Question 1: **Correct**

As an AWS Certified Developer Associate, you are given a document written in YAML that represents the architecture of a serverless application. The first line of the document contains `Transform: 'AWS::Serverless-2016-10-31'`.

What does the `Transform` section in the document represent?

- ◯ **Presence of `Transform` section indicates it is a CloudFormation Parameter**
- ◉ **Presence of `Transform` section indicates it is a Serverless Application Model (SAM) template** **(Correct)**
- ◯ **It represents an intrinsic function**
- ◯ **It represents a Lambda function definition**

Explanation
Correct option:

AWS CloudFormation template is a JSON- or YAML-formatted text file that describes your AWS infrastructure. Templates include several major sections. The "Resources" section is the only required section. The optional "Transform" section specifies one or more macros that AWS CloudFormation uses to process your template.

The AWS Serverless Application Model (SAM) is an open-source framework for building serverless applications. It provides shorthand syntax to express functions, APIs, databases, and event source mappings. With just a few lines per resource, you can define the application you want and model it using YAML.

Presence of 'Transform' section indicates it is a Serverless Application Model (SAM) template - The AWS::Serverless transform, which is a macro hosted by AWS CloudFormation, takes an entire template written in the AWS Serverless Application Model (AWS SAM) syntax and transforms and expands it into a compliant AWS CloudFormation template. So, presence of "Transform" section indicates, the document is a SAM template.

Sample CloudFormation YAML template:

YAML

The following example shows a YAML-formatted template fragment.

```
---
AWSTemplateFormatVersion: "version date"

Description:
  String

Metadata:
  template metadata

Parameters:
  set of parameters

Mappings:
  set of mappings

Conditions:
  set of conditions

Transform:
  set of transforms

Resources:
  set of resources

Outputs:
  set of outputs
```

via - https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-anatomy.html

Incorrect options:

It represents a Lambda function definition - Lambda function is created using "AWS::Lambda::Function" resource and has no connection to 'Transform' section.

It represents an intrinsic function - Intrinsic Functions in templates are used to assign values to properties that are not available until runtime. They usually start with `Fn::` or `!`. Example: `!Sub` or `Fn::Sub`.

Presence of 'Transform' section indicates it is a CloudFormation Parameter - CloudFormation parameters are part of `Parameters` block of the template, like so:

Parameters in YAML:

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-

Question 2: **Incorrect**

A company is creating a gaming application that will be deployed on mobile devices. The application will send data to a Lambda function-based RESTful API. The application will assign each API request a unique identifier. The volume of API requests from the application can randomly vary at any given time of day. During request throttling, the application might need to retry requests. The API must be able to address duplicate requests without inconsistencies or data loss.

Which of the following would you recommend to handle these requirements?

- (●) **Persist the unique identifier for each request in an ElastiCache for Memcached cache. Change the Lambda function to check the cache for the identifier before processing the request** **(Incorrect)**
- ( ) **Persist the unique identifier for each request in an RDS MySQL table. Change the Lambda function to check the table for the identifier before processing the request**
- ( ) **Persist the unique identifier for each request in a DynamoDB table. Change the Lambda function to check the table for the identifier before processing the request** **(Correct)**
- ( ) **Persist the unique identifier for each request in a DynamoDB table. Change the Lambda function to send a client error response when the function receives a duplicate request**

Explanation

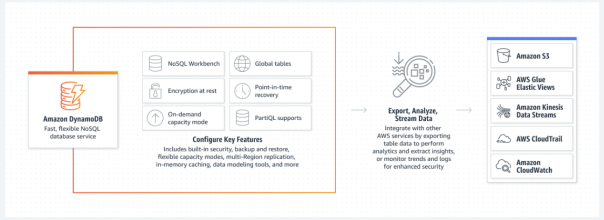
Correct option:

Persist the unique identifier for each request in a DynamoDB table. Change the Lambda function to check the table for the identifier before processing the request

DynamoDB is a fully managed, serverless, key-value NoSQL database designed to run high-performance applications at any scale. DynamoDB offers built-in security, continuous backups, automated multi-Region replication, in-memory caching, and data import and export tools. On-demand backup and restore allows you to create full backups of your DynamoDB. Point-in-time recovery (PITR) helps protect your DynamoDB tables from accidental write or delete operations. PITR provides continuous backups of your DynamoDB table data, and you can restore that table to any point in time up to the second during the preceding 35 days.

These features ensure that there is no data loss for the application, thereby meeting a key requirement for the given use case. The solution should also be able to address any duplicate requests without inconsistencies, so the Lambda function should be changed to inspect the table for the given identifier and process the request only if the identifier is unique.

DynamoDB Overview:



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

DynamoDB

Incorrect options:

Persist the unique identifier for each request in an ElastiCache for Memcached cache. Change the Lambda function to check the cache for the identifier before processing the request - Memcached is designed for simplicity and it does not offer any snapshot or replication features. This can lead to data loss for applications. Therefore, this option is not the right fit for the given use case.

	Memcached	Redis
Sub-millisecond latency	Yes	Yes
Developer ease of use	Yes	Yes
Data partitioning	Yes	Yes
Support for a broad set of programming languages	Yes	Yes
Advanced data structures	-	Yes
Multithreaded architecture	Yes	-
Snapshots	-	Yes
Replication	-	Yes
Transactions	-	Yes
Pub/Sub	-	Yes
Lua scripting	-	Yes
Geospatial support	-	Yes

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

Persist the unique identifier for each request in an RDS MySQL table. Change the Lambda function to check the table for the identifier before processing the request - DynamoDB is a better fit than RDS MySQL to handle massive traffic spikes for write requests. DynamoDB is a key-value and document database that supports tables of virtually any size with horizontal scaling. DynamoDB scales to more than 10 trillion requests per day and with tables that have more than ten million read and write requests per second and petabytes of data storage. DynamoDB can be used to build applications that need consistent single-digit millisecond performance. MySQL RDS can be scaled vertically, however, it cannot match the performance benefits offered by DynamoDB for the given use case.

Persist the unique identifier for each request in a DynamoDB table. Change the Lambda function to send a client error response when the function receives a duplicate request - The solution should be able to address any duplicates without any inconsistencies. If Lambda sends a client error response upon receiving a duplicate request, it represents an inconsistent response. So this option is incorrect.

References:

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

https://aws.amazon.com/elasticache/redis-vs-memcached/

Question 3: **Correct**

An e-commerce company has developed an API that is hosted on Amazon ECS. Variable traffic spikes on the application are causing order processing to take too long. The application processes orders using Amazon SQS queues. The `ApproximateNumberOfMessagesVisible` metric spikes at very high values throughout the day which triggers the CloudWatch alarm. Other ECS metrics for the API containers are well within limits.

As a Developer Associate, which of the following will you recommend for improving performance while keeping costs low?

- ◯ **Use Docker swarm**
- ◉ **Use backlog per instance metric with target tracking scaling policy** **(Correct)**
- ◯ **Use ECS service scheduler**
- ◯ **Use ECS step scaling policy**

Explanation
Correct option:

Use backlog per instance metric with target tracking scaling policy - If you use a target tracking scaling policy based on a custom Amazon SQS queue metric, dynamic scaling can adjust to the demand curve of your application more effectively.

The issue with using a CloudWatch Amazon SQS metric like `ApproximateNumberOfMessagesVisible` for target tracking is that the number of messages in the queue might not change proportionally to the size of the Auto Scaling group that processes messages from the queue. That's because the number of messages in your SQS queue does not solely define the number of instances needed. The number of instances in your Auto Scaling group can be driven by multiple factors, including how long it takes to process a message and the acceptable amount of latency (queue delay).

The solution is to use a backlog per instance metric with the target value being the acceptable backlog per instance to maintain. You can calculate these numbers as follows:

Backlog per instance: To calculate your backlog per instance, start with the ApproximateNumberOfMessages queue attribute to determine the length of the SQS queue (number of messages available for retrieval from the queue). Divide that number by the fleet's running capacity, which for an Auto Scaling group is the number of instances in the InService state, to get the backlog per instance.

Acceptable backlog per instance: To calculate your target value, first determine what your application can accept in terms of latency. Then, take the acceptable latency value and divide it by the average time that an EC2 instance takes to process a message.

To illustrate with an example, let's say that the current ApproximateNumberOfMessages is 1500 and the fleet's running capacity is 10. If the average processing time is 0.1 seconds for each message and the longest acceptable latency is 10 seconds, then the acceptable backlog per instance is 10 / 0.1, which equals 100. This means that 100 is the target value for your target tracking policy. If the backlog per instance is currently at 150 (1500 / 10), your fleet scales out, and it scales out by five instances to maintain proportion to the target value.

Incorrect options:

Incorrect options:

Use Docker swarm - A Docker swarm is a container orchestration tool, meaning that it allows the user to manage multiple containers deployed across multiple host machines. A swarm consists of multiple Docker hosts which run in swarm mode and act as managers (to manage membership and delegation) and workers (which run swarm services).

Use ECS service scheduler - Amazon ECS provides a service scheduler (for long-running tasks and applications), the ability to run tasks manually (for batch jobs or single run tasks), with Amazon ECS placing tasks on your cluster for you. You can specify task placement strategies and constraints that allow you to run tasks in the configuration you choose, such as spread out across Availability Zones. It is also possible to integrate with custom or third-party schedulers.

Use ECS step scaling policy - Although Amazon ECS Service Auto Scaling supports using Application Auto Scaling step scaling policies, AWS recommends using target tracking scaling policies instead. For example, if you want to scale your service when CPU utilization falls below or rises above a certain level, create a target tracking scaling policy based on the CPU utilization metric provided by Amazon ECS.

With step scaling policies, you create and manage the CloudWatch alarms that trigger the scaling process. If the target tracking alarms don't work for your use case, you can use step scaling. You can also use target tracking scaling with step scaling for an advanced scaling policy configuration. For example, you can configure a more aggressive response when utilization reaches a certain level.

Step Scaling scales your cluster on various lengths of steps based on different ranges of thresholds. Target tracking on the other hand intelligently picks the smart lengths needed for the given configuration.

References:

https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-using-sqs-queue.html

Question 4: **Correct**

A developer is testing Amazon Simple Queue Service (SQS) queues in a development environment. The queue along with all its contents has to be deleted after testing.

Which SQS API should be used for this requirement?

- ○ **PurgeQueue**
- ◉ **DeleteQueue** **(Correct)**
- ○ **RemovePermission**
- ○ **RemoveQueue**

Explanation
Correct option:

DeleteQueue - Deletes the queue specified by the QueueUrl, regardless of the queue's contents. When you delete a queue, any messages in the queue are no longer available.

When you delete a queue, the deletion process takes up to 60 seconds. Requests you send involving that queue during the 60 seconds might succeed. For example, a SendMessage request might succeed, but after 60 seconds the queue and the message you sent no longer exist.

When you delete a queue, you must wait at least 60 seconds before creating a queue with the same name.

Incorrect options:

PurgeQueue - Deletes the messages in a queue specified by the QueueURL parameter. When you use the PurgeQueue action, you can't retrieve any messages deleted from a queue. The queue however remains.

RemoveQueue - This is an invalid option, given only as a distractor.

RemovePermission - Revokes any permissions in the queue policy that matches the specified Label parameter.

Reference:

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

Question 5: **Correct**

A developer has an application that stores data in an Amazon S3 bucket. The application uses an HTTP API to store and retrieve objects. When the PutObject API operation adds objects to the S3 bucket the developer must encrypt these objects at rest by using server-side encryption with Amazon S3-managed keys (SSE-S3).

Which solution will guarantee that any upload request without the mandated encryption is not processed?

- ○ **Invoke the PutObject API operation and set the `x-amz-server-side-encryption` header as `aws:kms`. Use an S3 bucket policy to deny permission to upload an object unless the request has this header**
- ○ **Set the encryption key for SSE-S3 in the HTTP header of every request. Use an S3 bucket policy to deny permission to upload an object unless the request has this header**
- ◉ **Invoke the PutObject API operation and set the `x-amz-server-side-encryption` header as `AES256`. Use an S3 bucket policy to deny permission to upload an object unless the request has this header** (Correct)
- ○ **Invoke the PutObject API operation and set the `x-amz-server-side-encryption` header as `sse:s3`. Use an S3 bucket policy to deny permission to upload an object unless the request has this header**

Explanation
Correct option:

Invoke the PutObject API operation and set the `x-amz-server-side-encryption` header as `AES256`. Use an S3 bucket policy to deny permission to upload an object unless the request has this header

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

You can use the following bucket policy to deny permissions to upload an object unless the request includes the x-amz-server-side-encryption header to request server-side encryption using SSE-S3:

encryption using SSE-S3:

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

Incorrect options:

Invoke the PutObject API operation and set the `x-amz-server-side-encryption` header as `aws:kms`. Use an S3 bucket policy to deny permission to upload an object unless the request has this header - As mentioned above, you need to use `AES256` rather than `aws:kms` for the given use case. `aws:kms` is used when you want to use server-side encryption with AWS KMS (SSE-KMS).

