transit, and also be accessible from other AWS accounts.

What is the most efficient solution to address this scenario?

- ○ **Leverage AWS Systems Manager Parameter Store with an AWS KMS customer-managed key to store the access token as a SecureString parameter and configure a resource-based policy for the parameter to allow access from other accounts. Modify the IAM role of the EC2 instances with permissions to access Parameter Store. Fetch the token from Parameter Store using the `with decryption` flag and then use the decrypted access token to send the message to the chat**

- ○ **Store AWS KMS encrypted access token in a DynamoDB table and configure a resource-based policy for the DynamoDB table to allow access from other accounts. Modify the IAM role of the EC2 instances with permissions to access the DynamoDB table. Fetch the token from the Dynamodb table and then use the decrypted access token to send the message to the chat**

- ◉ **Leverage SSE-KMS to store the access token as an encrypted object on S3 and configure a resource-based policy for the S3 bucket to allow access from other accounts. Modify the IAM role of the EC2 instances with permissions to access the S3 object. Fetch the token from S3 and then use the decrypted access token to send the message to the chat** (Incorrect)

- ○ **Leverage AWS Secrets Manager with an AWS KMS customer-managed key to store the access token as a secret and configure a resource-based policy for the secret to allow access from other accounts. Modify the IAM role of the EC2 instances with permissions to access Secrets Manager. Fetch the token from Secrets Manager and then use the decrypted access token to send the message to the chat** (Correct)

Explanation

Correct option:

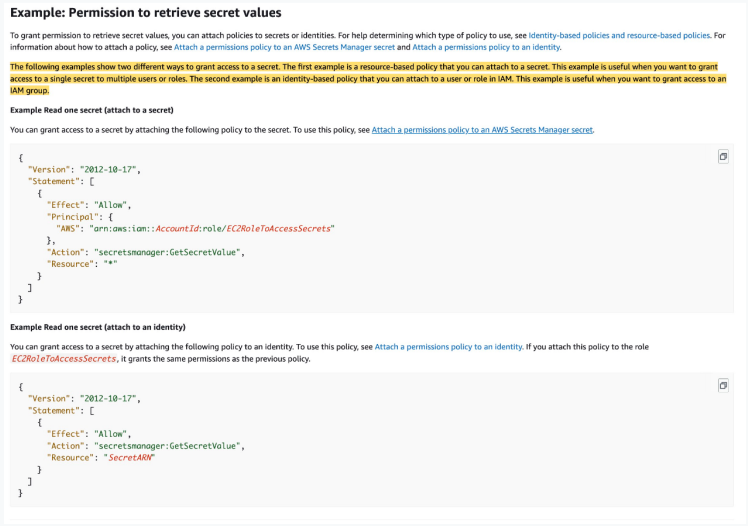
Leverage AWS Secrets Manager with an AWS KMS customer-managed key to store the access token as a secret and configure a resource-based policy for the secret to allow access from other accounts. Modify the IAM role of the EC2 instances with permissions to access Secrets Manager. Fetch the token from Secrets Manager and then use the decrypted access token to send the message to the chat

AWS Secrets Manager is an AWS service that encrypts and stores your secrets, and transparently decrypts and returns them to you in plaintext. It's designed especially to store application secrets, such as login credentials, that change periodically and should not be hard-coded or stored in plaintext in the application. In place of hard-coded credentials or table lookups, your application calls Secrets Manager.

Secrets Manager also supports features that periodically rotate the secrets associated with commonly used databases. It always encrypts newly rotated secrets before they are stored.

Secrets Manager integrates with AWS Key Management Service (AWS KMS) to encrypt every version of every secret value with a unique data key that is protected by an AWS KMS key. This integration protects your secrets under encryption keys that never leave AWS KMS unencrypted. It also enables you to set custom permissions on the KMS key and audit the operations that generate, encrypt, and decrypt the data keys that protect your secrets.

To grant permission to retrieve secret values, you can attach policies to secrets or identities.



Example: Permission to retrieve secret values

To grant permission to retrieve secret values, you can attach policies to secrets or identities. For help determining which type of policy to use, see Identity-based policies and resource-based policies. For information about how to attach a policy, see Attach a permissions policy to an AWS Secrets Manager secret and Attach a permissions policy to an identity.

The following examples show two different ways to grant access to a secret. The first example is a resource-based policy that you can attach to a secret. This example is useful when you want to grant access to a single secret to multiple users or roles. The second example is an identity-based policy that you can attach to a user or role in IAM. This example is useful when you want to grant access to an IAM group.

Example Read one secret (attach to a secret)

You can grant access to a secret by attaching the following policy to the secret. To use this policy, see Attach a permissions policy to an AWS Secrets Manager secret.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::AccountId:role/EC2RoleToAccessSecrets"
      },
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "*"
    }
  ]
}
```

Example Read one secret (attach to an identity)

You can grant access to a secret by attaching the following policy to an identity. To use this policy, see Attach a permissions policy to an identity. If you attach this policy to the role EC2RoleToAccessSecrets, it grants the same permissions as the previous policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "secretsmanager:GetSecretValue",
      "Resource": "SecretARN"
    }
  ]
}
```

via - https://docs.aws.amazon.com/secretsmanager/latest/userguide/auth-and-access_examples.html

For the given use case, you can use the resource-based policy to the secret to allow access from other accounts. Then you need to update the IAM role of the EC2 instances with permissions to access Secrets Manager which will retrieve the token from Secrets Manager and use the decrypted access token to send the message to the support team via the chat API.

Incorrect options:

Leverage AWS Systems Manager Parameter Store with an AWS KMS customer-managed key to store the access token as a SecureString parameter and configure a resource-based policy for the parameter to allow access from other accounts. Modify the IAM role of the EC2 instances with permissions to access Parameter Store. Fetch the token from Parameter Store using the `with decryption` flag and then use the decrypted access token to send the message to the chat - You cannot use a resource-based policy with a parameter in the Parameter Store. Parameter Store supports parameter policies that are available for parameters that use the advanced parameters tier. Parameter policies help you manage a growing set of parameters by allowing you to assign specific criteria to a parameter such as an expiration date or time to live. Parameter policies are especially helpful in forcing you to update or delete passwords and configuration data stored in Parameter Store, a capability of AWS Systems Manager. So this option is incorrect.

Store AWS KMS encrypted access token in a DynamoDB table and configure a resource-based policy for the DynamoDB table to allow access from other accounts. Modify the IAM role of the EC2 instances with permissions to access the DynamoDB table. Fetch the token from the Dynamodb table and then use the decrypted access token to send the message to the chat - You should note that DynamoDB does not support resource-based policies. Moreover, it's a security bad practice to keep sensitive access credentials in code, database or a flat file on a file system or object storage. Therefore, this option is incorrect.

Leverage SSE-KMS to store the access token as an encrypted object on S3 and configure a resource-based policy for the S3 bucket to allow access from other accounts. Modify the IAM role of the EC2 instances with permissions to access the S3 object. Fetch the token from S3 and then use the decrypted access token to send the message to the chat - It is considered a security bad practice to keep sensitive access credentials in code, database, or a flat file on a file system or object storage. Therefore, this option is incorrect.

References:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/auth-and-access_examples.html

https://docs.aws.amazon.com/systems-manager/latest/userguide/parameter-store-policies.html

Question 41: **Incorrect**

You are creating a mobile application that needs access to the AWS API Gateway. Users will need to register first before they can access your API and you would like the user management to be fully managed.

Which authentication option should you use for your API Gateway layer?

- (●) **Use API Gateway User Pools** **(Incorrect)**
- ( ) **Use Cognito User Pools** **(Correct)**
- ( ) **Use Lambda Authorizer**
- ( ) **Use IAM permissions with sigv4**

Explanation
Correct option:

Use Cognito User Pools - As an alternative to using IAM roles and policies or Lambda authorizers, you can use an Amazon Cognito user pool to control who can access your API in Amazon API Gateway. To use an Amazon Cognito user pool with your API, you must first create an authorizer of the COGNITO_USER_POOLS type and then configure an API method to use that authorizer. After the API is deployed, the client must first sign the user into the user pool, obtain an identity or access token for the user, and then call the API method with one of the tokens, which are typically set to the request's Authorization header. The API call succeeds only if the required token is supplied and the supplied token is valid, otherwise, the client isn't authorized to make the call because the client did not have credentials that could be authorized.

Incorrect options:

Use Lambda Authorizer- A Lambda authorizer (formerly known as a custom authorizer) is an API Gateway feature that uses a Lambda function to control access to your API. A Lambda authorizer is useful if you want to implement a custom authorization scheme that uses a bearer token authentication strategy such as OAuth or SAML, or that uses request parameters to determine the caller's identity. This won't be a fully managed user management solution but it would allow you to check for access at the AWS API Gateway level.

Use IAM permissions with sigv4 - Signature Version 4 is the process to add authentication information to AWS requests sent by HTTP. For security, most requests to AWS must be signed with an access key, which consists of an access key ID and secret access key. These two keys are commonly referred to as your security credentials. But, we cannot possibly create an IAM user for every visitor of the site, so this is where social identity providers come in to help.

Use API Gateway User Pools - This is a made-up option.

References:

https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-integrate-with-cognito.html

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

https://docs.aws.amazon.com/general/latest/gr/signature-version-4.html

Question 42: **Incorrect**

A large firm stores its static data assets on Amazon S3 buckets. Each service line of the firm has its own AWS account. For a business use case, the Finance department needs to give access to their S3 bucket's data to the Human Resources department.

Which of the below options is NOT feasible for cross-account access of S3 bucket objects?