Invoke the PutObject API operation and set the `x-amz-server-side-encryption` header as `sse:s3`. Use an S3 bucket policy to deny permission to upload an object unless the request has this header - This is a made-up option as the `x-amz-server-side-encryption` header has no such value as `sse:s3`.

Set the encryption key for SSE-S3 in the HTTP header of every request. Use an S3 bucket policy to deny permission to upload an object unless the request has this header - This option has been added as a distractor. For SSE-S3, Amazon S3 encrypts each object with a unique key. As an additional safeguard, it encrypts the key itself with a key that it rotates regularly. The encryption key for SSE-S3 encryption key cannot be accessed.

References:

https://docs.aws.amazon.com/AmazonS3/latest/API/API_PutObject.html

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

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

Question 6: **Incorrect**

An organization has offices across multiple locations and the technology team has configured an Application Load Balancer across targets in multiple Availability Zones. The team wants to analyze the incoming requests for latencies and the client's IP address patterns.

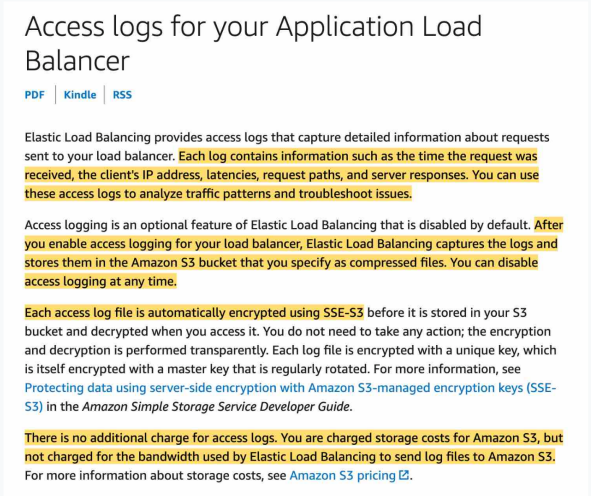
Which feature of the Load Balancer will help collect the required information?

- ○ **CloudTrail logs**
- ◉ **ALB request tracing** (Incorrect)
- ○ **CloudWatch metrics**
- ○ **ALB access logs** (Correct)

Explanation
Correct option:

ALB access logs - Elastic Load Balancing provides access logs that capture detailed information about requests sent to your load balancer. Each log contains information such as the time the request was received, the client's IP address, latencies, request paths, and server responses. You can use these access logs to analyze traffic patterns and troubleshoot issues. Access logging is an optional feature of Elastic Load Balancing that is disabled by default.

Access logs for your Application Load Balancer:



Access logs for your Application Load Balancer

PDF | Kindle | RSS

Elastic Load Balancing provides access logs that capture detailed information about requests sent to your load balancer. Each log contains information such as the time the request was received, the client's IP address, latencies, request paths, and server responses. You can use these access logs to analyze traffic patterns and troubleshoot issues.

Access logging is an optional feature of Elastic Load Balancing that is disabled by default. After you enable access logging for your load balancer, Elastic Load Balancing captures the logs and stores them in the Amazon S3 bucket that you specify as compressed files. You can disable access logging at any time.

Each access log file is automatically encrypted using SSE-S3 before it is stored in your S3 bucket and decrypted when you access it. You do not need to take any action; the encryption and decryption is performed transparently. Each log file is encrypted with a unique key, which is itself encrypted with a master key that is regularly rotated. For more information, see Protecting data using server-side encryption with Amazon S3-managed encryption keys (SSE-S3) in the *Amazon Simple Storage Service Developer Guide*.

There is no additional charge for access logs. You are charged storage costs for Amazon S3, but not charged for the bandwidth used by Elastic Load Balancing to send log files to Amazon S3. For more information about storage costs, see Amazon S3 pricing.

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

Incorrect options:

CloudTrail logs - Elastic Load Balancing is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Elastic Load Balancing. CloudTrail captures all API calls for Elastic Load Balancing as events. You can use AWS CloudTrail to capture detailed information about the calls made to the Elastic Load Balancing API and store them as log files in Amazon S3. You can use these CloudTrail logs to determine which API calls were made, the source IP address where the API call came from, who made the call, when the call was made, and so on.

CloudWatch metrics - Elastic Load Balancing publishes data points to Amazon CloudWatch for your load balancers and your targets. CloudWatch enables you to retrieve statistics about those data points as an ordered set of time-series data, known as metrics. You can use metrics to verify that your system is performing as expected. This is the right feature if you wish to track a certain metric.

ALB request tracing - You can use request tracing to track HTTP requests. The load balancer adds a header with a trace identifier to each request it receives. Request tracing will not help you to analyze latency specific data.

Reference:

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

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

Question 7: **Correct**

A development team at a social media company uses AWS Lambda for its serverless stack on AWS Cloud. For a new deployment, the Team Lead wants to send only a certain portion of the traffic to the new Lambda version. In case the deployment goes wrong, the solution should also support the ability to roll back to a previous version of the Lambda function, with MIMINUM downtime for the application.

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

- ○ **Set up the application to use an alias that points to the current version. Deploy the new version of the code and configure alias to send all users to this new version. If the deployment goes wrong, reset the alias to point to the current version**
- ◉ **Set up the application to use an alias that points to the current version. Deploy the new version of the code and configure the alias to send 10% of the users to this new version. If the deployment goes wrong, reset the alias to point all traffic to the current version** **(Correct)**
- ○ **Set up the application to have multiple alias of the Lambda function. Deploy the new version of the code. Configure a new alias that points to the current alias of the Lambda function for handling 10% of the traffic. If the deployment goes wrong, reset the new alias to point all traffic to the most recent working alias of the Lambda function**
- ○ **Set up the application to directly deploy the new Lambda version. If the deployment goes wrong, reset the application back to the current version using the version number in the ARN**

Explanation
Correct option:

Set up the application to use an alias that points to the current version. Deploy the new version of the code and configure the alias to send 10% of the users to this new version. If the deployment goes wrong, reset the alias to point all traffic to the current version

You can use versions to manage the deployment of your AWS Lambda functions. For example, you can publish a new version of a function for beta testing without affecting users of the stable production version. You can change the function code and settings only on the unpublished version of a function. When you publish a version, the code and most of the settings are locked to ensure a consistent experience for users of that version.

You can create one or more aliases for your AWS Lambda function. A Lambda alias is like a pointer to a specific Lambda function version. You can use routing configuration on an alias to send a portion of traffic to a Lambda function version. For example, you can reduce the risk of deploying a new version by configuring the alias to send most of the traffic to the existing version, and only a small percentage of traffic to the new version.

AWS Lambda function versions

PDF | Kindle | RSS

You can use versions to manage the deployment of your AWS Lambda functions. For example, you can publish a new version of a function for beta testing without affecting users of the stable production version.

The system creates a new version of your Lambda function each time that you publish the function. The new version is a copy of the unpublished version of the function. The function version includes the following information:

- The function code and all associated dependencies.
- The Lambda runtime that executes the function.
- All of the function settings, including the environment variables.
- A unique Amazon Resource Name (ARN) to identify this version of the function.

You can change the function code and settings only on the unpublished version of a function. When you publish a version, the code and most of the settings are locked to ensure a consistent experience for users of that version. For more information about configuring function settings, see Configuring functions in the AWS Lambda console.

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

Using aliases

Each alias has a unique 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.

In a resource policy, you can grant permissions for event sources to use your Lambda function. If you specify an alias ARN in the policy, you don't need to update the policy when the function version changes.

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

Alias routing configuration

Use routing configuration on an alias to send a portion of traffic to a second function version. For example, you can reduce the risk of deploying a new version by configuring the alias to send most of the traffic to the existing version, and only a small percentage of traffic to the new version.

You can point an alias to a maximum of two Lambda function versions. The versions must meet the following criteria:

- Both versions must have the same IAM execution role.
- Both versions must have the same dead-letter queue configuration, or no dead-letter queue configuration.
- Both versions must be published. The alias cannot point to `$LATEST`.

To configure routing on an alias

1. Open the Lambda console Functions page.
2. Choose a function.
3. Verify that the function has at least two published versions. To do this, choose **Qualifiers** and then choose **Versions** to display the list of versions. If you need to create additional versions, follow the instructions in AWS Lambda function versions.
4. On the **Actions** menu, choose **Create alias**.
5. In the **Create a new alias** window, enter a value for **Name**, optionally enter a value for **Description**, and choose the **Version** of the Lambda function that the alias references.
6. Under **Additional version**, specify the following items:
   a. Choose the second Lambda function version.
   b. Enter a weight value for the function. *Weight* is the percentage of traffic that is assigned to that version when the alias is invoked. The first version receives the residual weight. For example, if you specify 10 percent to **Additional version**, the first version is assigned 90 percent automatically.
7. Choose **Create**.

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

Incorrect options:

Set up the application to use an alias that points to the current version. Deploy the new version of the code and configure alias to send all users to this new version. If the deployment goes wrong, reset the alias to point to the current version - In this case, the application uses an alias to send all traffic to the new version which does not meet the requirement of sending only a certain portion of the traffic to the new Lambda version. In addition, if the deployment goes wrong, the application would see a downtime. Hence this option is incorrect.

Set up the application to directly deploy the new Lambda version. If the deployment goes wrong, reset the application back to the current version using the version number in the ARN - In this case, the application sends all traffic to the new version which does not meet the requirement of sending only a certain portion of the traffic to the new Lambda version. In addition, if the deployment goes wrong, the application would see a downtime. Hence this option is incorrect.

Set up the application to have multiple alias of the Lambda function. Deploy the new version of the code. Configure a new alias that points to the current alias of the Lambda function for handling 10% of the traffic. If the deployment goes wrong, reset the new alias to point all traffic to the most recent working alias of the Lambda function - This option has been added as a distractor. The alias for a Lambda function can only point to a Lambda function version. It cannot point to another alias.

References:

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

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

Question 8: **Correct**

Your company has stored all application secrets in SSM Parameter Store. The audit team has requested to get a report to better understand when and who has issued API calls against SSM Parameter Store.

Which of the following options can be used to produce your report?

- ○ **Use SSM Parameter Store Access Logs in CloudWatch Logs to get a record of actions taken by a user**
- ○ **Use SSM Parameter Store List feature to get a record of actions taken by a user**
- ◉ **Use AWS CloudTrail to get a record of actions taken by a user** **(Correct)**
- ○ **Use SSM Parameter Store Access Logs in S3 to get a record of actions taken by a user**

Explanation

Correct option:

Use AWS CloudTrail to get a record of actions taken by a user

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, Amazon Machine Image (AMI) IDs, and license codes as parameter values. You can store values as plain text or encrypted data.

AWS CloudTrail provides a record of actions taken by a user, role, or an AWS service in Systems Manager. Using the information collected by AWS CloudTrail, you can determine the request that was made to Systems Manager, the IP address from which the request was made, who made the request, when it was made, and additional details.

Incorrect options:

Use SSM Parameter Store List feature to get a record of actions taken by a user - This option has been added as a distractor.

Use SSM Parameter Store Access Logs in CloudWatch Logs to get a record of actions taken by a user - CloudWatch Logs can be integrated but that will not help determine who issued API calls.

Use SSM Parameter Store Access Logs in S3 to get a record of actions taken by a user - S3 Access Logs can be integrated but that will not help determine who issued API calls.

Question 9: **Correct**

A development team has configured inbound traffic for the relevant ports in both the Security Group of the EC2 instance as well as the Network Access Control List (NACL) of the subnet for the EC2 instance. The team is, however, unable to connect to the service running on the Amazon EC2 instance.

As a developer associate, which of the following will you recommend to fix this issue?

- ○ **Rules associated with Network ACLs should never be modified from the command line. An attempt to modify rules from the command line blocks the rule and results in an erratic behavior**
- ◉ **Security Groups are stateful, so allowing inbound traffic to the necessary ports enables the connection. Network ACLs are stateless, so you must allow both inbound and outbound traffic** **(Correct)**
- ○ **IAM Role defined in the Security Group is different from the IAM Role that is given access in the Network ACLs**
- ○ **Network ACLs are stateful, so allowing inbound traffic to the necessary ports enables the connection. Security Groups are stateless, so you must allow both inbound and outbound traffic**

Explanation
Correct option:

Security Groups are stateful, so allowing inbound traffic to the necessary ports enables the connection. Network ACLs are stateless, so you must allow both inbound and outbound traffic - Security groups are stateful, so allowing inbound traffic to the necessary ports enables the connection. Network ACLs are stateless, so you must allow both inbound and outbound traffic.

To enable the connection to a service running on an instance, the associated network ACL must allow both inbound traffic on the port that the service is listening on as well as allow outbound traffic from ephemeral ports. When a client connects to a server, a random port from the ephemeral port range (1024-65535) becomes the client's source port.