- ○ **Use IAM roles and resource-based policies delegate access across accounts within different partitions via programmatic access only** (Correct)
- ○ **Use Resource-based policies and AWS Identity and Access Management (IAM) policies for programmatic-only access to S3 bucket objects**
- ◉ **Use Cross-account IAM roles for programmatic and console access to S3 bucket objects** (Incorrect)
- ○ **Use Access Control List (ACL) and IAM policies for programmatic-only access to S3 bucket objects**

Explanation
Correct option:

Use IAM roles and resource-based policies delegate access across accounts within different partitions via programmatic access only - This statement is incorrect and hence the right choice for this question. IAM roles and resource-based policies delegate access across accounts only within a single partition. For example, assume that you have an account in US West (N. California) in the standard `aws` partition. You also have an account in China (Beijing) in the `aws-cn` partition. You can't use an Amazon S3 resource-based policy in your account in China (Beijing) to allow access for users in your standard AWS account.

Incorrect options:

Use Resource-based policies and AWS Identity and Access Management (IAM) policies for programmatic-only access to S3 bucket objects - Use bucket policies to manage cross-account control and audit the S3 object's permissions. If you apply a bucket policy at the bucket level, you can define who can access (Principal element), which objects they can access (Resource element), and how they can access (Action element). Applying a bucket policy at the bucket level allows you to define granular access to different objects inside the bucket by using multiple policies to control access. You can also review the bucket policy to see who can access objects in an S3 bucket.

Use Access Control List (ACL) and IAM policies for programmatic-only access to S3 bucket objects - Use object ACLs to manage permissions only for specific scenarios and only if ACLs meet your needs better than IAM and S3 bucket policies. Amazon S3 ACLs allow users to define only the following permissions sets: READ, WRITE, READ_ACP, WRITE_ACP, and FULL_CONTROL. You can use only an AWS account or one of the predefined Amazon S3 groups as a grantee for the Amazon S3 ACL.

Use Cross-account IAM roles for programmatic and console access to S3 bucket objects - Not all AWS services support resource-based policies. This means that you can use cross-account IAM roles to centralize permission management when providing cross-account access to multiple services. Using cross-account IAM roles simplifies provisioning cross-account access to S3 objects that are stored in multiple S3 buckets, removing the need to manage multiple policies for S3 buckets. This method allows cross-account access to objects that are owned or uploaded by another AWS account or AWS services. If you don't use cross-account IAM roles, the object ACL must be modified.

References:

https://docs.aws.amazon.com/AmazonS3/latest/dev/example-walkthroughs-managing-access-example3.html

https://docs.aws.amazon.com/IAM/latest/UserGuide/tutorial_cross-account-with-roles.html

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_compare-resource-policies.html

https://aws.amazon.com/premiumsupport/knowledge-center/cross-account-access-s3/

Question 43: **Correct**

Your company has a three-year contract with a healthcare provider. The contract states that monthly database backups must be retained for the duration of the contract for compliance purposes. Currently, the limit on backup retention for automated backups, on Amazon Relational Database Service (RDS), does not meet your requirements.

Which of the following solutions can help you meet your requirements?

- ○ **Enable RDS Read replicas**
- ◉ **Create a cron event in CloudWatch, which triggers an AWS Lambda function that triggers the database snapshot** **(Correct)**
- ○ **Enable RDS Multi-AZ**
- ○ **Enable RDS automatic backups**

Explanation

Correct option:

Create a cron event in CloudWatch, which triggers an AWS Lambda function that triggers the database snapshot - There are multiple ways to run periodic jobs in AWS. CloudWatch Events with Lambda is the simplest of all solutions. To do this, create a CloudWatch Rule and select “Schedule” as the Event Source. You can either use a cron expression or provide a fixed rate (such as every 5 minutes). Next, select “Lambda Function” as the Target. Your Lambda will have the necessary code for snapshot functionality.

Incorrect options:

Enable RDS automatic backups - You can enable automatic backups but as of 2020, the retention period is 0 to 35 days.

Enable RDS Read replicas - Amazon RDS server's built-in replication functionality to create a special type of DB instance called a read replica from a source DB instance. Updates made to the source DB instance are asynchronously copied to the read replica. Read replicas are useful for heavy read-only data workloads. These are not suitable for the given use-case.

Enable RDS Multi-AZ - Multi-AZ allows you to create a highly available application with RDS. It does not directly help in database backups or retention periods.

Reference:

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateSnapshot.html

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

Question 44: **Correct**

An IT company has migrated to a serverless application stack on the AWS Cloud with the compute layer being implemented via Lambda functions. The engineering managers would like to actively troubleshoot any failures in the Lambda functions.

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

- (●) **The developers should insert logging statements in the Lambda function code which are then available via CloudWatch logs** **(Correct)**
- ( ) **Use CodeCommit to identify and notify any failures in the Lambda code**
- ( ) **Use CodeDeploy to identify and notify any failures in the Lambda code**
- ( ) **Use CloudWatch Events to identify and notify any failures in the Lambda code**

Explanation

Correct option:

"The developers should insert logging statements in the Lambda function code which are then available via CloudWatch logs"

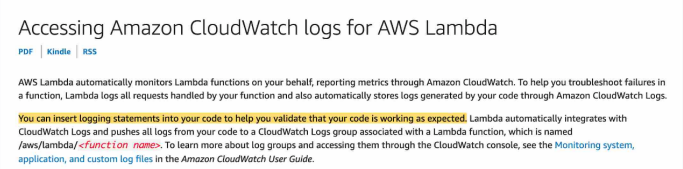
When you invoke a Lambda function, two types of error can occur. Invocation errors occur when the invocation request is rejected before your function receives it. Function errors occur when your function's code or runtime returns an error. Depending on the type of error, the type of invocation, and the client or service that invokes the function, the retry behavior, and the strategy for managing errors varies.

Lambda function failures are commonly caused by:

Permissions issues Code issues Network issues Throttling Invoke API 500 and 502 errors

You can insert logging statements into your code to help you validate that your code is working as expected. Lambda automatically integrates with CloudWatch Logs and pushes all logs from your code to a CloudWatch Logs group associated with a Lambda function, which is named `/aws/lambda/<function name>`.

Please see this note for more details:



Accessing Amazon CloudWatch logs for AWS Lambda

PDF | Kindle | RSS

AWS Lambda automatically monitors Lambda functions on your behalf, reporting metrics through Amazon CloudWatch. To help you troubleshoot failures in a function, Lambda logs all requests handled by your function and also automatically stores logs generated by your code through Amazon CloudWatch Logs.

You can insert logging statements into your code to help you validate that your code is working as expected. Lambda automatically integrates with CloudWatch Logs and pushes all logs from your code to a CloudWatch Logs group associated with a Lambda function, which is named /aws/lambda/*<function name>*. To learn more about log groups and accessing them through the CloudWatch console, see the Monitoring system, application, and custom log files in the *Amazon CloudWatch User Guide*.

via - https://docs.aws.amazon.com/lambda/latest/dg/monitoring-cloudwatchlogs.html

Incorrect options:

"Use CloudWatch Events to identify and notify any failures in the Lambda code" - Typically Lambda functions are triggered as a response to a CloudWatch Event. CloudWatch Events cannot identify and notify failures in the Lambda code.

"Use CodeCommit to identify and notify any failures in the Lambda code"

"Use CodeDeploy to identify and notify any failures in the Lambda code"

AWS CodeCommit is a fully-managed source control service that hosts secure Git-based repositories. It makes it easy for teams to collaborate on code in a secure and highly scalable ecosystem.

AWS CodeDeploy is a fully managed deployment service that automates software deployments to a variety of compute services such as Amazon EC2, AWS Fargate, AWS Lambda, and your on-premises servers.

Neither CodeCommit nor CodeDeploy can identify and notify failures in the Lambda code.

Reference:

https://docs.aws.amazon.com/lambda/latest/dg/monitoring-cloudwatchlogs.html

Question 45: **Incorrect**

A development team uses shared Amazon S3 buckets to upload files. Due to this shared access, objects in S3 buckets have different owners making it difficult to manage the objects.

As a developer associate, which of the following would you suggest to automatically make the S3 bucket owner, also the owner of all objects in the bucket, irrespective of the AWS account used for uploading the objects?

- ○ **Use S3 Object Ownership to default bucket owner to be the owner of all objects in the bucket** **(Correct)**
- ◉ **Use Bucket Access Control Lists (ACLs) to control access on S3 bucket and then define its owner** **(Incorrect)**
- ○ **Use S3 Access Analyzer to identify the owners of all objects and change the ownership to the bucket owner**
- ○ **Use S3 CORS to make the S3 bucket owner, the owner of all objects in the bucket**

Explanation
Correct option:

Use S3 Object Ownership to default bucket owner to be the owner of all objects in the bucket

S3 Object Ownership is an Amazon S3 bucket setting that you can use to control ownership of new objects that are uploaded to your buckets. By default, when other AWS accounts upload objects to your bucket, the objects remain owned by the uploading account. With S3 Object Ownership, any new objects that are written by other accounts with the bucket-owner-full-control canned access control list (ACL) automatically become owned by the bucket owner, who then has full control of the objects.

S3 Object Ownership has two settings: 1. Object writer – The uploading account will own the object. 2. Bucket owner preferred – The bucket owner will own the object if the object is uploaded with the `bucket-owner-full-control` canned ACL. Without this setting and canned ACL, the object is uploaded and remains owned by the uploading account.

Incorrect options:

Use S3 CORS to make the S3 bucket owner, the owner of all objects in the bucket - Cross-origin resource sharing (CORS) defines a way for client web applications that are loaded in one domain to interact with resources in a different domain.

Use S3 Access Analyzer to identify the owners of all objects and change the ownership to the bucket owner - Access Analyzer for S3 helps review all buckets that have bucket access control lists (ACLs), bucket policies, or access point policies that

grant public or shared access. Access Analyzer for S3 alerts you to buckets that are configured to allow access to anyone on the internet or other AWS accounts, including AWS accounts outside of your organization.

Use Bucket Access Control Lists (ACLs) to control access on S3 bucket and then define its owner - Amazon S3 access control lists (ACLs) enable you to manage access to buckets and objects. Each bucket and object has an ACL attached to it as a subresource. A bucket ACLs allow you to control access at bucket level.

None of the above features are useful for the current scenario and hence are incorrect options.

References:

https://docs.aws.amazon.com/AmazonS3/latest/userguide/about-object-ownership.html

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

Question 46: **Incorrect**

You have a workflow process that pulls code from AWS CodeCommit and deploys to EC2 instances associated with tag group ProdBuilders. You would like to configure the instances to archive no more than two application revisions to conserve disk space.

Which of the following will allow you to implement this?

- ○ **AWS CloudWatch Log Agent**
- ○ **Have a load balancer in front of your instances**
- ○ **CodeDeploy Agent** **(Correct)**
- ◉ **Integrate with AWS CodePipeline** **(Incorrect)**

Explanation
Correct option:

"CodeDeploy Agent"

Explanation
Correct option:

"CodeDeploy Agent"

The CodeDeploy agent is a software package that, when installed and configured on an instance, makes it possible for that instance to be used in CodeDeploy deployments. The CodeDeploy agent archives revisions and log files on instances. The CodeDeploy agent cleans up these artifacts to conserve disk space. You can use the :max_revisions: option in the agent configuration file to specify the number of application revisions to the archive by entering any positive integer. CodeDeploy also archives the log files for those revisions. All others are deleted, except for the log file of the last successful deployment.

More info here:

Application Revision and Log File Cleanup

The CodeDeploy agent archives revisions and log files on instances. The CodeDeploy agent cleans up these artifacts to conserve disk space.

Application revision deployment logs: You can use the **:max_revisions:** option in the agent configuration file to specify the number of application revisions to archive by entering any positive integer. CodeDeploy also archives the log files for those revisions. All others are deleted, with the exception of the log file of the last successful deployment. That log file is always retained, even if the number of failed deployments exceeds the number of retained revisions. If no value is specified, CodeDeploy retains the five most recent revisions in addition to the currently deployed revision.

CodeDeploy logs: For Amazon Linux, Ubuntu Server, and RHEL instances, the CodeDeploy agent rotates the log files under the `/var/log/aws/codedeploy-agent` folder. The log file is rotated at 00:00:00 (instance time) daily. Log files are deleted after seven days. The naming pattern for rotated log files is `codedeploy-agent.YYYYMMDD.log`.

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

Incorrect options:

AWS CloudWatch Log Agent - The CloudWatch Logs agent provides an automated way to send log data to CloudWatch Logs from Amazon EC2 instances. This is an incorrect choice for the current use case.

Integrate with AWS CodePipeline - AWS CodePipeline is a fully managed continuous delivery service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodeCommit and CodePipeline are already integrated services. CodePipeline cannot help in version control and management of archives on an EC2 instance.

Have a load balancer in front of your instances - Load Balancer helps balance incoming traffic across different EC2 instances. It is an incorrect choice for the current use case.

References:

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

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

Question 47: **Incorrect**

You have been asked by your Team Lead to enable detailed monitoring of the Amazon EC2 instances your team uses. As a Developer working on AWS CLI, which of the below command will you run?

- ○ **aws ec2 monitor-instances --instance-id i-1234567890abcdef0**
- ○ **aws ec2 run-instances --image-id ami-09092360 --monitoring State=enabled**
- ○ **aws ec2 monitor-instances --instance-ids i-1234567890abcdef0** (Correct)
- ◉ **aws ec2 run-instances --image-id ami-09092360 --monitoring Enabled=true** (Incorrect)

Explanation

Correct option:

`aws ec2 monitor-instances --instance-ids i-1234567890abcdef0` - This enables detailed monitoring for a running instance.

EC2 detailed monitoring:

```
  monitor-instances
--instance-ids <value>
[--dry-run | --no-dry-run]
[--cli-input-json <value>]
[--generate-cli-skeleton <value>]
```

Options

`--instance-ids` (list)

> The IDs of the instances.
>
> (string)

Syntax:

```
"string" "string" ...
```

`--dry-run` | `--no-dry-run` (boolean)

> Checks whether you have the required permissions for the action, without actually making the request, and provides an error response. If you have the required permissions, the error response is `DryRunOperation`. Otherwise, it is `UnauthorizedOperation`.

`--cli-input-json` (string) Performs service operation based on the JSON string provided. The JSON string follows the format provided by `--generate-cli-skeleton`. If other arguments are provided on the command line, the CLI values will override the JSON-provided values. It is not possible to pass arbitrary binary values using a JSON-provided value as the string will be taken literally.

`--generate-cli-skeleton` (string) Prints a JSON skeleton to standard output without sending an API request. If provided with no value or the value `input`, prints a sample input JSON that can be used as an argument for `--cli-input-json`. If provided with the value `output`, it validates the command inputs and returns a sample output JSON for that command.

via - https://docs.aws.amazon.com/cli/latest/reference/ec2/monitor-instances.html

Incorrect options:

`aws ec2 run-instances --image-id ami-09092360 --monitoring Enabled=true` - This syntax is used to enable detailed monitoring when launching an instance from AWS CLI.

`aws ec2 run-instances --image-id ami-09092360 --monitoring State=enabled` - This is an invalid syntax

`aws ec2 monitor-instances --instance-id i-1234567890abcdef0` - This is an invalid syntax

References:

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

https://docs.aws.amazon.com/cli/latest/reference/ec2/run-instances.html

Question 48: **Correct**

Your company hosts a static website on Amazon Simple Storage Service (S3) written in HTML5. The website targets aviation enthusiasts and it has grown a worldwide audience with hundreds of thousands of visitors accessing the website now on a monthly basis. While users in the United States have a great user experience, users from other parts of the world are experiencing slow responses and lag.

Which service can mitigate this issue?

- ○ **Use Amazon S3 Caching**
- ○ **Use Amazon ElastiCache for Redis**
- ◉ **Use Amazon CloudFront** (Correct)
- ○ **Use Amazon S3 Transfer Acceleration**

Explanation
Correct option:

Use Amazon CloudFront

Storing your static content with S3 provides a lot of advantages. But to help optimize your application's performance and security while effectively managing cost, AWS recommends that you also set up Amazon CloudFront to work with your S3 bucket to serve and protect the content. 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.

By caching your content in Edge Locations, CloudFront reduces the load on your S3 bucket and helps ensure a faster response for your users when they request content. In addition, data transfer out for content by using CloudFront is often more cost-effective than serving files directly from S3, and there is no data transfer fee from S3 to CloudFront.

A security feature of CloudFront is Origin Access Identity (OAI), which restricts access to an S3 bucket and its content to only CloudFront and operations it performs.

CloudFront Overview:

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

Incorrect options:

Use Amazon ElastiCache for Redis - Amazon ElastiCache can be used to significantly improve latency and throughput for many read-heavy application workloads (such as social networking, gaming, media sharing, and Q&A portals). ElastiCache is often used with databases that need millisecond latency. For the current scenario, we do not need a caching layer since the data load is not that heavy.

Use Amazon S3 Caching - This is a made-up option, given as a distractor.

Use Amazon S3 Transfer Acceleration - Amazon S3 Transfer Acceleration enables fast, easy, and secure transfers of files over long distances between your client and your Amazon S3 bucket. However, S3 Transfer Acceleration leverages Amazon CloudFront's globally distributed AWS Edge Locations. Each time S3 Transfer Acceleration is used to upload an object, AWS checks whether S3 Transfer Acceleration is likely to be faster than a regular Amazon S3 transfer. If it finds that S3 Transfer Acceleration might not be significantly faster, AWS shifts back to normal Amazon S3 transfer mode. So, this is not the right option for our use case.

References:

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

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

Question 49: **Correct**

A new member of your team is working on creating Dead Letter Queue (DLQ) for AWS Lambda functions.

As a Developer Associate, can you help him identify the use cases, wherein AWS Lambda will add a message into a DLQ after being processed? (Select two)

- [x] **The Lambda function invocation is asynchronous** **(Correct)**
- [ ] **The Lambda function invocation is synchronous**
- [x] **The event fails all processing attempts** **(Correct)**
- [ ] **The event has been processed successfully**
- [ ] **The Lambda function invocation failed only once but succeeded thereafter**

Explanation
Correct option:

The Lambda function invocation is asynchronous - When an asynchronous invocation event exceeds the maximum age or fails all retry attempts, Lambda discards it. Or sends it to dead-letter queue if you have configured one.

The event fails all processing attempt - A dead-letter queue acts the same as an on-failure destination in that it is used when an event fails all processing attempts or expires without being processed.

Incorrect options:

The Lambda function invocation is synchronous - When you invoke a function synchronously, Lambda runs the function and waits for a response. Queues are generally used with asynchronous invocations since queues implement the decoupling feature of various connected systems. It does not make sense to use queues when the calling code will wait on it for a response.

The event has been processed successfully - A successfully processed event is not sent to the dead-letter queue.

The event processing failed only once but succeeded thereafter - A successfully processed event is not sent to the dead-letter queue.

Reference:

https://docs.aws.amazon.com/lambda/latest/dg/invocation-async.html

Question 50: **Incorrect**

The development team at a company wants to encrypt a 111 GB object using AWS KMS.

Which of the following represents the best solution?

- ○ **Make a `GenerateDataKeyWithPlaintext` API call that returns an encrypted copy of a data key. Use a plaintext key to encrypt the data**
- ◉ **Make a `GenerateDataKeyWithoutPlaintext` API call that returns an encrypted copy of a data key. Use an encrypted key to encrypt the data** (Incorrect)
- ○ **Make a `GenerateDataKey` API call that returns a plaintext key and an encrypted copy of a data key. Use a plaintext key to encrypt the data** (Correct)
- ○ **Make an `Encrypt` API call to encrypt the plaintext data as ciphertext using a customer master key (CMK) with imported key material**

Explanation
Correct option:

Make a `GenerateDataKey` API call that returns a plaintext key and an encrypted copy of a data key. Use a plaintext key to encrypt the data - `GenerateDataKey` API, generates a unique symmetric data key for client-side encryption. This operation returns a plaintext copy of the data key and a copy that is encrypted under a customer master key (CMK) that you specify. You can use the plaintext key to encrypt your data outside of AWS KMS and store the encrypted data key with the encrypted data.

`GenerateDataKey` returns a unique data key for each request. The bytes in the plaintext key are not related to the caller or the CMK.