The designated ephemeral port then becomes the destination port for return traffic from the service, so outbound traffic from the ephemeral port must be allowed in the network ACL.

By default, network ACLs allow all inbound and outbound traffic. If your network ACL is more restrictive, then you need to explicitly allow traffic from the ephemeral port range.

If you accept traffic from the internet, then you also must establish a route through an internet gateway. If you accept traffic over VPN or AWS Direct Connect, then you must establish a route through a virtual private gateway.

Incorrect options:

Network ACLs are stateful, so allowing inbound traffic to the necessary ports enables the connection. Security Groups are stateless, so you must allow both inbound and outbound traffic - This is incorrect as already discussed.

IAM Role defined in the Security Group is different from the IAM Role that is given access in the Network ACLs - This is a made-up option and just added as a distractor.

Rules associated with Network ACLs should never be modified from the command line. An attempt to modify rules from the command line blocks the rule and results in an erratic behavior - This option is a distractor. AWS does not support modifying rules of Network ACLs from the command line tool.

Question 10: **Correct**

You're a developer working on a large scale order processing application. After developing the features, you commit your code to AWS CodeCommit and begin building the project with AWS CodeBuild before it gets deployed to the server. The build is taking too long and the error points to an issue resolving dependencies from a third-party. You would like to prevent a build running this long in the future for similar underlying reasons.

Which of the following options represents the best solution to address this use-case?

- ○ **Use AWS Lambda**
- ◉ **Enable CodeBuild timeouts** **(Correct)**
- ○ **Use AWS CloudWatch Events**
- ○ **Use VPC Flow Logs**

Explanation

Correct option:

Enable CodeBuild timeouts

A build represents a set of actions performed by AWS CodeBuild to create output artifacts (for example, a JAR file) based on a set of input artifacts (for example, a collection of Java class files).

The following rules apply when you run multiple builds:

When possible, builds run concurrently. The maximum number of concurrently running builds can vary.

Builds are queued if the number of concurrently running builds reaches its limit. The maximum number of builds in a queue is five times the concurrent build limit.

A build in a queue that does not start after the number of minutes specified in its time out value is removed from the queue. The default timeout value is eight hours. You can override the build queue timeout with a value between five minutes and eight hours when you run your build.

By setting the timeout configuration, the build process will automatically terminate post the expiry of the configured timeout.

Incorrect options:

Use AWS Lambda - AWS Lambda lets you run code without provisioning or managing servers. You pay only for the compute time you consume. Lambda cannot be used to impact the code build process.

Use AWS CloudWatch Events - Amazon CloudWatch allows you to monitor AWS cloud resources and the applications you run on AWS. Metrics are provided automatically for a number of AWS products and services. CloudWatch is good for monitoring and viewing logs. CloudWatch cannot be used to impact the code build process.

Use VPC Flow Logs - VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC but not for code compiling configuration. VPC Flow Logs cannot be used to impact the code build process.

Reference:

https://docs.aws.amazon.com/codebuild/latest/userguide/builds-working.html

Question 11: **Correct**

The Technical Lead of your team has reviewed a CloudFormation YAML template written by a new recruit and specified that an invalid section has been added to the template.

Which of the following represents an invalid section of the CloudFormation template?

- ○ **'Parameters' section of the template**
- ◉ **'Dependencies' section of the template** (Correct)
- ○ **'Conditions' section of the template**
- ○ **'Resources' section of the template**

Explanation
Correct option:

Templates include several major sections. The Resources section is the only required section.

Sample CloudFormation YAML template:

YAML

The following example shows a YAML-formatted template fragment.

```yaml
---
AWSTemplateFormatVersion: "version date"

Description:
  String

Metadata:
  template metadata

Parameters:
  set of parameters

Mappings:
  set of mappings

Conditions:
  set of conditions

Transform:
  set of transforms

Resources:
  set of resources

Outputs:
  set of outputs
```

via - https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-anatomy.html

'Dependencies' section of the template - As you can see, there is no section called 'Dependencies' in the template. Although dependencies can be mentioned, there is no section itself for dependencies.

Incorrect options:

'Conditions' section of the template - This optional section includes conditions that control whether certain resources are created or whether certain resource properties are assigned a value during stack creation or update. For example, you could conditionally create a resource that depends on whether the stack is for a production or test environment.

'Resources' section of the template - This is the only required section and specifies the stack resources and their properties, such as an Amazon Elastic Compute Cloud instance or an Amazon Simple Storage Service bucket. You can refer to resources in the Resources and Outputs sections of the template.

'Parameters' section of the template - This optional section is helpful in passing Values to your template at runtime (when you create or update a stack). You can refer to parameters from the Resources and Outputs sections of the template.

Reference:

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/template-anatomy.html

Question 12: **Correct**

The development team has just configured and attached the IAM policy needed to access AWS Billing and Cost Management for all users under the Finance department. But, the users are unable to see AWS Billing and Cost Management service in the AWS console.

What could be the reason for this issue?

- ○ **IAM user should be created under AWS Billing and Cost Management and not under AWS account to have access to Billing console**
- ○ **Only root user has access to AWS Billing and Cost Management console**
- ○ **The users might have another policy that restricts them from accessing the Billing information**
- ◉ **You need to activate IAM user access to the Billing and Cost Management console for all the users who need access** (Correct)

Explanation
Correct option:

You need to activate IAM user access to the Billing and Cost Management console for all the users who need access - By default, IAM users do not have access to the AWS Billing and Cost Management console. You or your account administrator must grant users access. You can do this by activating IAM user access to the Billing and Cost Management console and attaching an IAM policy to your users. Then, you need to activate IAM user access for IAM policies to take effect. You only need to activate IAM user access once.

Incorrect options:

The users might have another policy that restricts them from accessing the Billing information - This is an incorrect option, as deduced from the given use-case.

Only root user has access to AWS Billing and Cost Management console - This is an incorrect statement. AWS Billing and Cost Management access can be provided to any user through user activation and policies, as discussed above.

IAM user should be created under AWS Billing and Cost Management and not under the AWS account to have access to Billing console - IAM is a feature of your AWS account. All IAM users are created and managed from a single place, irrespective of the services they wish to you.

Reference:

Question 13: **Correct**

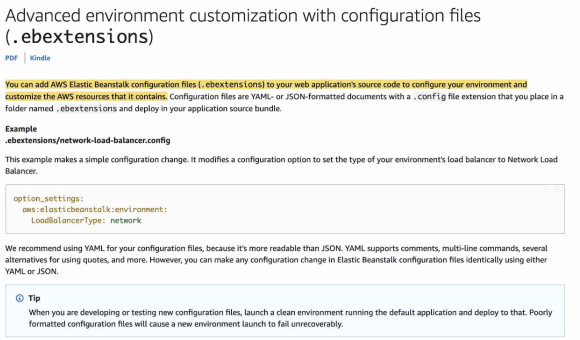
You have chosen AWS Elastic Beanstalk to upload your application code and allow it to handle details such as provisioning resources and monitoring.

When creating configuration files for AWS Elastic Beanstalk which naming convention should you follow?

- (●) `.ebextensions/<mysettings>.config` **(Correct)**
- ( ) `.config_<mysettings>.ebextensions`
- ( ) `.config/<mysettings>.ebextensions`
- ( ) `.ebextensions_<mysettings>.config`

Explanation
Correct option:

`.ebextensions/<mysettings>.config` : You can add AWS Elastic Beanstalk configuration files (`.ebextensions`) to your web application's source code to configure your environment and customize the AWS resources that it contains. Configuration files are YAML or JSON formatted documents with a .config file extension that you place in a folder named .ebextensions and deploy in your application source bundle.



Advanced environment customization with configuration files (.ebextensions)

PDF | Kindle

You can add AWS Elastic Beanstalk configuration files (.ebextensions) to your web application's source code to configure your environment and customize the AWS resources that it contains. Configuration files are YAML- or JSON-formatted documents with a .config file extension that you place in a folder named .ebextensions and deploy in your application source bundle.

Example
.ebextensions/network-load-balancer.config

This example makes a simple configuration change. It modifies a configuration option to set the type of your environment's load balancer to Network Load Balancer.

```
option_settings:
  aws:elasticbeanstalk:environment:
    LoadBalancerType: network
```

We recommend using YAML for your configuration files, because it's more readable than JSON. YAML supports comments, multi-line commands, several alternatives for using quotes, and more. However, you can make any configuration change in Elastic Beanstalk configuration files identically using either YAML or JSON.

Tip
When you are developing or testing new configuration files, launch a clean environment running the default application and deploy to that. Poorly formatted configuration files will cause a new environment launch to fail unrecoverably.

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

Incorrect options:

`.ebextensions_<mysettings>.config`

`.config/<mysettings>.ebextensions`

`.config_<mysettings>.ebextensions`

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

Reference:

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

Question 14: **Correct**

You have created an Elastic Load Balancer that has marked all the EC2 instances in the target group as unhealthy. Surprisingly, when you enter the IP address of the EC2 instances in your web browser, you can access your website.

What could be the reason your instances are being marked as unhealthy? (Select two)

- [x] **The security group of the EC2 instance does not allow for traffic from the security group of the Application Load Balancer** (Correct)
- [ ] **The EBS volumes have been improperly mounted**
- [ ] **Your web-app has a runtime that is not supported by the Application Load Balancer**
- [ ] **You need to attach Elastic IP to the EC2 instances**
- [x] **The route for the health check is misconfigured** (Correct)

Explanation

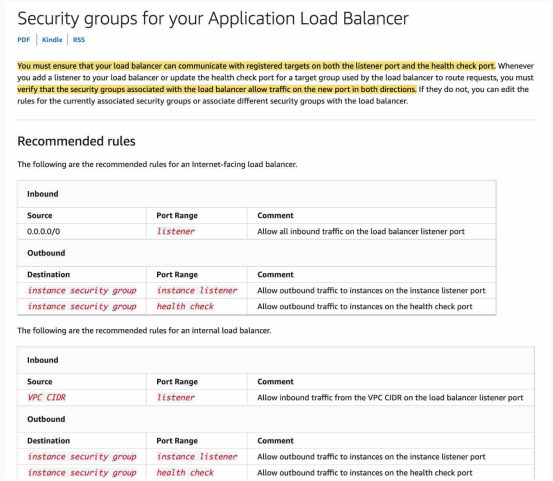
Correct options

The security group of the EC2 instance does not allow for traffic from the security group of the Application Load Balancer

The route for the health check is misconfigured

You must ensure that your load balancer can communicate with registered targets on both the listener port and the health check port. Whenever you add a listener to your load balancer or update the health check port for a target group used by the load balancer to route requests, you must verify that the security groups associated with the load balancer allow traffic on the new port in both directions.

Application Load Balancer Configuration for Security Groups and Health Check Routes:



Security groups for your Application Load Balancer

PDF | Kindle | RSS

You must ensure that your load balancer can communicate with registered targets on both the listener port and the health check port. Whenever you add a listener to your load balancer or update the health check port for a target group used by the load balancer to route requests, you must verify that the security groups associated with the load balancer allow traffic on the new port in both directions. If they do not, you can edit the rules for the currently associated security groups or associate different security groups with the load balancer.

Recommended rules

The following are the recommended rules for an Internet-facing load balancer.

Inbound		
Source	**Port Range**	**Comment**
0.0.0.0/0	*listener*	Allow all inbound traffic on the load balancer listener port
Outbound		
Destination	**Port Range**	**Comment**
instance security group	*instance listener*	Allow outbound traffic to instances on the instance listener port
instance security group	*health check*	Allow outbound traffic to instances on the health check port

The following are the recommended rules for an internal load balancer.

Inbound		
Source	**Port Range**	**Comment**
VPC CIDR	*listener*	Allow inbound traffic from the VPC CIDR on the load balancer listener port
Outbound		
Destination	**Port Range**	**Comment**
instance security group	*instance listener*	Allow outbound traffic to instances on the instance listener port
instance security group	*health check*	Allow outbound traffic to instances on the health check port

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

Incorrect options:

The EBS volumes have been improperly mounted - You can access the website using the IP address which means there is no issue with the EBS volumes. So this option is not correct.

Your web-app has a runtime that is not supported by the Application Load Balancer - There is no connection between a web app and the application load balancer. This option has been added as a distractor.

You need to attach Elastic IP to the EC2 instances - This option is a distractor as Elastic IPs do not need to be assigned to EC2 instances while using an Application Load Balancer.

References:

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

Question 15: **Correct**

When running a Rolling deployment in Elastic Beanstalk environment, only two batches completed the deployment successfully, while rest of the batches failed to deploy the updated version. Following this, the development team terminated the instances from the failed deployment.

What will be the status of these failed instances post termination?