To encrypt data outside of AWS KMS:

1. Use the `GenerateDataKey` operation to get a data key.
2. Use the plaintext data key (in the Plaintext field of the response) to encrypt your data outside of AWS KMS. Then erase the plaintext data key from memory.
3. Store the encrypted data key (in the CiphertextBlob field of the response) with the encrypted data.

To decrypt data outside of AWS KMS:

1. Use the Decrypt operation to decrypt the encrypted data key. The operation returns a plaintext copy of the data key.
2. Use the plaintext data key to decrypt data outside of AWS KMS, then erase the plaintext data key from memory.

Incorrect options:

Make a `GenerateDataKeyWithPlaintext` API call that returns an encrypted copy of a data key. Use a plaintext key to encrypt the data - This is a made-up option, given only as a distractor.

Make an `Encrypt` API call to encrypt the plaintext data as ciphertext using a customer master key (CMK) with imported key material - `Encrypt` API is used to encrypt plaintext into ciphertext by using a customer master key (CMK). The Encrypt operation has two primary use cases:

1. To encrypt small amounts of arbitrary data, such as a personal identifier or database password, or other sensitive information.
2. To move encrypted data from one AWS Region to another.

Neither of the two is useful for the given scenario.

Make a `GenerateDataKeyWithoutPlaintext` API call that returns an encrypted copy of a data key. Use an encrypted key to encrypt the data - `GenerateDataKeyWithoutPlaintext` API, generates a unique symmetric data key. This operation returns a data key that is encrypted under a customer master key (CMK) that you specify.

`GenerateDataKeyWithoutPlaintext` is identical to the `GenerateDataKey` operation except that returns only the encrypted copy of the data key. This operation is useful for systems that need to encrypt data at some point, but not immediately. When you need to encrypt the data, you call the Decrypt operation on the encrypted copy of the key.

References:

https://docs.aws.amazon.com/kms/latest/APIReference/API_GenerateDataKey.html

https://docs.aws.amazon.com/kms/latest/APIReference/API_Encrypt.html

https://docs.aws.amazon.com/kms/latest/APIReference/API_GenerateDataKeyWithoutPlaintext.html

Question 51: **Correct**

You are working for a shipping company that is automating the creation of ECS clusters with an Auto Scaling Group using an AWS CloudFormation template that accepts cluster name as its parameters. Initially, you launch the template with input value 'MainCluster', which deployed five instances across two availability zones. The second time, you launch the template with an input value 'SecondCluster'. However, the instances created in the second run were also launched in 'MainCluster' even after specifying a different cluster name.

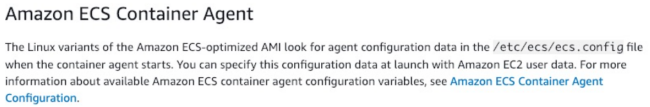
What is the root cause of this issue?

- ○ **The security groups on the EC2 instance are pointing to the wrong ECS cluster**
- ○ **The ECS agent Docker image must be re-built to connect to the other clusters**
- ○ **The EC2 instance is missing IAM permissions to join the other clusters**
- ◉ **The cluster name Parameter has not been updated in the file /etc/ecs/ecs.config during bootstrap** **(Correct)**

Explanation
Correct option:

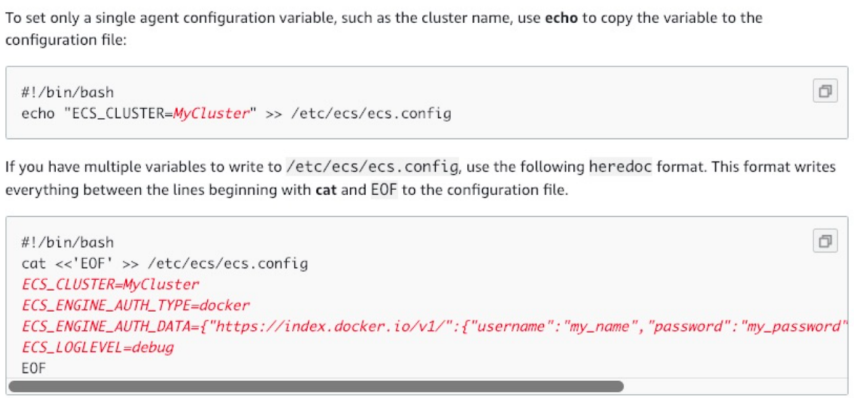
The cluster name Parameter has not been updated in the file /etc/ecs/ecs.config during bootstrap - In the ecs.config file you have to configure the parameter ECS_CLUSTER='your_cluster_name' to register the container instance with a cluster named 'your_cluster_name'.

Sample config for ECS Container Agent:



Amazon ECS Container Agent

The Linux variants of the Amazon ECS-optimized AMI look for agent configuration data in the `/etc/ecs/ecs.config` file when the container agent starts. You can specify this configuration data at launch with Amazon EC2 user data. For more information about available Amazon ECS container agent configuration variables, see Amazon ECS Container Agent Configuration.



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

Incorrect options:

The EC2 instance is missing IAM permissions to join the other clusters - EC2 instances are getting registered to the first cluster, so permissions are not an issue here and hence this statement is an incorrect choice for the current use case.

The ECS agent Docker image must be re-built to connect to the other clusters - Since the first set of instances got created from the template without any issues, there is no issue with the ECS agent here.

The security groups on the EC2 instance are pointing to the wrong ECS cluster - Security groups govern the rules about the incoming network traffic to your ECS containers. The issue here is not about user access and hence is a wrong choice for the current use case.

References:

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

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

Question 52: **Incorrect**

A company's e-commerce website is expecting hundreds of thousands of visitors on Black Friday. The marketing department is concerned that high volumes of orders might stress SQS leading to message failures. The company has approached you for the steps to be taken as a precautionary measure against the high volumes.

What step will you suggest as a Developer Associate?

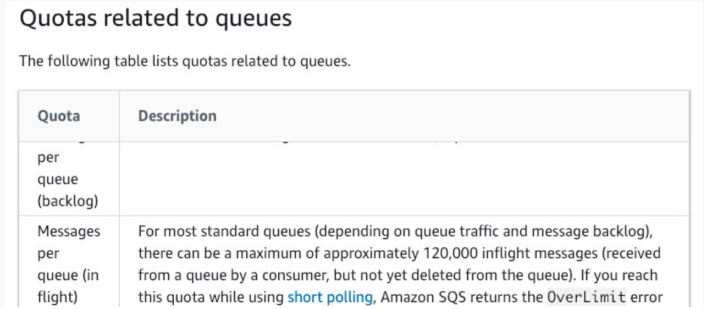
- ○ **Enable auto-scaling in the SQS queue**
- ◉ **Pre-configure the SQS queue to increase the capacity when messages hit a certain threshold** **(Incorrect)**
- ○ **Amazon SQS is highly scalable and does not need any intervention to handle the expected high volumes** **(Correct)**
- ○ **Convert the queue into FIFO ordered queue, since messages to the down system will be processed faster once they are ordered**

Explanation
Correct option:

Amazon SQS is highly scalable and does not need any intervention to handle the expected high volumes

Amazon SQS leverages the AWS cloud to dynamically scale, based on demand. SQS scales elastically with your application so you don't have to worry about capacity planning and pre-provisioning. For most standard queues (depending on queue traffic and message backlog), there can be a maximum of approximately 120,000 inflight messages (received from a queue by a consumer, but not yet deleted from the queue).

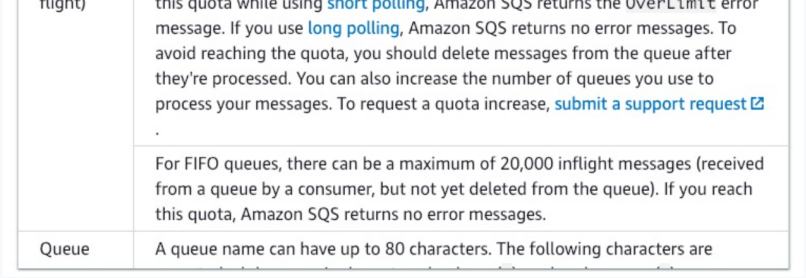
Info on Queue Quotas:



Quotas related to queues

The following table lists quotas related to queues.

Quota	Description
per queue (backlog)	
Messages per queue (in flight)	For most standard queues (depending on queue traffic and message backlog), there can be a maximum of approximately 120,000 inflight messages (received from a queue by a consumer, but not yet deleted from the queue). If you reach this quota while using short polling, Amazon SQS returns the `OverLimit` error



flight)	this quota while using short polling, Amazon SQS returns the OverLimit error message. If you use long polling, Amazon SQS returns no error messages. To avoid reaching the quota, you should delete messages from the queue after they're processed. You can also increase the number of queues you use to process your messages. To request a quota increase, submit a support request .
	For FIFO queues, there can be a maximum of 20,000 inflight messages (received from a queue by a consumer, but not yet deleted from the queue). If you reach this quota, Amazon SQS returns no error messages.
Queue	A queue name can have up to 80 characters. The following characters are

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

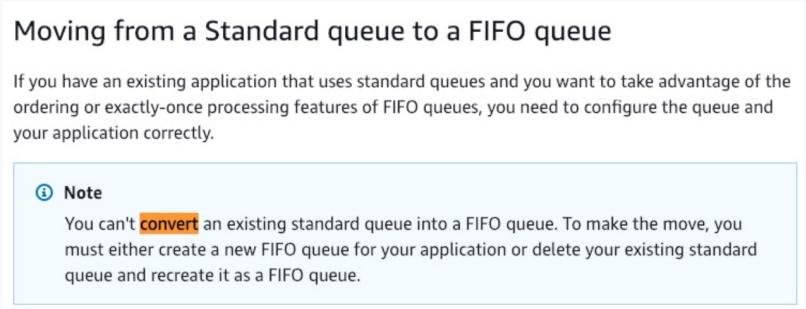
Incorrect options:

Pre-configure the SQS queue to increase the capacity when messages hit a certain threshold - This is an incorrect statement. Amazon SQS scales dynamically, automatically provisioning the needed capacity.