- (●) **Elastic Beanstalk will replace the failed instances with instances running the application version from the most recent successful deployment** **(Correct)**
- ( ) **Elastic Beanstalk will replace the failed instances after the application version to be installed is manually chosen from AWS Console**
- ( ) **Elastic Beanstalk will replace the failed instances with instances running the application version from the oldest successful deployment**
- ( ) **Elastic Beanstalk will not replace the failed instances**

Explanation
Correct option:

Elastic Beanstalk will replace them with instances running the application version from the most recent successful deployment

When processing a batch, Elastic Beanstalk detaches all instances in the batch from the load balancer, deploys the new application version, and then reattaches the instances. If you enable connection draining, Elastic Beanstalk drains existing connections from the Amazon EC2 instances in each batch before beginning the deployment.

If a deployment fails after one or more batches completed successfully, the completed batches run the new version of your application while any pending batches continue to run the old version. You can identify the version running on the instances in your environment on the health page in the console. This page displays the deployment ID of the most recent deployment that was executed on each instance in your environment. If you terminate instances from the failed deployment, Elastic Beanstalk replaces them with instances running the application version from the most recent successful

deployment.

Incorrect options:

Elastic Beanstalk will not replace the failed instances

Elastic Beanstalk will replace the failed instances with instances running the application version from the oldest successful deployment

Elastic Beanstalk will replace the failed instances after the application version to be installed is manually chosen from AWS Console

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

Reference:

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

Question 16: **Correct**

ECS Fargate container tasks are usually spread across Availability Zones (AZs) and the underlying workloads need persistent cross-AZ shared access to the data volumes configured for the container tasks.

Which of the following solutions is the best choice for these workloads?

- (●) **Amazon EFS volumes** **(Correct)**
- ( ) **Bind mounts**
- ( ) **Docker volumes**
- ( ) **AWS Gateway Storage volumes**

Explanation
Correct option:

Amazon EFS volumes - EFS volumes provide a simple, scalable, and persistent file storage for use with your Amazon ECS tasks. With Amazon EFS, storage capacity is elastic, growing and shrinking automatically as you add and remove files. Your applications can have the storage they need, when they need it. Amazon EFS volumes are supported for tasks hosted on Fargate or Amazon EC2 instances.

You can use Amazon EFS file systems with Amazon ECS to export file system data across your fleet of container instances. That way, your tasks have access to the same persistent storage, no matter the instance on which they land. However, you must configure your container instance AMI to mount the Amazon EFS file system before the Docker daemon starts. Also, your task definitions must reference volume mounts on the container instance to use the file system.

Incorrect options:

Docker volumes - A Docker-managed volume that is created under /var/lib/docker/volumes on the host Amazon EC2 instance. Docker volume drivers (also referred to as plugins) are used to integrate the volumes with external storage systems, such as Amazon EBS. The built-in local volume driver or a third-party volume driver can be used. Docker volumes are only supported when running tasks on Amazon EC2 instances.

Bind mounts - A file or directory on the host, such as an Amazon EC2 instance or AWS Fargate, is mounted into a container. Bind mount host volumes are supported for tasks hosted on Fargate or Amazon EC2 instances. Bind mounts provide temporary storage, and hence these are a wrong choice for this use case.

AWS Storage Gateway volumes - This is an incorrect choice, given only as a distractor.

Reference:

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

https://aws.amazon.com/blogs/containers/amazon-ecs-availability-best-practices/

Question 17: **Correct**

As an AWS Certified Developer Associate, you have been asked to create an AWS Elastic Beanstalk environment to handle deployment for an application that has high traffic and high availability needs. You need to deploy the new version using Beanstalk while making sure that performance and availability are not affected.

Which of the following is the MOST optimal way to do this while keeping the solution cost-effective?

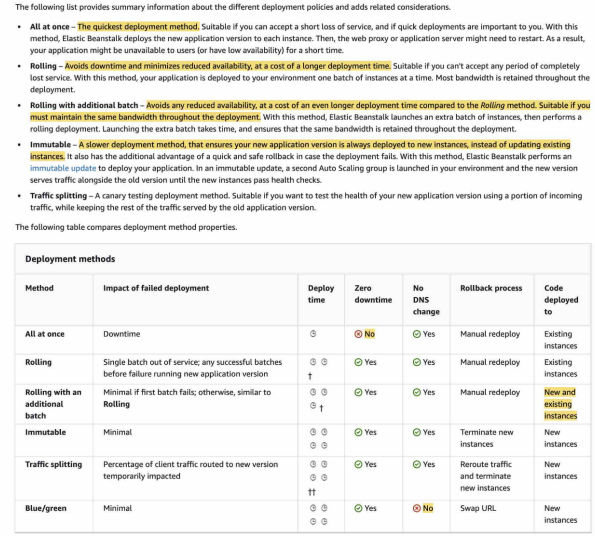
- ○ **Deploy using 'Rolling' deployment policy**
- ○ **Deploy using 'All at once' deployment policy**
- ○ **Deploy using 'Immutable' deployment policy**
- ◉ **Deploy using 'Rolling with additional batch' deployment policy** (Correct)

Explanation
Correct option:

AWS Elastic Beanstalk offers several deployment policies and settings. Choosing the right deployment policy for your application is a tradeoff based on a few considerations and depends on your business needs.

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

Overview of Elastic Beanstalk Deployment Policies:



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

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

The following table compares deployment method properties.

Deployment methods

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

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

Incorrect options:

Deploy using 'Immutable' deployment policy - A slower deployment method, that ensures your new application version is always deployed to new instances, instead of updating existing instances. It also has the additional advantage of a quick and safe rollback in case the deployment fails. With this method, Elastic Beanstalk performs an immutable update to deploy your application. In an immutable update, a second Auto Scaling group is launched in your environment and the new version serves traffic alongside the old version until the new instances pass health checks.

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

Deploy using 'Rolling' deployment policy - With this method, your application is deployed to your environment one batch of instances at a time. Most bandwidth is retained throughout the deployment. Avoids downtime and minimizes reduced availability, at a cost of a longer deployment time. Suitable if you can't accept any period of completely lost service. The use case states that the application has high traffic and high availability requirements, so full capacity must be maintained during deployments, hence rolling with additional batch deployment is a better fit than the rolling deployment.

Reference:

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

Question 18: **Correct**

Your company has configured AWS Organizations to manage multiple AWS accounts. Within each AWS account, there are many CloudFormation scripts running. Your manager has requested that each script output the account number of the account the script was executed in.

Which Pseudo parameter will you use to get this information?

- ○ **AWS::Region**
- ○ **AWS::StackName**
- ◉ **AWS::AccountId** **(Correct)**
- ○ **AWS::NoValue**

Explanation
Correct option:

AWS::AccountId

Using CloudFormation, you can create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and AWS CloudFormation takes care of provisioning and configuring those resources for you.

Pseudo parameters are parameters that are predefined by AWS CloudFormation. You do not declare them in your template. Use them the same way as you would a parameter, as the argument for the Ref function.

AWS::AccountId returns the AWS account ID of the account in which the stack is being created.

Incorrect options:

AWS::NoValue - This removes the corresponding resource property when specified as a return value in the Fn::If intrinsic function.

AWS::Region - Returns a string representing the AWS Region in which the encompassing resource is being created, such as us-west-2.

AWS::StackName - Returns the name of the stack as specified with the aws cloudformation create-stack command, such as "teststack".

Reference:

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

Question 19: **Incorrect**

A Developer has been entrusted with the job of securing certain S3 buckets that are shared by a large team of users. Last time, a bucket policy was changed, the bucket was erroneously available for everyone, outside the organization too.

Which feature/service will help the developer identify similar security issues with minimum effort?

- ○ **S3 Object Lock**
- ○ **IAM Access Analyzer** (Correct)
- ○ **S3 Analytics**
- ◉ **Access Advisor feature on IAM console** (Incorrect)

Explanation
Correct option:

IAM Access Analyzer - AWS IAM Access Analyzer helps you identify the resources in your organization and accounts, such as Amazon S3 buckets or IAM roles, that are shared with an external entity. This lets you identify unintended access to your resources and data, which is a security risk.

You can set the scope for the analyzer to an organization or an AWS account. This is your zone of trust. The analyzer scans all of the supported resources within your zone of trust. When Access Analyzer finds a policy that allows access to a resource from outside of your zone of trust, it generates an active finding.

Incorrect options:

Access Advisor feature on IAM console - To help identify the unused roles, IAM reports the last-used timestamp that represents when a role was last used to make an AWS request. Your security team can use this information to identify, analyze, and then confidently remove unused roles. This helps improve the security posture of your AWS environments. This does not provide information about non-IAM entities such as S3, hence it's not a correct choice here.

S3 Object Lock - S3 Object Lock enables you to store objects using a "Write Once Read Many" (WORM) model. S3 Object Lock can help prevent accidental or inappropriate deletion of data, it is not the right choice for the current scenario.

S3 Analytics - By using Amazon S3 analytics Storage Class Analysis you can analyze storage access patterns to help you decide when to transition the right data to the right storage class. You cannot use S3 Analytics to identify unintended access to your S3 resources.

References:

https://docs.aws.amazon.com/IAM/latest/UserGuide/what-is-access-analyzer.html

https://docs.aws.amazon.com/AmazonS3/latest/dev/security-best-practices.html

Question 19: **Incorrect**

A Developer has been entrusted with the job of securing certain S3 buckets that are shared by a large team of users. Last time, a bucket policy was changed, the bucket was erroneously available for everyone, outside the organization too.

Which feature/service will help the developer identify similar security issues with minimum effort?

- ○ **S3 Object Lock**
- ○ **IAM Access Analyzer** (Correct)
- ○ **S3 Analytics**
- ◉ **Access Advisor feature on IAM console** (Incorrect)

Explanation
Correct option:

IAM Access Analyzer - AWS IAM Access Analyzer helps you identify the resources in your organization and accounts, such as Amazon S3 buckets or IAM roles, that are shared with an external entity. This lets you identify unintended access to your resources and data, which is a security risk.

You can set the scope for the analyzer to an organization or an AWS account. This is your zone of trust. The analyzer scans all of the supported resources within your zone of trust. When Access Analyzer finds a policy that allows access to a resource from outside of your zone of trust, it generates an active finding.

Incorrect options:

Access Advisor feature on IAM console - To help identify the unused roles, IAM reports the last-used timestamp that represents when a role was last used to make an AWS request. Your security team can use this information to identify, analyze, and then confidently remove unused roles. This helps improve the security posture of your AWS environments. This does not provide information about non-IAM entities such as S3, hence it's not a correct choice here.

S3 Object Lock - S3 Object Lock enables you to store objects using a "Write Once Read Many" (WORM) model. S3 Object Lock can help prevent accidental or inappropriate deletion of data, it is not the right choice for the current scenario.

S3 Analytics - By using Amazon S3 analytics Storage Class Analysis you can analyze storage access patterns to help you decide when to transition the right data to the right storage class. You cannot use S3 Analytics to identify unintended access to your S3 resources.

References:

https://docs.aws.amazon.com/IAM/latest/UserGuide/what-is-access-analyzer.html

https://docs.aws.amazon.com/AmazonS3/latest/dev/security-best-practices.html

Question 21: **Incorrect**

A developer has been asked to create an application that can be deployed across a fleet of EC2 instances. The configuration must allow for full control over the deployment steps using the blue-green deployment.

Which service will help you achieve that?

- ○ **CodeBuild**
- ○ **CodeDeploy** (Correct)
- ◉ **Elastic Beanstalk** (Incorrect)
- ○ **CodePipeline**

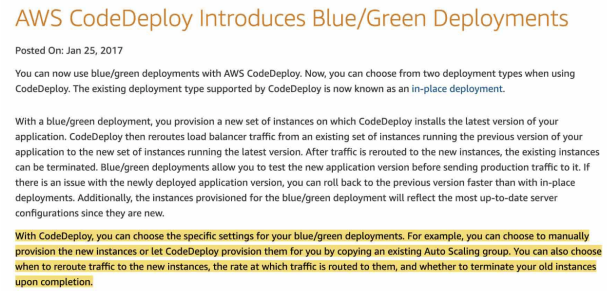
Explanation
Correct option:

CodeDeploy

AWS CodeDeploy is a deployment service that automates application deployments to Amazon EC2 instances, on-premises instances, or serverless Lambda functions. AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces software packages that are ready to deploy.

The blue/green deployment type uses the blue/green deployment model controlled by CodeDeploy. This deployment type enables you to verify a new deployment of service before sending production traffic to it.

CodeDeploy offers lot of control over deployment steps. Please see this note for more details:



AWS CodeDeploy Introduces Blue/Green Deployments

Posted On: Jan 25, 2017

You can now use blue/green deployments with AWS CodeDeploy. Now, you can choose from two deployment types when using CodeDeploy. The existing deployment type supported by CodeDeploy is now known as an in-place deployment.

With a blue/green deployment, you provision a new set of instances on which CodeDeploy installs the latest version of your application. CodeDeploy then reroutes load balancer traffic from an existing set of instances running the previous version of your application to the new set of instances running the latest version. After traffic is rerouted to the new instances, the existing instances can be terminated. Blue/green deployments allow you to test the new application version before sending production traffic to it. If there is an issue with the newly deployed application version, you can roll back to the previous version faster than with in-place deployments. Additionally, the instances provisioned for the blue/green deployment will reflect the most up-to-date server configurations since they are new.

With CodeDeploy, you can choose the specific settings for your blue/green deployments. For example, you can choose to manually provision the new instances or let CodeDeploy provision them for you by copying an existing Auto Scaling group. You can also choose when to reroute traffic to the new instances, the rate at which traffic is routed to them, and whether to terminate your old instances upon completion.

via - https://aws.amazon.com/about-aws/whats-new/2017/01/aws-codedeploy-introduces-blue-green-deployments/

Incorrect options:

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. It cannot be used to deploy applications.

Elastic Beanstalk - AWS Elastic Beanstalk offers hooks but not as much control as CodeDeploy. Because AWS Elastic Beanstalk performs an in-place update when you update your application versions, your application can become unavailable to users for a short period of time. You can avoid this downtime by performing a blue/green deployment, where you deploy the new version to a separate environment, and then swap CNAMEs of the two environments to redirect traffic to the new version instantly.

CodePipeline - CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change. CodePipeline by itself cannot deploy applications.

Reference:

https://docs.amazonaws.cn/en_us/codedeploy/latest/userguide/deployment-configurations.html

Question 22: **Incorrect**

A company wants to improve the performance of its popular API service that offers unauthenticated read access to daily updated statistical information via Amazon API Gateway and AWS Lambda.

What measures can the company take?

- ◯ **Enable API caching in API Gateway** (Correct)
- ◉ **Configure API Gateway to use Elasticache for Memcached** (Incorrect)
- ◯ **Set up usage plans and API keys in API Gateway**
- ◯ **Configure API Gateway to use Gateway VPC Endpoint**

Explanation
Correct option:

Enable API caching in API Gateway

API Gateway provides a few strategies for optimizing your API to improve responsiveness, like response caching and payload compression. 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.

Incorrect options:

Set up usage plans and API keys in API Gateway - After you create, test, and deploy your APIs, you can use API Gateway usage plans to make them available as product offerings for your customers. You can configure usage plans and API keys to allow customers to access selected APIs, and begin throttling requests to those APIs based on defined limits and quotas. These can be set at the API, or API method level. This option is incorrect as usage plans and API keys cannot be used to improve the responsiveness of the API.

Configure API Gateway to use Elasticache for Memcached - This option has been added as a distractor. Elasticache for Memcached cannot be used with API Gateway to improve the responsiveness of the API for the given use case. You should note that Elasticache for Memcached is a downstream service in the request flow.

Configure API Gateway to use Gateway VPC Endpoint - This option has been added as a distractor. 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.

References:

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

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

Question 23: **Correct**

A development team wants to build an application using serverless architecture. The team plans to use AWS Lambda functions extensively to achieve this goal. The developers of the team work on different programming languages like Python, .NET and Javascript. The team wants to model the cloud infrastructure using any of these programming languages.

Which AWS service/tool should the team use for the given use-case?

- ○ **AWS Serverless Application Model (SAM)**
- ○ **AWS CodeDeploy**
- ○ **AWS CloudFormation**
- ◉ **AWS Cloud Development Kit (CDK)** **(Correct)**

Explanation
Correct option:

AWS Cloud Development Kit (CDK) - The AWS Cloud Development Kit (AWS CDK) is an open-source software development framework to define your cloud application resources using familiar programming languages.

Provisioning cloud applications can be a challenging process that requires you to perform manual actions, write custom scripts, maintain templates, or learn domain-specific languages. AWS CDK uses the familiarity and expressive power of programming languages such as JavaScript/TypeScript, Python, Java, and .NET for modeling your applications. It provides you with high-level components called constructs that preconfigure cloud resources with proven defaults, so you can build cloud applications without needing to be an expert. AWS CDK provisions your resources in a safe, repeatable manner through AWS CloudFormation. It also enables you to compose and share your own custom constructs that incorporate your organization's requirements, helping you start new projects faster.

Incorrect options:

AWS CloudFormation - When AWS CDK applications are run, they compile down to fully formed CloudFormation JSON/YAML templates that are then submitted to the CloudFormation service for provisioning. Because the AWS CDK leverages CloudFormation, you still enjoy all the benefits CloudFormation provides such as safe deployment, automatic rollback, and drift detection. But, CloudFormation by itself is not sufficient for the current use case.

Explanation
Correct option:

AWS Cloud Development Kit (CDK) - The AWS Cloud Development Kit (AWS CDK) is an open-source software development framework to define your cloud application resources using familiar programming languages.

Provisioning cloud applications can be a challenging process that requires you to perform manual actions, write custom scripts, maintain templates, or learn domain-specific languages. AWS CDK uses the familiarity and expressive power of programming languages such as JavaScript/TypeScript, Python, Java, and .NET for modeling your applications. It provides you with high-level components called constructs that preconfigure cloud resources with proven defaults, so you can build cloud applications without needing to be an expert. AWS CDK provisions your resources in a safe, repeatable manner through AWS CloudFormation. It also enables you to compose and share your own custom constructs that incorporate your organization's requirements, helping you start new projects faster.

Incorrect options:

AWS CloudFormation - When AWS CDK applications are run, they compile down to fully formed CloudFormation JSON/YAML templates that are then submitted to the CloudFormation service for provisioning. Because the AWS CDK leverages CloudFormation, you still enjoy all the benefits CloudFormation provides such as safe deployment, automatic rollback, and drift detection. But, CloudFormation by itself is not sufficient for the current use case.

AWS Serverless Application Model (SAM) - The AWS Serverless Application Repository is a managed repository for serverless applications. It enables teams, organizations, and individual developers to store and share reusable applications, and easily assemble and deploy serverless architectures in powerful new ways. Using the Serverless Application Repository, you don't need to clone, build, package, or publish source code to AWS before deploying it.

AWS Serverless Application Model and AWS CDK both abstract AWS infrastructure as code making it easier for you to define your cloud infrastructure. If you prefer defining your serverless infrastructure in concise declarative templates, SAM is the better fit. If you want to define your AWS infrastructure in a familiar programming language, as is the requirement in the current use case, AWS CDK is the right fit.

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

Reference:

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

Question 24: **Incorrect**

After a test deployment in ElasticBeanstalk environment, a developer noticed that all accumulated Amazon EC2 burst balances were lost.

Which of the following options can lead to this behavior?

- (●) **When a canary deployment fails, it resets the EC2 burst balances to zero** **(Incorrect)**
- ( ) **The deployment was run as a Rolling deployment, resulting in the resetting of EC2 burst balances**
- ( ) **The deployment was run as a All-at-once deployment, flushing all the accumulated EC2 burst balances**
- ( ) **The deployment was either run with immutable updates or in traffic splitting mode** **(Correct)**

Explanation
Correct option:

The deployment was either run with immutable updates or in traffic splitting mode - Immutable deployments perform an immutable update to launch a full set of new instances running the new version of the application in a separate Auto Scaling group, alongside the instances running the old version. Immutable deployments can prevent issues caused by partially completed rolling deployments.

Traffic-splitting deployments let you perform canary testing as part of your application deployment. In a traffic-splitting deployment, Elastic Beanstalk launches a full set of new instances just like during an immutable deployment. It then forwards a specified percentage of incoming client traffic to the new application version for a specified evaluation period.

Some policies replace all instances during the deployment or update. This causes all accumulated Amazon EC2 burst balances to be lost. It happens in the following cases:

1. Managed platform updates with instance replacement enabled
2. Immutable updates
3. Deployments with immutable updates or traffic splitting enabled

Incorrect options:

The deployment was run as a Rolling deployment, resulting in the resetting of EC2 burst balances - With rolling deployments, Elastic Beanstalk splits the environment's Amazon EC2 instances into batches and deploys the new version of the application to one batch at a time. Rolling deployments do not result in loss of EC2 burst balances.

The deployment was run as a All-at-once deployment, flushing all the accumulated EC2 burst balances - The traditional All-at-once deployment, wherein all the instances are updated simultaneously, does not result in loss of EC2 burst balances.

When a canary deployment fails, it resets the EC2 burst balances to zero - This is incorrect and given only as a distractor.

Reference:

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

Question 25: **Correct**

A development team wants to deploy an AWS Lambda function that requires significant CPU utilization.

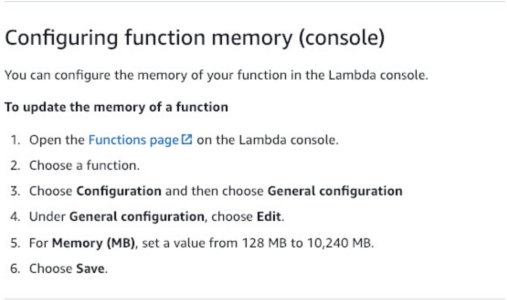
As a Developer Associate, which of the following would you suggest for reducing the average runtime of the function?

- ◯ **Deploy the function with its CPU allocation set to the maximum amount**
- ◉ **Deploy the function with its memory allocation set to the maximum amount** **(Correct)**
- ◯ **Deploy the function using Lambda layers**
- ◯ **Deploy the function into multiple AWS Regions**

Explanation
Correct option:

Deploy the function with its memory allocation set to the maximum amount - Lambda allocates CPU power in proportion to the amount of memory configured. Memory is the amount of memory available to your Lambda function at runtime. You can increase or decrease the memory and CPU power allocated to your function using the Memory (MB) setting. To configure the memory for your function, set a value between 128 MB and 10,240 MB in 1-MB increments. At 1,769 MB, a function has the equivalent of one vCPU (one vCPU-second of credits per second).

Configuring function memory from AWS console:



Configuring function memory (console)

You can configure the memory of your function in the Lambda console.

To update the memory of a function

1. Open the Functions page on the Lambda console.
2. Choose a function.
3. Choose **Configuration** and then choose **General configuration**
4. Under **General configuration**, choose **Edit**.
5. For **Memory (MB)**, set a value from 128 MB to 10,240 MB.
6. Choose **Save**.

via -

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

Incorrect options:

Deploy the function into multiple AWS Regions - Deploying the Lambda function to multiple AWS Regions does not increase the compute capacity or CPU utilization capacity of Lambda. So, this option is irrelevant.

Deploy the function using Lambda layers - A Lambda layer is a .zip file archive that can contain additional code or data. A layer can contain libraries, a custom runtime, data, or configuration files. Layers promote code sharing and separation of responsibilities so that you can iterate faster on writing business logic. Layers do not increase the computational capacity of Lambda.

Deploy the function with its CPU allocation set to the maximum amount - This statement is given as a distractor. `CPU allocation` is an invalid parameter. As discussed above, the CPU is allocated in proportion to the memory allocated to the function.

References:

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

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

Question 26: **Correct**

An organization has hosted its EC2 instances in two AZs. AZ1 has two instances and AZ2 has 8 instances. The Elastic Load Balancer managing the instances in the two AZs has cross-zone load balancing enabled in its configuration.

What percentage traffic will each of the instances in AZ1 receive?

- ○ **20**
- ○ **25**
- ○ **15**
- ◉ **10** **(Correct)**

Explanation
Correct option:

A load balancer accepts incoming traffic from clients and routes requests to its registered targets (such as EC2 instances) in one or more Availability Zones.

The nodes for a load balancer distribute requests from clients to registered targets. When cross-zone load balancing is enabled, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones. When cross-zone load balancing is disabled, each load balancer node distributes traffic only across the registered targets in its Availability Zone. With Application Load Balancers, cross-zone load balancing is always enabled.

10 - When cross-zone load balancing is enabled, each of the 10 targets receives 10% of the traffic. This is because each load balancer node can route its 50% of the client traffic to all 10 targets (present in both AZs).

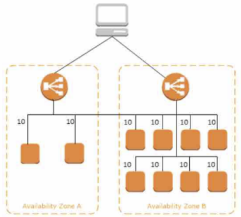
Cross-Zone Load Balancing Overview:

Cross-zone load balancing

The nodes for your load balancer distribute requests from clients to registered targets. When cross-zone load balancing is enabled, each load balancer node distributes traffic across the registered targets in all enabled Availability Zones. When cross-zone load balancing is disabled, each load balancer node distributes traffic only across the registered targets in its Availability Zone.

The following diagrams demonstrate the effect of cross-zone load balancing. There are two enabled Availability Zones, with two targets in Availability Zone A and eight targets in Availability Zone B. Clients send requests, and Amazon Route 53 responds to each request with the IP address of one of the load balancer nodes. This distributes traffic such that each load balancer node receives 50% of the traffic from the clients. Each load balancer node distributes its share of the traffic across the registered targets in its scope.

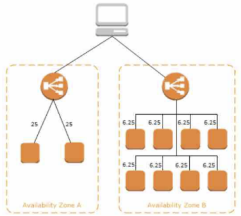
If cross-zone load balancing is enabled, each of the 10 targets receives 10% of the traffic. This is because each load balancer node can route its 50% of the client traffic to all 10 targets.