Enable auto-scaling in the SQS queue - SQS queues are, by definition, auto-scalable and do not need any configuration changes for auto-scaling.

Convert the queue into FIFO ordered queue, since messages to the down system will be processed faster once they are ordered - This is a wrong statement. You cannot convert an existing standard queue to FIFO queue. To make the move, you must either create a new FIFO queue for your application or delete your existing standard queue and recreate it as a FIFO queue.

Standard to FIFO queue conversion:



Moving from a Standard queue to a FIFO queue

If you have an existing application that uses standard queues and you want to take advantage of the ordering or exactly-once processing features of FIFO queues, you need to configure the queue and your application correctly.

Note

You can't convert an existing standard queue into a FIFO queue. To make the move, you must either create a new FIFO queue for your application or delete your existing standard queue and recreate it as a FIFO queue.

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

References:

References:

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

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

Question 53: **Incorrect**

Two policies are attached to an IAM user. The first policy states that the user has explicitly been denied all access to EC2 instances. The second policy states that the user has been allowed permission for EC2:Describe action.

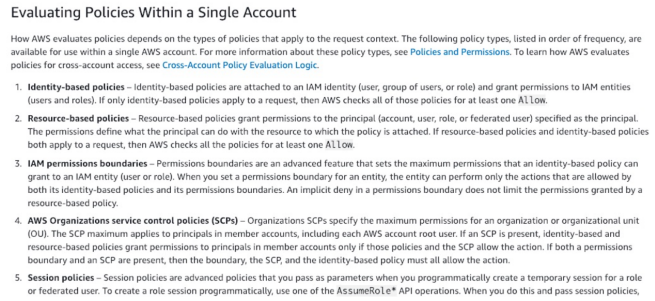
When the user tries to use 'Describe' action on an EC2 instance using the CLI, what will be the output?

- (●) **The user will get access because it has an explicit allow** **(Incorrect)**
- ( ) **The IAM user stands in an invalid state, because of conflicting policies**
- ( ) **The user will be denied access because one of the policies has an explicit deny on it** **(Correct)**
- ( ) **The order of the policy matters. If policy 1 is before 2, then the user is denied access. If policy 2 is before 1, then the user is allowed access**

Explanation
Correct option:

The user will be denied access because the policy has an explicit deny on it - User will be denied access because any explicit deny overrides the allow.

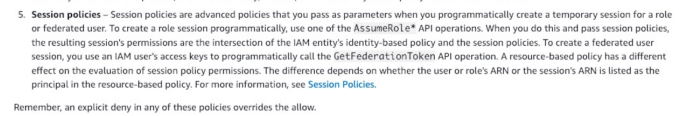
Policy Evaluation explained:



Evaluating Policies Within a Single Account

How AWS evaluates policies depends on the types of policies that apply to the request context. The following policy types, listed in order of frequency, are available for use within a single AWS account. For more information about these policy types, see Policies and Permissions. To learn how AWS evaluates policies for cross-account access, see Cross-Account Policy Evaluation Logic.

1. **Identity-based policies** – Identity-based policies are attached to an IAM identity (user, group of users, or role) and grant permissions to IAM entities (users and roles). If only identity-based policies apply to a request, then AWS checks all of those policies for at least one `Allow`.
2. **Resource-based policies** – Resource-based policies grant permissions to the principal (account, user, role, or federated user) specified as the principal. The permissions define what the principal can do with the resource to which the policy is attached. If resource-based policies and identity-based policies both apply to a request, then AWS checks all the policies for at least one `Allow`.
3. **IAM permissions boundaries** – Permissions boundaries are an advanced feature that sets the maximum permissions that an identity-based policy can grant to an IAM entity (user or role). When you set a permissions boundary for an entity, the entity can perform only the actions that are allowed by both its identity-based policies and its permissions boundaries. An implicit deny in a permissions boundary does not limit the permissions granted by a resource-based policy.
4. **AWS Organizations service control policies (SCPs)** – Organizations SCPs specify the maximum permissions for an organization or organizational unit (OU). The SCP maximum applies to principals in member accounts, including each AWS account root user. If an SCP is present, identity-based and resource-based policies grant permissions to principals in member accounts only if those policies and the SCP allow the action. If both a permissions boundary and an SCP are present, then the boundary, the SCP, and the identity-based policy must all allow the action.
5. **Session policies** – Session policies are advanced policies that you pass as parameters when you programmatically create a temporary session for a role or federated user. To create a role session programmatically, use one of the `AssumeRole*` API operations. When you do this and pass session policies,



5. **Session policies** – Session policies are advanced policies that you pass as parameters when you programmatically create a temporary session for a role or federated user. To create a role session programmatically, use one of the `AssumeRole*` API operations. When you do this and pass session policies, the resulting session's permissions are the intersection of the IAM entity's identity-based policy and the session policies. To create a federated user session, you use an IAM user's access keys to programmatically call the `GetFederationToken` API operation. A resource-based policy has a different effect on the evaluation of session policy permissions. The difference depends on whether the user or role's ARN or the session's ARN is listed as the principal in the resource-based policy. For more information, see Session Policies.

Remember, an explicit deny in any of these policies overrides the allow.

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

Incorrect options:

The IAM user stands in an invalid state, because of conflicting policies - This is an incorrect statement. Access policies can have allow and deny permissions on them and based on policy rules they are evaluated. A user account does not get invalid because of policies.

The user will get access because it has an explicit allow - As discussed above, explicit deny overrides all other permissions and hence the user will be denied access.

The order of the policy matters. If policy 1 is before 2, then the user is denied access. If policy 2 is before 1, then the user is allowed access - If policies that apply to a request include an Allow statement and a Deny statement, the Deny statement trumps the Allow statement. The request is explicitly denied.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_evaluation-logic.html

Question 54: **Incorrect**

A financial services company wants to ensure that the customer data is always kept encrypted on Amazon S3 but wants an AWS managed solution that allows full control to create, rotate and remove the encryption keys.

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

- ○ **Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3)**
- ◉ **Server-Side Encryption with Secrets Manager** **(Incorrect)**
- ○ **Server-Side Encryption with Customer Master Keys (CMKs) Stored in AWS Key Management Service (SSE-KMS)** **(Correct)**
- ○ **Server-Side Encryption with Customer-Provided Keys (SSE-C)**

Explanation
Correct option:

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

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

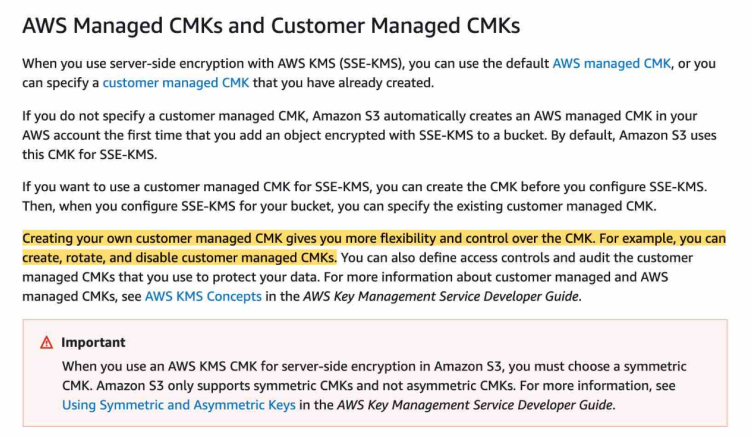
Server-Side Encryption – Request Amazon S3 to encrypt your object before saving it on disks in its data centers and then decrypt it when you download the objects.

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.

When you use server-side encryption with AWS KMS (SSE-KMS), you can use the default AWS managed CMK, or you can specify a customer-managed CMK that you have already created.

Creating your own customer-managed CMK gives you more flexibility and control over the CMK. For example, you can create, rotate, and disable customer-managed CMKs. You can also define access controls and audit the customer-managed CMKs that you use to protect your data.

Please see this note for more details:



AWS Managed CMKs and Customer Managed CMKs

When you use server-side encryption with AWS KMS (SSE-KMS), you can use the default AWS managed CMK, or you can specify a customer managed CMK that you have already created.

If you do not specify a customer managed CMK, Amazon S3 automatically creates an AWS managed CMK in your AWS account the first time that you add an object encrypted with SSE-KMS to a bucket. By default, Amazon S3 uses this CMK for SSE-KMS.

If you want to use a customer managed CMK for SSE-KMS, you can create the CMK before you configure SSE-KMS. Then, when you configure SSE-KMS for your bucket, you can specify the existing customer managed CMK.

Creating your own customer managed CMK gives you more flexibility and control over the CMK. For example, you can create, rotate, and disable customer managed CMKs. You can also define access controls and audit the customer managed CMKs that you use to protect your data. For more information about customer managed and AWS managed CMKs, see AWS KMS Concepts in the *AWS Key Management Service Developer Guide*.

Important

When you use an AWS KMS CMK for server-side encryption in Amazon S3, you must choose a symmetric CMK. Amazon S3 only supports symmetric CMKs and not asymmetric CMKs. For more information, see Using Symmetric and Asymmetric Keys in the *AWS Key Management Service Developer Guide*.

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

Incorrect options:

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, AWS encrypts the key itself with a master key that it regularly rotates. So this option is incorrect for the given use-case.

Server-Side Encryption with Customer-Provided Keys (SSE-C) - With Server-Side Encryption with Customer-Provided Keys (SSE-C), you will need to create the encryption keys as well as manage the corresponding process to rotate and remove the encryption keys. Amazon S3 manages the data encryption, as it writes to disks, as well as the data decryption when you access your objects. So this option is incorrect for the given use-case.

Server-Side Encryption with Secrets Manager - AWS Secrets Manager helps you protect secrets needed to access your applications, services, and IT resources. The service enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. You cannot combine Server-Side Encryption with Secrets Manager to create, rotate, or disable the encryption keys.