If cross-zone load balancing is disabled:

- Each of the two targets in Availability Zone A receives 25% of the traffic.
- Each of the eight targets in Availability Zone B receives 6.25% of the traffic.

This is because each load balancer node can route its 50% of the client traffic only to targets in its Availability Zone.



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

Incorrect options:

25 - If cross-zone load balancing is disabled, each of the two targets in AZ1 will receive 25% of the traffic. Because the load balancer is only able to send to the targets registered in AZ1 (AZ2 instances are not accessible for load balancer on AZ1)

20 - Invalid option, given only as a distractor.

15 - Invalid option, given only as a distractor.

Reference:

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

Question 27: **Correct**

A data analytics company processes Internet-of-Things (IoT) data using Amazon Kinesis. The development team has noticed that the IoT data feed into Kinesis experiences periodic spikes. The PutRecords API call occasionally fails and the logs show that the failed call returns the response shown below:

```
HTTP/1.1 200 OK
x-amzn-RequestId: <RequestId>
Content-Type: application/x-amz-json-1.1
Content-Length: <PayloadSizeBytes>
Date: <Date>
{
    "FailedRecordCount": 2,
    "Records": [
        {
            "SequenceNumber": "49543463076548007577105092703039560359975228518395012686",
            "ShardId": "shardId-000000000000"
        },
        {
            "ErrorCode": "ProvisionedThroughputExceededException",
            "ErrorMessage": "Rate exceeded for shard shardId-000000000001 in stream exampleStreamName under account 111111111111."
        },
        {
            "ErrorCode": "InternalFailure",
            "ErrorMessage": "Internal service failure."
        }
    ]
}
```

As an AWS Certified Developer Associate, which of the following options would you recommend to address this use case? (Select two)

- [ ] **Increase the frequency or size of your requests**
- [x] **Use an error retry and exponential backoff mechanism** (Correct)
- [ ] **Decrease the number of KCL consumers**
- [ ] **Merge the shards to decrease the number of shards in the stream**
- [x] **Decrease the frequency or size of your requests** (Correct)

Explanation

Correct options:

Use an error retry and exponential backoff mechanism

Decrease the frequency or size of your requests

You can use PutRecords API call to write multiple data records into a Kinesis data stream in a single call. Each PutRecords request can support up to 500 records. Each record in the request can be as large as 1 MiB, up to a limit of 5 MiB for the entire request, including partition keys. Each shard can support writes up to 1,000 records per second, up to a maximum data write of 1 MiB per second.

The response Records array includes both successfully and unsuccessfully processed records. Kinesis Data Streams attempts to process all records in each PutRecords request. A single record failure does not stop the processing of subsequent records. As a result, PutRecords doesn't guarantee the ordering of records. An unsuccessfully processed record includes ErrorCode and ErrorMessage values. ErrorCode reflects the type of error and can be one of the following values: `ProvisionedThroughputExceededException` or `InternalFailure`. `ProvisionedThroughputExceededException` indicates that the request rate for the stream is too high, or the requested data is too large for the available throughput. Reduce the frequency or size of your requests.

To address the given use case, you can apply these best practices:

Reshard your stream to increase the number of shards in the stream.

Reduce the frequency or size of your requests.

Distribute read and write operations as evenly as possible across all of the shards in Data Streams.

Use an error retry and exponential backoff mechanism.

Incorrect options:

Merge the shards to decrease the number of shards in the stream

Increase the frequency or size of your requests

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

Decrease the number of KCL consumers - This option has been added as a distractor. The number of KCL consumers is irrelevant for the given use case since the `ProvisionedThroughputExceededException` is due to the PutRecords API call being used by the producers.

References:

https://aws.amazon.com/premiumsupport/knowledge-center/kinesis-readprovisionedthroughputexceeded/

https://docs.aws.amazon.com/kinesis/latest/APIReference/API_PutRecords.html

Question 28: **Correct**

A retail company is migrating its on-premises database to Amazon RDS for PostgreSQL. The company has read-heavy workloads. The development team at the company is looking at refactoring the code to achieve optimum read performance for SQL queries.

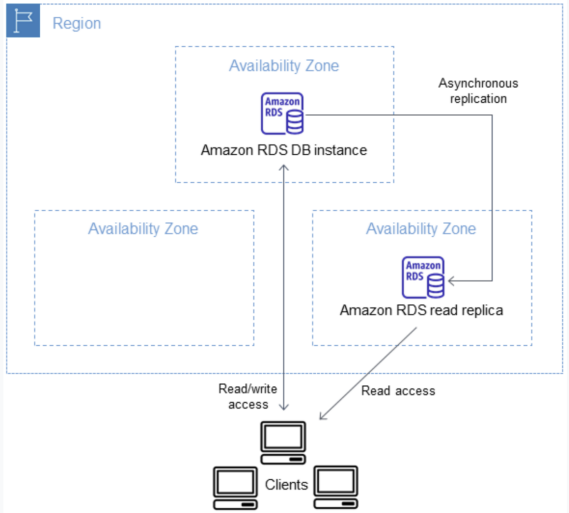
Which solution will address this requirement with the least current as well as future development effort?

- ○ **Configure Elasticache for Redis to act as a caching layer for Amazon RDS. Refactor the application code so that the queries use the Elasticache for Redis endpoint**
- ◉ **Set up Amazon RDS with one or more read replicas. Refactor the application code so that the queries use the endpoint for the read replicas** (Correct)
- ○ **Configure Elasticache for Memcached to act as a caching layer for Amazon RDS. Refactor the application code so that the queries use the Elasticache for Memcached endpoint**
- ○ **Set up Amazon RDS in the multi-AZ configuration with a single standby instance. Refactor the application code so that the queries use the standby instance endpoint**

Explanation
Correct option:

Set up Amazon RDS with one or more read replicas. Refactor the application code so that the queries use the endpoint for the read replicas

Amazon RDS uses the PostgreSQL DB engine's built-in replication functionality to create a special type of DB instance called a read replica from a source DB instance. The source DB instance becomes the primary DB instance. Updates made to the primary DB instance are asynchronously copied to the read replica. You can reduce the load on your primary DB instance by routing read queries from your applications to the read replica. Using read replicas, you can elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads. For the given use case, you can achieve optimum read performance for SQL queries by using the read-replica endpoint for the read-heavy workload.



via -

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

Incorrect options:

Configure Elasticache for Redis to act as a caching layer for Amazon RDS. Refactor the application code so that the queries use the Elasticache for Redis endpoint

Configure Elasticache for Memcached to act as a caching layer for Amazon RDS. Refactor the application code so that the queries use the Elasticache for Memcached endpoint

Both Redis and Memcached are popular, open-source, in-memory data stores (also known as in-memory caches). These are not relational databases and cannot be used to run SQL queries. So, both these options are incorrect.

Set up Amazon RDS in the multi-AZ configuration with a single standby instance. Refactor the application code so that the queries use the standby instance endpoint - In an Amazon RDS Multi-AZ deployment with a single standby instance, Amazon RDS automatically creates a primary database (DB) instance and synchronously replicates the data to an instance in a different AZ. When it detects a failure, Amazon RDS automatically fails over to a standby instance without manual intervention. You cannot route the read queries from an application to the standby instance of a multi-AZ RDS database as it's not accessible for the read traffic in the single standby instance configuration.

References:

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

https://aws.amazon.com/premiumsupport/knowledge-center/requests-rds-read-replicas/

https://aws.amazon.com/rds/features/multi-az/

https://aws.amazon.com/blogs/database/readable-standby-instances-in-amazon-rds-multi-az-deployments-a-new-high-availability-option/

Question 29: **Incorrect**

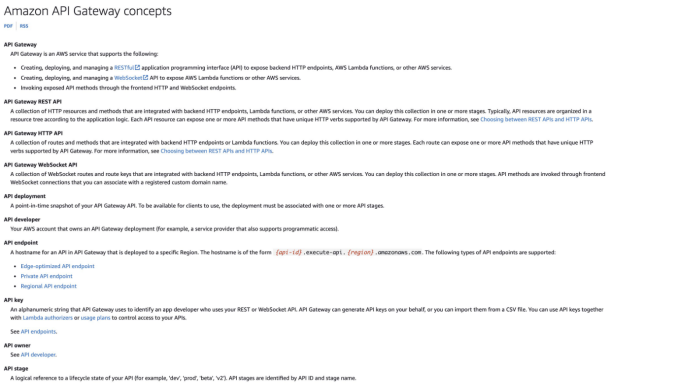
A company wants to provide beta access to some developers on its development team for a new version of the company's Amazon API Gateway REST API, without causing any disturbance to the existing customers who are using the API via a frontend UI and Amazon Cognito authentication. The new version has new endpoints and backward-incompatible interface changes, and the company's development team is responsible for its maintenance.

Which of the following will satisfy these requirements in the MOST operationally efficient manner?

- ○ **Configure a canary release deployment on the API Gateway API and then have the developers point to the relevant deployment by referencing the stage variable in the endpoint**
- ◉ **Create a new API Gateway API that points to the new API application code and then have the developers point the endpoints to the new API** **(Incorrect)**
- ○ **Create a development stage on the API Gateway API and then have the developers point the endpoints to the development stage** **(Correct)**
- ○ **Create new API keys on the API Gateway API and then have the developers point the endpoints by passing the new API keys**

Explanation
Correct option:

Create a development stage on the API Gateway API and then have the developers point the endpoints to the development stage



Amazon API Gateway concepts

PDF | RSS

API Gateway
API Gateway is an AWS service that supports the following:

- Creating, deploying, and managing a RESTful application programming interface (API) to expose backend HTTP endpoints, AWS Lambda functions, or other AWS services.
- Creating, deploying, and managing a WebSocket API to expose AWS Lambda functions or other AWS services.
- Invoking exposed API methods through the frontend HTTP and WebSocket endpoints.

API Gateway REST API
A collection of HTTP resources and methods that are integrated with backend HTTP endpoints, Lambda functions, or other AWS services. You can deploy this collection in one or more stages. Typically, API resources are organized in a resource tree according to the application logic. Each API resource can expose one or more API methods that have unique HTTP verbs supported by API Gateway. For more information, see Choosing between REST APIs and HTTP APIs.

API Gateway HTTP API
A collection of routes and methods that are integrated with backend HTTP endpoints or Lambda functions. You can deploy this collection in one or more stages. Each route can expose one or more API methods that have unique HTTP verbs supported by API Gateway. For more information, see Choosing between REST APIs and HTTP APIs.

API Gateway WebSocket API
A collection of WebSocket routes and route keys that are integrated with backend HTTP endpoints, Lambda functions, or other AWS services. You can deploy this collection in one or more stages. API methods are invoked through frontend WebSocket connections that you can associate with a registered custom domain name.

API deployment
A point-in-time snapshot of your API Gateway API. To be available for clients to use, the deployment must be associated with one or more API stages.

API developer
Your AWS account that owns an API Gateway deployment (for example, a service provider that also supports programmatic access).

API endpoint
A hostname for an API in API Gateway that is deployed to a specific Region. The hostname is of the form `{api-id}.execute-api.{region}.amazonaws.com`. The following types of API endpoints are supported:

- Edge-optimized API endpoint
- Private API endpoint
- Regional API endpoint

API key
An alphanumeric string that API Gateway uses to identify an app developer who uses your REST or WebSocket API. API Gateway can generate API keys on your behalf, or you can import them from a CSV file. You can use API keys together with Lambda authorizers or usage plans to control access to your APIs.

See API endpoints.

API owner
See API developer.

API stage
A logical reference to a lifecycle state of your API (for example, 'dev', 'prod', 'beta', 'v2'). API stages are identified by API ID and stage name.

via - https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-basic-concept.html

A stage is a logical reference to a lifecycle state of your API (for example, 'dev', 'prod', 'beta', 'v2'). API stages are identified by API ID and stage name. You use a stage to manage and optimize a particular deployment. For example, you can configure stage settings to enable caching, customize request throttling, configure logging, define stage variables, or attach a canary release for testing. After the initial deployment, you can add more stages and associate them with existing deployments. You can use the API Gateway console to create a new stage, or you can choose an existing stage while deploying an API. In general, you can add a new stage to an API deployment before redeploying the API.

For the given use case, you can configure a development stage for your API Gateway API and then integrate it with the new version of the backend functionality that has new endpoints and backward-incompatible interface changes. The customers can continue to use the existing API.

Incorrect options:

Configure a canary release deployment on the API Gateway API and then have the developers point to the relevant deployment by referencing the stage variable in the endpoint - An API deployment is a point-in-time snapshot of your API Gateway API. To be available for clients to use, the deployment must be associated with one or more API stages. Canary release is a software development strategy in which a new version of an API (as well as other software) is deployed for testing purposes, and the base version remains deployed as a production release for normal operations on the same stage. In a canary release deployment, total API traffic is separated at random into a production release and a canary release with a pre-configured ratio. Typically, the canary release receives a small percentage of API traffic and the production release takes up the rest. The updated API features are only visible to API traffic through the canary. You can adjust the canary traffic percentage to optimize test coverage or performance. By keeping canary traffic small and the selection random, most users are not adversely affected at any time by potential bugs in the new version, and no single user is adversely affected all the time.

This option is incorrect for the given use case as some of the customers would also access the new version of the API.

Create new API keys on the API Gateway API and then have the developers point the endpoints by passing the new API keys - AN API key is an alphanumeric string that API Gateway uses to identify an app developer who uses your REST or WebSocket API. This option is a distractor as you cannot selectively provide access to the new version just based on API keys.

Create a new API Gateway API that points to the new API application code and then have the developers point the endpoints to the new API - This is an overkill for the given requirement as there is no need to create a completely new API just to provide some developers early access to the beta version.

References:

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

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

https://docs.aws.amazon.com/apigateway/latest/developerguide/canary-release.html

Question 30: **Correct**

A SaaS company runs a HealthCare web application that is used worldwide by users. There have been requests by mobile developers to expose public APIs for the application-specific functionality. You decide to make the APIs available to mobile developers as product offerings.

Which of the following options will allow you to do that?

- ○ **Use CloudFront Usage Plans**
- ○ **Use AWS Lambda Custom Authorizers**
- ◉ **Use API Gateway Usage Plans** **(Correct)**
- ○ **Use AWS Billing Usage Plans**

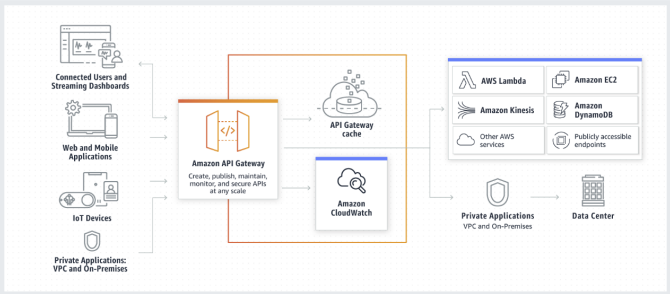
Explanation

Correct option:

Use API Gateway Usage Plans

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

How API Gateway Works:

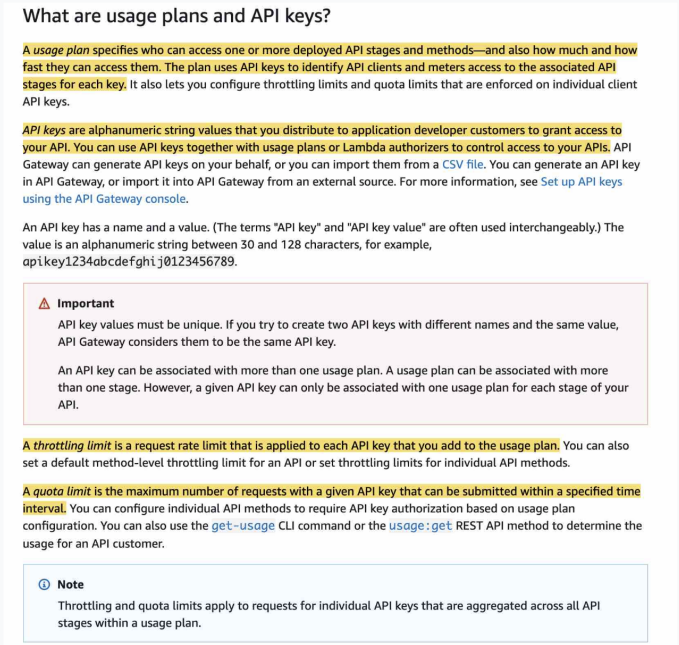


via - https://aws.amazon.com/api-gateway/

A usage plan specifies who can access one or more deployed API stages and methods—and also how much and how fast they can access them. The plan uses API keys to identify API clients and meters access to the associated API stages for each key.

You can configure usage plans and API keys to allow customers to access selected APIs at agreed-upon request rates and quotas that meet their business requirements and budget constraints.

Overview of API Gateway Usage Plans and API keys:



What are usage plans and API keys?

A *usage plan* specifies who can access one or more deployed API stages and methods—and also how much and how fast they can access them. The plan uses API keys to identify API clients and meters access to the associated API stages for each key. It also lets you configure throttling limits and quota limits that are enforced on individual client API keys.

API keys are alphanumeric string values that you distribute to application developer customers to grant access to your API. You can use API keys together with usage plans or Lambda authorizers to control access to your APIs. API Gateway can generate API keys on your behalf, or you can import them from a CSV file. You can generate an API key in API Gateway, or import it into API Gateway from an external source. For more information, see Set up API keys using the API Gateway console.

An API key has a name and a value. (The terms "API key" and "API key value" are often used interchangeably.) The value is an alphanumeric string between 30 and 128 characters, for example, `apikey1234abcdefghij0123456789`.

Important

API key values must be unique. If you try to create two API keys with different names and the same value, API Gateway considers them to be the same API key.

An API key can be associated with more than one usage plan. A usage plan can be associated with more than one stage. However, a given API key can only be associated with one usage plan for each stage of your API.

A *throttling limit* is a request rate limit that is applied to each API key that you add to the usage plan. You can also set a default method-level throttling limit for an API or set throttling limits for individual API methods.

A *quota limit* is the maximum number of requests with a given API key that can be submitted within a specified time interval. You can configure individual API methods to require API key authorization based on usage plan configuration. You can also use the `get-usage` CLI command or the `usage:get` REST API method to determine the usage for an API customer.

Note

Throttling and quota limits apply to requests for individual API keys that are aggregated across all API stages within a usage plan.

via - https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-api-usage-plans.html

Incorrect options:

Use AWS Billing Usage Plans - AWS Billing and Cost Management is the service that you use to pay your AWS bill, monitor your usage, and analyze and control your costs. There is no such thing as AWS Billing Usage Plans. You cannot use AWS Billing to set up public APIs for the application.

Use CloudFront Usage Plans - Amazon CloudFront is a fast content delivery network (CDN) service that securely delivers data, videos, applications, and APIs to customers globally with low latency, high transfer speeds, all within a developer-friendly environment. There is no such thing as CloudFront Usage Plans. You cannot use CloudFront to set up public APIs for the application.

Use AWS Lambda Custom Authorizers - Lambda is a separate service than Gateway API, therefore, it cannot be used to determine the API usage limits.

Reference:

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

Question 31: **Correct**

A cybersecurity firm wants to run their applications on single-tenant hardware to meet security guidelines.

Which of the following is the MOST cost-effective way of isolating their Amazon EC2 instances to a single tenant?

- ○ **Dedicated Hosts**
- ○ **Spot Instances**
- ◉ **Dedicated Instances** **(Correct)**
- ○ **On-Demand Instances**

Explanation
Correct option:

Dedicated Instances - Dedicated Instances are Amazon EC2 instances that run in a virtual private cloud (VPC) on hardware that's dedicated to a single customer. Dedicated Instances that belong to different AWS accounts are physically isolated at a hardware level, even if those accounts are linked to a single-payer account. However, Dedicated Instances may share hardware with other instances from the same AWS account that are not Dedicated Instances.

A Dedicated Host is also a physical server that's dedicated for your use. With a Dedicated Host, you have visibility and control over how instances are placed on the server.

Differences between Dedicated Hosts and Dedicated Instances:

Differences between Dedicated Hosts and Dedicated Instances

Dedicated Hosts and Dedicated Instances can both be used to launch Amazon EC2 instances onto physical servers that are dedicated for your use.

There are no performance, security, or physical differences between Dedicated Instances and instances on Dedicated Hosts. However, there are some differences between the two. The following table highlights some of the key differences between Dedicated Hosts and Dedicated Instances:

	Dedicated Host	Dedicated Instance
Billing	Per-host billing	Per-instance billing
Visibility of sockets, cores, and host ID	Provides visibility of the number of sockets and physical cores	No visibility
Host and instance affinity	Allows you to consistently deploy your instances to the same physical server over time	Not supported
Targeted instance placement	Provides additional visibility and control over how instances are placed on a physical server	Not supported
Automatic instance recovery	Supported. For more information, see Host recovery.	Supported
Bring Your Own License (BYOL)	Supported	Not supported

via - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/dedicated-hosts-overview.html#dedicated-hosts-dedicated-instances

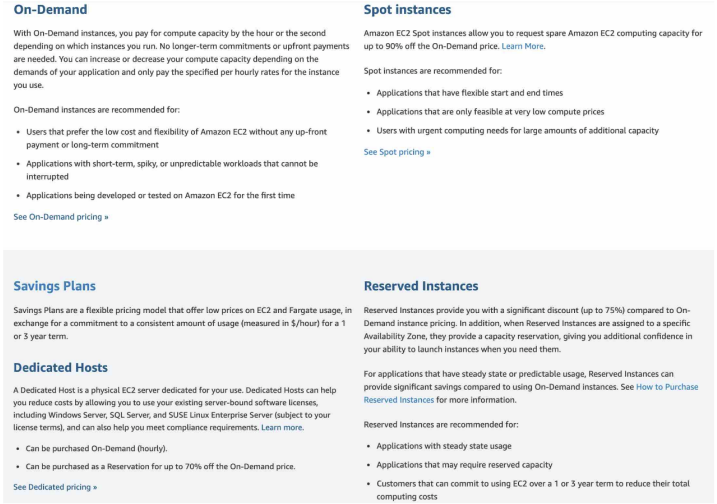
Incorrect options:

Spot Instances - A Spot Instance is an unused EC2 instance that is available for less than the On-Demand price. Your Spot Instance runs whenever capacity is available and the maximum price per hour for your request exceeds the Spot price. Any instance present with unused capacity will be allocated. Even though this is cost-effective, it does not fulfill the single-tenant hardware requirement of the client and hence is not the correct option.

Dedicated Hosts - An Amazon EC2 Dedicated Host is a physical server with EC2 instance capacity fully dedicated to your use. Dedicated Hosts allow you to use your existing software licenses on EC2 instances. With a Dedicated Host, you have visibility and control over how instances are placed on the server. This option is costlier than the Dedicated Instance and hence is not the right choice for the current requirement.

On-Demand Instances - With On-Demand Instances, you pay for compute capacity by the second with no long-term commitments. You have full control over its lifecycle—you decide when to launch, stop, hibernate, start, reboot, or terminate it. Hardware isolation is not possible and on-demand has one of the costliest instance charges and hence is not the correct answer for current requirements.

High Level Overview of EC2 Instance Purchase Options:



via - https://aws.amazon.com/ec2/pricing/

References:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/dedicated-instance.html

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-purchasing-options.html

Question 32: **Correct**

You have deployed a Java application to an EC2 instance where it uses the X-Ray SDK. When testing from your personal computer, the application sends data to X-Ray but when the application runs from within EC2, the application fails to send data to X-Ray.

Which of the following does **NOT** help with debugging the issue?

- ○ **EC2 X-Ray Daemon**
- ○ **EC2 Instance Role**
- ◉ **X-Ray sampling** (Correct)
- ○ **CloudTrail**

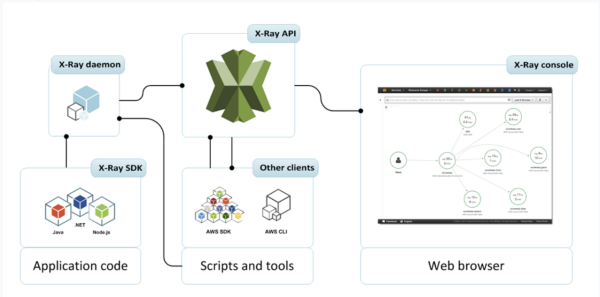
Explanation

Correct option:

X-Ray sampling

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. X-Ray SDK applies a sampling algorithm to determine which requests get traced however because our application is failing to send data to X-Ray it does not help in determining the cause of failure.

X-Ray Overview:



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

Incorrect options:

EC2 X-Ray Daemon - 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 logs could help with figuring out the problem.

EC2 Instance Role - The X-Ray daemon uses the AWS SDK to upload trace data to X-Ray, and it needs AWS credentials with permission to do that. On Amazon EC2, the daemon uses the instance's instance profile role automatically. Eliminates API permission issues (in case the role doesn't have IAM permissions to write data to the X-Ray service)

CloudTrail - With CloudTrail, you can log, continuously monitor, and retain account activity related to actions across your AWS infrastructure. You can use AWS CloudTrail to answer questions such as - "Who made an API call to modify this resource?". CloudTrail provides event history of your AWS account activity thereby enabling governance, compliance, operational auditing, and risk auditing of your AWS account. You can check CloudTrail to see if any API call is being denied on X-Ray.

Reference:

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

Question 33: **Correct**

Which of the following best describes how KMS Encryption works?