Reference:

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

Question 55: **Correct**

Your web application architecture consists of multiple Amazon EC2 instances running behind an Elastic Load Balancer with an Auto Scaling group having the desired capacity of 5 EC2 instances. You would like to integrate AWS CodeDeploy for automating application deployment. The deployment should re-route traffic from your application's original environment to the new environment.

Which of the following options will meet your deployment criteria?

- ◯ **Opt for In-place deployment**
- ◯ **Opt for Rolling deployment**
- ◉ **Opt for Blue/Green deployment** **(Correct)**
- ◯ **Opt for Immutable deployment**

Explanation
Correct option:

Opt for Blue/Green deployment - A Blue/Green deployment is used to update your applications while minimizing interruptions caused by the changes of a new application version. CodeDeploy provisions your new application version alongside the old version before rerouting your production traffic. The behavior of your deployment depends on

which compute platform you use:

1. AWS Lambda: Traffic is shifted from one version of a Lambda function to a new version of the same Lambda function.
2. Amazon ECS: Traffic is shifted from a task set in your Amazon ECS service to an updated, replacement task set in the same Amazon ECS service.
3. EC2/On-Premises: Traffic is shifted from one set of instances in the original environment to a replacement set of instances.

Incorrect options:

Opt for Rolling deployment - This deployment type is present for AWS Elastic Beanstalk and not for EC2 instances directly.

Opt for Immutable deployment - This deployment type is present for AWS Elastic Beanstalk and not for EC2 instances directly.

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

References:

https://docs.aws.amazon.com/codedeploy/latest/userguide/welcome.html#welcome-deployment-overview-blue-green

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

Question 56: **Incorrect**

A junior developer working on ECS instances terminated a container instance in Amazon Elastic Container Service (Amazon ECS) as per instructions from the team lead. But the container instance continues to appear as a resource in the ECS cluster.

As a Developer Associate, which of the following solutions would you recommend to fix this behavior?

- ○ **The container instance has been terminated with AWS CLI, whereas, for ECS instances, Amazon ECS CLI should be used to avoid any synchronization issues**
- ○ **You terminated the container instance while it was in STOPPED state, that lead to this synchronization issues** **(Correct)**
- ○ **You terminated the container instance while it was in RUNNING state, that lead to this synchronization issues**
- ◉ **A custom software on the container instance could have failed and resulted in the container hanging in an unhealthy state till restarted again** **(Incorrect)**

Explanation
Correct option:

You terminated the container instance while it was in STOPPED state, that lead to this synchronization issues - If you terminate a container instance while it is in the STOPPED state, that container instance isn't automatically removed from the cluster. You will need to deregister your container instance in the STOPPED state by using the Amazon ECS console or AWS Command Line Interface. Once deregistered, the container instance will no longer appear as a resource in your Amazon ECS cluster.

Incorrect options:

You terminated the container instance while it was in RUNNING state, that lead to this synchronization issues - This is an incorrect statement. If you terminate a container instance in the RUNNING state, that container instance is automatically removed, or deregistered, from the cluster.

The container instance has been terminated with AWS CLI, whereas, for ECS instances, Amazon ECS CLI should be used to avoid any synchronization issues - This is incorrect and has been added as a distractor.

A custom software on the container instance could have failed and resulted in the container hanging in an unhealthy state till restarted again - This is an incorrect statement. It is already mentioned in the question that the developer has terminated the

instance.

References:

https://aws.amazon.com/premiumsupport/knowledge-center/deregister-ecs-instance/

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

Question 57: **Correct**

A developer wants to enable X-Ray tracing on an on-premises Linux server running a custom application that is accessed through Amazon API Gateway.

What is the most efficient solution that requires minimal configuration?

- (●) **Install and run the X-Ray daemon on the on-premises servers to capture and relay the data to the X-Ray service** **(Correct)**
- ( ) **Install and run the X-Ray SDK on the on-premises servers to capture and relay the data to the X-Ray service**
- ( ) **Install and run the CloudWatch Unified Agent on the on-premises servers to capture and relay the X-Ray data to the X-Ray service using the PutTraceSegments API call**
- ( ) **Configure a Lambda function to analyze the incoming traffic data on the on-premises servers and then relay the X-Ray data to the X-Ray service using the PutTelemetryRecords API call**

Explanation

Correct option:

Install and run the X-Ray daemon on the on-premises servers to capture and relay the data to the X-Ray service

The AWS X-Ray daemon is a software application that listens for traffic on UDP port 2000, gathers raw segment data, and relays it to the AWS X-Ray API. The daemon works in conjunction with the AWS X-Ray SDKs and must be running so that data sent by the SDKs can reach the X-Ray service.

To run the X-Ray daemon locally, on-premises, or on other AWS services, download it, run it, and then give it permission to upload segment documents to X-Ray.

Incorrect options:

Install and run the X-Ray SDK on the on-premises servers to capture and relay the data to the X-Ray service - As mentioned above, you need to run the X-Ray daemon on

X-Ray service. So this option is incorrect.

Install and run the CloudWatch Unified Agent on the on-premises servers to capture and relay the X-Ray data to the X-Ray service using the PutTraceSegments API call - This option has been added as a distractor. CloudWatch Agent cannot relay X-Ray data to the X-Ray service using the PutTraceSegments API call.

Configure a Lambda function to analyze the incoming traffic data on the on-premises servers and then relay the X-Ray data to the X-Ray service using the PutTelemetryRecords API call - This option is incorrect as the Lambda function cannot process the X-Ray data for an on-premises instance and then relay it to the X-Ray service.

Reference:

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

Question 58: **Incorrect**

Recently, you started an online learning platform using AWS Lambda and AWS Gateway API. Your first version was successful, and you began developing new features for the second version. You would like to gradually introduce the second version by routing only 10% of the incoming traffic to the new Lambda version.

Which solution should you opt for?

- ◯ **Use AWS Lambda aliases** **(Correct)**
- ◯ **Use environment variables**
- ◯ **Deploy your Lambda in a VPC**
- ◉ **Use Tags to distinguish the different versions** **(Incorrect)**

Explanation
Correct option:

Use AWS Lambda aliases - A Lambda alias is like a pointer to a specific Lambda function version. You can create one or more aliases for your AWS Lambda function. Users can access the function version using the alias ARN. An alias can only point to a function version, not to another alias. You can update an alias to point to a new version of the function. Event sources such as Amazon S3 invoke your Lambda function. These event sources maintain a mapping that identifies the function to invoke when events

occur. If you specify a Lambda function alias in the mapping configuration, you don't need to update the mapping when the function version changes. This is the right choice for the current requirement.

Incorrect options:

Use Tags to distinguish the different versions - You can tag Lambda functions to organize them by owner, project or department. Tags are freeform key-value pairs that are supported across AWS services for use in filtering resources and adding detail to billing reports. This does not address the given use-case.

Use environment variables - You can use environment variables to store secrets securely and adjust your function's behavior without updating code. An environment variable is a pair of strings that are stored in a function's version-specific configuration. The Lambda runtime makes environment variables available to your code and sets additional environment variables that contain information about the function and invocation request. For example, you can use environment variables to point to test, development or production databases by passing it as an environment variable during runtime. This option does not address the given use-case.

Deploy your Lambda in a VPC - Amazon Virtual Private Cloud (Amazon VPC) enables you to launch AWS resources into a virtual network that you've defined. This adds another layer of security for your entire architecture. Not the right choice for the given scenario.

References:

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

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

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

Question 59: **Incorrect**

A telecom service provider stores its critical customer data on Amazon Simple Storage Service (Amazon S3).

Which of the following options can be used to control access to data stored on Amazon S3? (Select two)

- [x] **Permissions boundaries, Identity and Access Management (IAM) policies** (Incorrect)
- [x] **Bucket policies, Identity and Access Management (IAM) policies** (Correct)
- [ ] **Query String Authentication, Access Control Lists (ACLs)** (Correct)
- [ ] **Query String Authentication, Permissions boundaries**
- [ ] **IAM database authentication, Bucket policies**

Explanation
Correct options:

Bucket policies, Identity and Access Management (IAM) policies

Query String Authentication, Access Control Lists (ACLs)

Customers may use four mechanisms for controlling access to Amazon S3 resources: Identity and Access Management (IAM) policies, bucket policies, Access Control Lists (ACLs), and Query String Authentication.

IAM enables organizations with multiple employees to create and manage multiple users under a single AWS account. With IAM policies, customers can grant IAM users fine-grained control to their Amazon S3 bucket or objects while also retaining full control over everything the users do.

With bucket policies, customers can define rules which apply broadly across all requests to their Amazon S3 resources, such as granting write privileges to a subset of Amazon S3 resources. Customers can also restrict access based on an aspect of the request, such as HTTP referrer and IP address.

With ACLs, customers can grant specific permissions (i.e. READ, WRITE, FULL_CONTROL) to specific users for an individual bucket or object.

With Query String Authentication, customers can create a URL to an Amazon S3 object which is only valid for a limited time. Using query parameters to authenticate requests is useful when you want to express a request entirely in a URL. This method is also referred as presigning a URL.

Incorrect options:

Permissions boundaries, Identity and Access Management (IAM) policies

Query String Authentication, Permissions boundaries

IAM database authentication, Bucket policies

Permissions boundary - A Permissions boundary is an advanced feature for using a managed policy to set the maximum permissions that an identity-based policy can grant to an IAM entity. An entity's permissions boundary allows it to perform only the actions that are allowed by both its identity-based policies and its permissions boundaries. When you use a policy to set the permissions boundary for a user, it limits the user's permissions but does not provide permissions on its own.

IAM database authentication - IAM database authentication works with MySQL and PostgreSQL. With this authentication method, you don't need to use a password when you connect to a DB instance. Instead, you use an authentication token. It is a database authentication technique and cannot be used to authenticate for S3.

Therefore, all three options are incorrect.

References:

https://docs.aws.amazon.com/AmazonS3/latest/userguide/access-control-overview.html

https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/UsingWithRDS.IAMDBAuth.html

Question 60: **Correct**

A HealthCare mobile app uses proprietary Machine Learning algorithms to provide early diagnosis using patient health metrics. To protect this sensitive data, the development team wants to transition to a scalable user management system with log-in/sign-up functionality that also supports Multi-Factor Authentication (MFA)

Which of the following options can be used to implement a solution with the LEAST amount of development effort? (Select two)

- [ ] **Use Lambda functions and DynamoDB to create a custom solution for user management**
- [ ] **Use Lambda functions and RDS to create a custom solution for user management**
- [x] **Use Amazon Cognito to enable Multi-Factor Authentication (MFA) when users log-in** (Correct)
- [ ] **Use Amazon SNS to send Multi-Factor Authentication (MFA) code via SMS to mobile app users**

☑ **Use Amazon Cognito for user-management and facilitating the log-in/sign-up process** **(Correct)**

Explanation
Correct options:

Use Amazon Cognito for user-management and facilitating the log-in/sign-up process

Use Amazon Cognito to enable Multi-Factor Authentication (MFA) when users log-in

Amazon Cognito lets you add user sign-up, sign-in, and access control to your web and mobile apps quickly and easily. Amazon Cognito scales to millions of users and supports sign-in with social identity providers, such as Facebook, Google, and Amazon, and enterprise identity providers via SAML 2.0.

A Cognito user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito, or federate through a third-party identity provider (IdP). Whether your users sign-in directly or through a third party, all members of the user pool have a directory profile that you can access through an SDK.

Cognito user pools provide support for sign-up and sign-in services as well as security features such as multi-factor authentication (MFA).

What Is Amazon Cognito?

PDF | Kindle

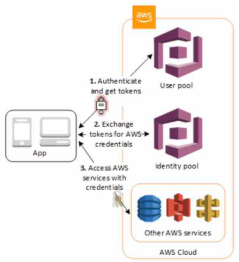
Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. Your users can sign in directly with a user name and password, or through a third party such as Facebook, Amazon, Google or Apple.

The two main components of Amazon Cognito are user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools enable you to grant your users access to other AWS services. You can use identity pools and user pools separately or together.

An Amazon Cognito user pool and identity pool used together

See the diagram for a common Amazon Cognito scenario. Here the goal is to authenticate your user, and then grant your user access to another AWS service.

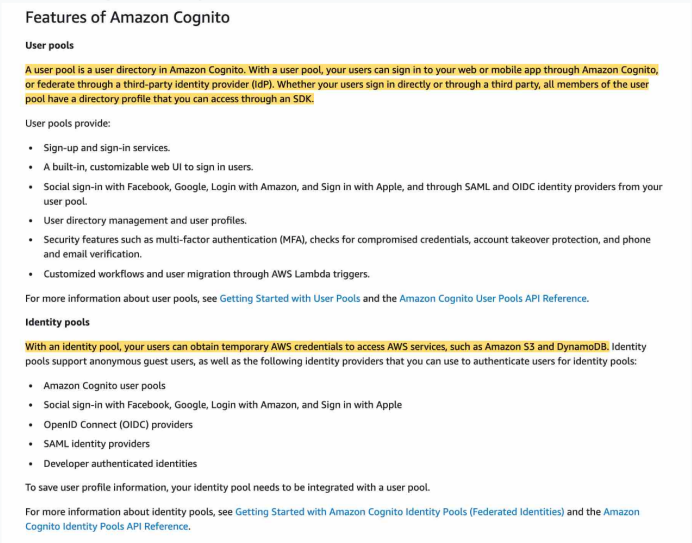
1. In the first step your app user signs in through a user pool and receives user pool tokens after a successful authentication.
2. Next, your app exchanges the user pool tokens for AWS credentials through an identity pool.
3. Finally, your app user can then use those AWS credentials to access other AWS services such as Amazon S3 or DynamoDB.



via - https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

Exam Alert:

Please review the following note to understand the differences between Cognito User Pools and Cognito Identity Pools:



Features of Amazon Cognito

User pools

A user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito, or federate through a third-party identity provider (IdP). Whether your users sign in directly or through a third party, all members of the user pool have a directory profile that you can access through an SDK.

User pools provide:

- Sign-up and sign-in services.
- A built-in, customizable web UI to sign in users.
- Social sign-in with Facebook, Google, Login with Amazon, and Sign in with Apple, and through SAML and OIDC identity providers from your user pool.
- User directory management and user profiles.
- Security features such as multi-factor authentication (MFA), checks for compromised credentials, account takeover protection, and phone and email verification.
- Customized workflows and user migration through AWS Lambda triggers.

For more information about user pools, see Getting Started with User Pools and the Amazon Cognito User Pools API Reference.

Identity pools

With an identity pool, your users can obtain temporary AWS credentials to access AWS services, such as Amazon S3 and DynamoDB. Identity pools support anonymous guest users, as well as the following identity providers that you can use to authenticate users for identity pools:

- Amazon Cognito user pools
- Social sign-in with Facebook, Google, Login with Amazon, and Sign in with Apple
- OpenID Connect (OIDC) providers
- SAML identity providers
- Developer authenticated identities

To save user profile information, your identity pool needs to be integrated with a user pool.

For more information about identity pools, see Getting Started with Amazon Cognito Identity Pools (Federated Identities) and the Amazon Cognito Identity Pools API Reference.

via - https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

Incorrect options:

Use Lambda functions and DynamoDB to create a custom solution for user management

Use Lambda functions and RDS to create a custom solution for user management

As the problem statement mentions that the solution should require the least amount of development effort, so you cannot use Lambda functions with DynamoDB or RDS to create a custom solution. So both these options are incorrect.

Use Amazon SNS to send Multi-Factor Authentication (MFA) code via SMS to mobile app users - Amazon SNS cannot be used to send MFA codes via SMS to the user's mobile devices as this functionality is only meant to be used for IAM users. An SMS (short message service) MFA device can be any mobile device with a phone number that can receive standard SMS text messages. AWS will soon end support for SMS multi-factor authentication (MFA).

Please see this for more details:

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

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

Reference:

https://docs.aws.amazon.com/cognito/latest/developerguide/what-is-amazon-cognito.html

Question 61: **Correct**

A developer from your team has configured the load balancer to route traffic equally between instances or across Availability Zones. However, Elastic Load Balancing (ELB) routes more traffic to one instance or Availability Zone than the others.

Why is this happening and how can it be fixed? (Select two)

- [ ] **After you disable an Availability Zone, the targets in that Availability Zone remain registered with the load balancer, thereby receiving random bursts of traffic**
- [x] **Instances of a specific capacity type aren't equally distributed across Availability Zones** (Correct)
- [ ] **There could be short-lived TCP connections between clients and instances**
- [ ] **For Application Load Balancers, cross-zone load balancing is disabled by default**
- [x] **Sticky sessions are enabled for the load balancer** (Correct)

Explanation

Correct option:

Sticky sessions are enabled for the load balancer - This can be the reason for potential unequal traffic routing by the load balancer. Sticky sessions are a mechanism to route requests to the same target in a target group. This is useful for servers that maintain state information in order to provide a continuous experience to clients. To use sticky sessions, the clients must support cookies.

When a load balancer first receives a request from a client, it routes the request to a target, generates a cookie named AWSALB that encodes information about the selected target, encrypts the cookie, and includes the cookie in the response to the client. The client should include the cookie that it receives in subsequent requests to the load balancer. When the load balancer receives a request from a client that contains the cookie, if sticky sessions are enabled for the target group and the request goes to the

same target group, the load balancer detects the cookie and routes the request to the same target.

If you use duration-based session stickiness, configure an appropriate cookie expiration time for your specific use case. If you set session stickiness from individual applications, use session cookies instead of persistent cookies where possible.

Instances of a specific capacity type aren't equally distributed across Availability Zones - A Classic Load Balancer with HTTP or HTTPS listeners might route more traffic to higher-capacity instance types. This distribution aims to prevent lower-capacity instance types from having too many outstanding requests. It's a best practice to use similar instance types and configurations to reduce the likelihood of capacity gaps and traffic imbalances.

A traffic imbalance might also occur if you have instances of similar capacities running on different Amazon Machine Images (AMIs). In this scenario, the imbalance of the traffic in favor of higher-capacity instance types is desirable.

Incorrect options:

There could be short-lived TCP connections between clients and instances - This is an incorrect statement. Long-lived TCP connections between clients and instances can potentially lead to unequal distribution of traffic by the load balancer. Long-lived TCP connections between clients and instances cause uneven traffic load distribution by design. As a result, new instances take longer to reach connection equilibrium. Be sure to check your metrics for long-lived TCP connections that might be causing routing issues in the load balancer.

For Application Load Balancers, cross-zone load balancing is disabled by default - This is an incorrect statement. With Application Load Balancers, cross-zone load balancing is always enabled.

After you disable an Availability Zone, the targets in that Availability Zone remain registered with the load balancer, thereby receiving random bursts of traffic - This is an incorrect statement. After you disable an Availability Zone, the targets in that Availability Zone remain registered with the load balancer. However, even though they remain registered, the load balancer does not route traffic to them.

References:

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

https://aws.amazon.com/premiumsupport/knowledge-center/elb-fix-unequal-traffic-routing/

https://docs.aws.amazon.com/elasticloadbalancing/latest/userguide/how-elastic-load-balancing-works.html#availability-zones

Question 62: **Incorrect**

As a developer, you are looking at creating a custom configuration for Amazon EC2 instances running in an Auto Scaling group. The solution should allow the group to auto-scale based on the metric of 'average RAM usage' for your Amazon EC2 instances.

Which option provides the best solution?

- ○ **Migrate your application to AWS Lambda**
- ○ **Enable detailed monitoring for EC2 and ASG to get the RAM usage data and create a CloudWatch Alarm on top of it**
- ◉ **Create a custom alarm for your ASG and make your instances trigger the alarm using PutAlarmData API** **(Incorrect)**
- ○ **Create a custom metric in CloudWatch and make your instances send data to it using PutMetricData. Then, create an alarm based on this metric** **(Correct)**

Explanation

Correct option:

Create a custom metric in CloudWatch and make your instances send data to it using PutMetricData. Then, create an alarm based on this metric - You can create a custom CloudWatch metric for your EC2 Linux instance statistics by creating a script through the AWS Command Line Interface (AWS CLI). Then, you can monitor that metric by pushing it to CloudWatch.

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

High-resolution metrics can give you more immediate insight into your application's sub-minute activity. But, every PutMetricData call for a custom metric is charged, so calling PutMetricData more often on a high-resolution metric can lead to higher charges.

Incorrect options:

Create a custom alarm for your ASG and make your instances trigger the alarm using PutAlarmData API - This solution will not work, your instances must be aware of each other's RAM utilization status, to know when the average RAM would be too high to trigger the alarm.

Enable detailed monitoring for EC2 and ASG to get the RAM usage data and create a CloudWatch Alarm on top of it - By enabling detailed monitoring you define the frequency at which the metric data has to be sent to CloudWatch, from 5 minutes to 1-minute frequency window. But, you still need to create and collect the custom metric you wish to track.

Migrate your application to AWS Lambda - This option has been added as a distractor. You cannot use Lambda for the given use-case.

References:

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

https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-instance-monitoring.html#CloudWatchAlarm

Question 63: **Incorrect**

The development team at a social media company is considering using Amazon ElastiCache to boost the performance of their existing databases.

As a Developer Associate, which of the following use-cases would you recommend as the BEST fit for ElastiCache? (Select two)

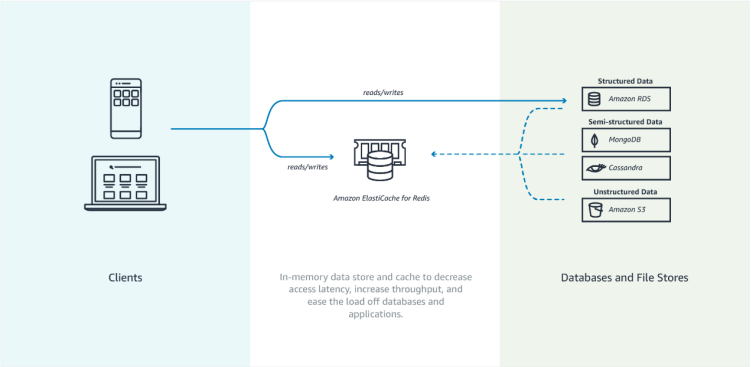
- [x] **Use ElastiCache to improve performance of Extract-Transform-Load (ETL) workloads** **(Incorrect)**
- [ ] **Use ElastiCache to improve latency and throughput for write-heavy application workloads**
- [ ] **Use ElastiCache to run highly complex JOIN queries**
- [ ] **Use ElastiCache to improve performance of compute-intensive workloads** **(Correct)**
- [x] **Use ElastiCache to improve latency and throughput for read-heavy application workloads** **(Correct)**

Explanation
Correct option:

Use ElastiCache to improve latency and throughput for read-heavy application workloads

Use ElastiCache to improve performance of compute-intensive workloads

Amazon ElastiCache allows you to run in-memory data stores in the AWS cloud. Amazon ElastiCache is a popular choice for real-time use cases like Caching, Session Stores, Gaming, Geospatial Services, Real-Time Analytics, and Queuing.



via - https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/elasticache-use-cases.html

Amazon ElastiCache can be used to significantly improve latency and throughput for many read-heavy application workloads (such as social networking, gaming, media sharing, and Q&A portals) or compute-intensive workloads (such as a recommendation engine) by allowing you to store the objects that are often read in the cache.

Overview of Amazon ElastiCache features:

Amazon ElastiCache has no upfront costs. With on-demand nodes you pay only for the resources you consume by the hour without any long-term commitments. With Reserved Nodes, you can make a low, one-time, up-front payment for each node you wish to reserve for a 1 or 3 year term. In return, you receive a significant discount off the ongoing hourly usage rate for the Node(s) you reserve.

The Amazon ElastiCache Free Usage Tier helps **new AWS customers** get started with a managed caching service in the cloud for free. Customers eligible for the AWS Free Usage tier receive 750 hours per month of a t2.micro or t3.micro node.

Pay only for what you use. There is no minimum fee. Estimate your monthly bill using the AWS Pricing Calculator.

View Detailed Pricing for Amazon ElastiCache »

Amazon ElastiCache can be used to significantly improve latency and throughput for many read-heavy application workloads (such as social networking, gaming, media sharing and Q&A portals) or compute-intensive workloads (such as a recommendation engine) by allowing you to store the objects that are often read in cache. Moreover, with Redis's support for advanced data structures, you can augment the database tier to provide features (such as leaderboard, counting, session and tracking) that are not easily achievable via databases in a cost-effective way.

Amazon ElastiCache simplifies and offloads the management, monitoring, and operation of in-memory cache environments, enabling you to focus on the differentiating parts of your applications.

Amazon ElastiCache provides:

- Support for two engines: Memcached and Redis
- Ease of management via the AWS Management Console. With a few clicks you can configure and launch cache nodes for the engine you wish to use.
- Compatibility with the specific engine protocol. This means most of the client libraries will work with the respective engines they were built for - no additional changes or tweaking required.
- Detailed monitoring statistics for the engine nodes at no extra cost via Amazon CloudWatch
- Pay only for the resources you consume based on node hours used

Amazon ElastiCache is available in all AWS regions and allows you to run your cache nodes in Amazon Virtual Private Cloud.

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

Incorrect options:

Use ElastiCache to improve latency and throughput for write-heavy application workloads - As mentioned earlier in the explanation, Amazon ElastiCache can be used to significantly improve latency and throughput for many read-heavy application workloads. Caching is not a good fit for write-heavy applications as the cache goes stale at a very fast rate.

Use ElastiCache to improve performance of Extract-Transform-Load (ETL) workloads - ETL workloads involve reading and transforming high volume data which is not a good fit for caching. You should use AWS Glue or Amazon EMR to facilitate ETL workloads.

Use ElastiCache to run highly complex JOIN queries - Complex JSON queries can be run on relational databases such as RDS or Aurora. ElastiCache is not a good fit for this use-case.

References:

https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/elasticache-use-cases.html

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

Question 64: **Incorrect**

As a Senior Developer, you manage 10 Amazon EC2 instances that make read-heavy database requests to the Amazon RDS for PostgreSQL. You need to make this architecture resilient for disaster recovery.

Which of the following features will help you prepare for database disaster recovery? (Select two)

- [ ] **Use cross-Region Read Replicas** (Correct)
- [ ] **Use RDS Provisioned IOPS (SSD) Storage in place of General Purpose (SSD) Storage**
- [x] **Use database cloning feature of the RDS DB cluster** (Incorrect)
- [x] **Enable the automated backup feature of Amazon RDS in a multi-AZ deployment that creates backups in a single AWS Region** (Correct)
- [ ] **Enable the automated backup feature of Amazon RDS in a multi-AZ deployment that creates backups across multiple Regions**

Explanation
Correct option:

Use cross-Region Read Replicas

In addition to using Read Replicas to reduce the load on your source DB instance, you can also use Read Replicas to implement a DR solution for your production DB environment. If the source DB instance fails, you can promote your Read Replica to a standalone source server. Read Replicas can also be created in a different Region than

the source database. Using a cross-Region Read Replica can help ensure that you get back up and running if you experience a regional availability issue.

Enable the automated backup feature of Amazon RDS in a multi-AZ deployment that creates backups in a single AWS Region

Amazon RDS provides high availability and failover support for DB instances using Multi-AZ deployments. Amazon RDS uses several different technologies to provide failover support. Multi-AZ deployments for MariaDB, MySQL, Oracle, and PostgreSQL DB instances use Amazon's failover technology.

The automated backup feature of Amazon RDS enables point-in-time recovery for your database instance. Amazon RDS will backup your database and transaction logs and store both for a user-specified retention period. If it's a Multi-AZ configuration, backups occur on the standby to reduce I/O impact on the primary. Automated backups are limited to a single AWS Region while manual snapshots and Read Replicas are supported across multiple Regions.

Incorrect options:

Enable the automated backup feature of Amazon RDS in a multi-AZ deployment that creates backups across multiple Regions - This is an incorrect statement. Automated backups are limited to a single AWS Region while manual snapshots and Read Replicas are supported across multiple Regions.

Use RDS Provisioned IOPS (SSD) Storage in place of General Purpose (SSD) Storage - Amazon RDS Provisioned IOPS Storage is an SSD-backed storage option designed to deliver fast, predictable, and consistent I/O performance. This storage type enhances the performance of the RDS database, but this isn't a disaster recovery option.

Use database cloning feature of the RDS DB cluster - This option has been added as a distractor. Database cloning is only available for Aurora and not for RDS.

References:

https://aws.amazon.com/rds/features/

https://aws.amazon.com/blogs/database/implementing-a-disaster-recovery-strategy-with-amazon-rds/

Question 65: **Incorrect**

An application runs on an EC2 instance and processes orders on a nightly basis. This EC2 instance needs to access the orders that are stored in S3.

How would you recommend the EC2 instance access the orders securely?

- ○ **Create an S3 bucket policy that authorises public access**
- ○ **Use an IAM role** **(Correct)**
- ◉ **Create an IAM programmatic user and store the access key and secret access key on the EC2 `~/.aws/credentials` file.** **(Incorrect)**
- ○ **Use EC2 User Data**

Explanation
Correct option:

Use an IAM role

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

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

This is the most secure option as the role assigned to EC2 can be used to access S3 without storing any credentials onto the EC2 instance.

Incorrect options:

Create an IAM programmatic user and store the access key and secret access key on the EC2 `~/.aws/credentials` file. - While this would work, this is highly insecure as anyone gaining access to the EC2 instance would be able to steal the credentials stored in that file.

Use EC2 User Data - EC2 User Data is used to run bootstrap scripts at an instance's first launch. This option is not the right fit for the given use-case.

Create an S3 bucket policy that authorizes public access - While this would work, it would allow anyone to access your S3 bucket files. So this option is ruled out.

Reference:

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