- ○ **KMS sends the CMK to the client, which performs the encryption and then deletes the CMK**
- ○ **KMS generates a new CMK for each Encrypt call and encrypts the data with it**
- ◉ **KMS stores the CMK, and receives data from the clients, which it encrypts and sends back** **(Correct)**
- ○ **KMS receives CMK from the client at every Encrypt call, and encrypts the data with that**

Explanation

Correct option:

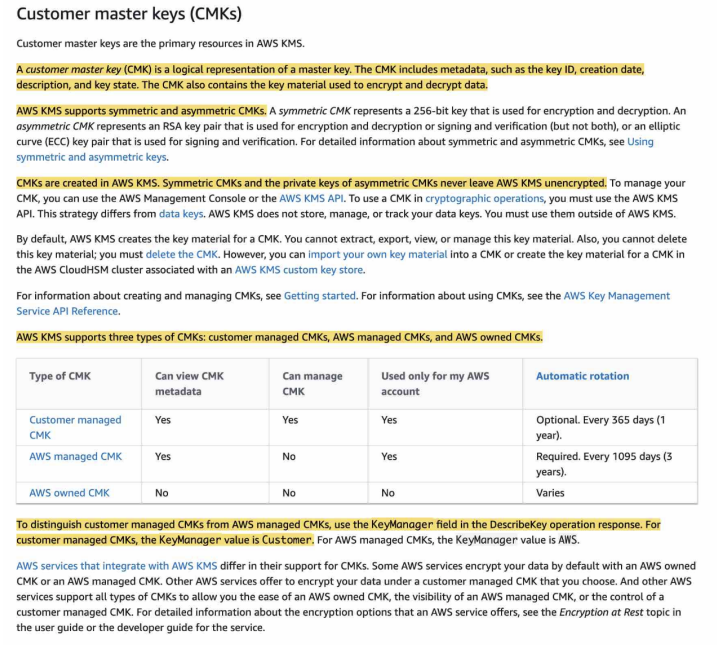
KMS stores the CMK, and receives data from the clients, which it encrypts and sends back

A customer master key (CMK) is a logical representation of a master key. The CMK includes metadata, such as the key ID, creation date, description, and key state. The CMK also contains the key material used to encrypt and decrypt data. You can generate CMKs in KMS, in an AWS CloudHSM cluster, or import them from your key management infrastructure.

AWS KMS supports symmetric and asymmetric CMKs. A symmetric CMK represents a 256-bit key that is used for encryption and decryption. An asymmetric CMK represents an RSA key pair that is used for encryption and decryption or signing and verification (but not both), or an elliptic curve (ECC) key pair that is used for signing and verification.

AWS KMS supports three types of CMKs: customer-managed CMKs, AWS managed CMKs, and AWS owned CMKs.

Overview of Customer master keys (CMKs):



Customer master keys (CMKs)

Customer master keys are the primary resources in AWS KMS.

A *customer master key* (CMK) is a logical representation of a master key. The CMK includes metadata, such as the key ID, creation date, description, and key state. The CMK also contains the key material used to encrypt and decrypt data.

AWS KMS supports symmetric and asymmetric CMKs. A *symmetric CMK* represents a 256-bit key that is used for encryption and decryption. An *asymmetric CMK* represents an RSA key pair that is used for encryption and decryption or signing and verification (but not both), or an elliptic curve (ECC) key pair that is used for signing and verification. For detailed information about symmetric and asymmetric CMKs, see Using symmetric and asymmetric keys.

CMKs are created in AWS KMS. Symmetric CMKs and the private keys of asymmetric CMKs never leave AWS KMS unencrypted. To manage your CMK, you can use the AWS Management Console or the AWS KMS API. To use a CMK in cryptographic operations, you must use the AWS KMS API. This strategy differs from data keys. AWS KMS does not store, manage, or track your data keys. You must use them outside of AWS KMS.

By default, AWS KMS creates the key material for a CMK. You cannot extract, export, view, or manage this key material. Also, you cannot delete this key material; you must delete the CMK. However, you can import your own key material into a CMK or create the key material for a CMK in the AWS CloudHSM cluster associated with an AWS KMS custom key store.

For information about creating and managing CMKs, see Getting started. For information about using CMKs, see the AWS Key Management Service API Reference.

AWS KMS supports three types of CMKs: customer managed CMKs, AWS managed CMKs, and AWS owned CMKs.

Type of CMK	Can view CMK metadata	Can manage CMK	Used only for my AWS account	Automatic rotation
Customer managed CMK	Yes	Yes	Yes	Optional. Every 365 days (1 year).
AWS managed CMK	Yes	No	Yes	Required. Every 1095 days (3 years).
AWS owned CMK	No	No	No	Varies

To distinguish customer managed CMKs from AWS managed CMKs, use the `KeyManager` field in the DescribeKey operation response. For customer managed CMKs, the `KeyManager` value is `Customer`. For AWS managed CMKs, the `KeyManager` value is `AWS`.

AWS services that integrate with AWS KMS differ in their support for CMKs. Some AWS services encrypt your data by default with an AWS owned CMK or an AWS managed CMK. Other AWS services offer to encrypt your data under a customer managed CMK that you choose. And other AWS services support all types of CMKs to allow you the ease of an AWS owned CMK, the visibility of an AWS managed CMK, or the control of a customer managed CMK. For detailed information about the encryption options that an AWS service offers, see the *Encryption at Rest* topic in the user guide or the developer guide for the service.

via - https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#master_keys

Incorrect options:

KMS receives CMK from the client at every encrypt call, and encrypts the data with that - You can import your own CMK (Customer Master Key) but it is done once and then you can encrypt/decrypt as needed.

KMS sends the CMK to the client, which performs the encryption and then deletes the CMK - KMS does not send CMK to the client, KMS itself encrypts, and then decrypts the data.

KMS generates a new CMK for each Encrypt call and encrypts the data with it - KMS does not generate a new key each time but you can have KMS rotate the keys for you. Best practices discourage extensive reuse of encryption keys so it is good practice to generate new keys.

References:

https://docs.aws.amazon.com/kms/latest/developerguide/concepts.html#master_keys

https://docs.aws.amazon.com/kms/latest/developerguide/rotate-keys.html

Question 34: **Incorrect**

A development team lead is configuring policies for his team at an IT company.

Which of the following policy types only limit permissions but cannot grant permissions (Select two)?

- [x] **Resource-based policy** **(Incorrect)**
- [ ] **Identity-based policy**
- [ ] **AWS Organizations Service Control Policy (SCP)** **(Correct)**
- [x] **Permissions boundary** **(Correct)**
- [ ] **Access control list (ACL)**

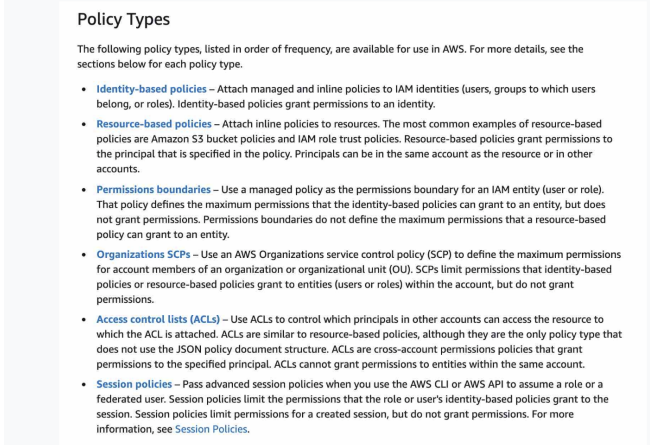
Explanation

Correct options:

AWS Organizations Service Control Policy (SCP) – Use an AWS Organizations Service Control Policy (SCP) to define the maximum permissions for account members of an organization or organizational unit (OU). SCPs limit permissions that identity-based policies or resource-based policies grant to entities (users or roles) within the account, but do not grant permissions.

Permissions boundary - Permissions boundary is a managed policy that is used for an IAM entity (user or role). The policy defines the maximum permissions that the identity-based policies can grant to an entity, but does not grant permissions.

Overview of Policy Types:



Policy Types

The following policy types, listed in order of frequency, are available for use in AWS. For more details, see the sections below for each policy type.

- **Identity-based policies** – Attach managed and inline policies to IAM identities (users, groups to which users belong, or roles). Identity-based policies grant permissions to an identity.
- **Resource-based policies** – Attach inline policies to resources. The most common examples of resource-based policies are Amazon S3 bucket policies and IAM role trust policies. Resource-based policies grant permissions to the principal that is specified in the policy. Principals can be in the same account as the resource or in other accounts.
- **Permissions boundaries** – Use a managed policy as the permissions boundary for an IAM entity (user or role). That policy defines the maximum permissions that the identity-based policies can grant to an entity, but does not grant permissions. Permissions boundaries do not define the maximum permissions that a resource-based policy can grant to an entity.
- **Organizations SCPs** – Use an AWS Organizations service control policy (SCP) to define the maximum permissions for account members of an organization or organizational unit (OU). SCPs limit permissions that identity-based policies or resource-based policies grant to entities (users or roles) within the account, but do not grant permissions.
- **Access control lists (ACLs)** – Use ACLs to control which principals in other accounts can access the resource to which the ACL is attached. ACLs are similar to resource-based policies, although they are the only policy type that does not use the JSON policy document structure. ACLs are cross-account permissions policies that grant permissions to the specified principal. ACLs cannot grant permissions to entities within the same account.
- **Session policies** – Pass advanced session policies when you use the AWS CLI or AWS API to assume a role or a federated user. Session policies limit the permissions that the role or user's identity-based policies grant to the session. Session policies limit permissions for a created session, but do not grant permissions. For more information, see Session Policies.

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

Incorrect options:

Access control list (ACL) - Use ACLs to control which principals in other accounts can access the resource to which the ACL is attached. ACLs are similar to resource-based policies, although they are the only policy type that does not use the JSON policy document structure. ACLs are cross-account permissions policies that grant permissions to the specified principal.

Resource-based policy - Resource-based policies grant permissions to the principal that is specified in the policy. Principals can be in the same account as the resource or in other accounts. The most common examples of resource-based policies are Amazon S3 bucket policies and IAM role trust policies.

Identity-based policy - Help attach managed and inline policies to IAM identities (users, groups to which users belong, or roles). Identity-based policies grant permissions to an identity.

Reference:

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

Question 35: **Incorrect**

You are a developer in a manufacturing company that has several servers on-site. The company decides to move new development to the cloud using serverless technology. You decide to use the AWS Serverless Application Model (AWS SAM) and work with an AWS SAM template file to represent your serverless architecture.

Which of the following is NOT a valid serverless resource type?

- ○ **AWS::Serverless::UserPool** (Correct)
- ○ **AWS::Serverless::Api**
- ○ **AWS::Serverless::Function**
- ◉ **AWS::Serverless::SimpleTable** (Incorrect)

Explanation
Correct option:

AWS::Serverless::UserPool

The AWS Serverless Application Model (SAM) is an open-source framework for building serverless applications. It provides shorthand syntax to express functions, APIs, databases, and event source mappings. With just a few lines per resource, you can define the application you want and model it using YAML.

SAM supports the following resource types:

AWS::Serverless::Api

AWS::Serverless::Application

AWS::Serverless::Function

AWS::Serverless::HttpApi

AWS::Serverless::LayerVersion

AWS::Serverless::SimpleTable

AWS::Serverless::StateMachine

UserPool applies to the Cognito service which is used for authentication for mobile app and web. There is no resource named UserPool in the Serverless Application Model.

Incorrect options:

AWS::Serverless::Function - This resource creates a Lambda function, IAM execution role, and event source mappings that trigger the function.

AWS::Serverless::Api - This creates a collection of Amazon API Gateway resources and methods that can be invoked through HTTPS endpoints. It is useful for advanced use cases where you want full control and flexibility when you configure your APIs.

AWS::Serverless::SimpleTable - This creates a DynamoDB table with a single attribute primary key. It is useful when data only needs to be accessed via a primary key.

Reference:

https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/sam-specification-resources-and-properties.html

Question 36: **Correct**

A company uses Elastic Beanstalk to manage its IT infrastructure on AWS Cloud and it would like to deploy the new application version to the EC2 instances. When the deployment is executed, some instances should serve requests with the old application version, while other instances should serve requests using the new application version until the deployment is completed.

Which deployment meets this requirement without incurring additional costs?

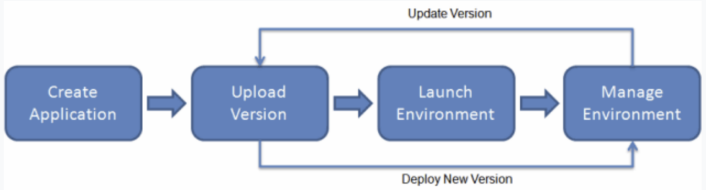
- (●) **Rolling** (Correct)
- ( ) **Immutable**
- ( ) **All at once**
- ( ) **Rolling with additional batches**

Explanation
Correct option:

Rolling

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.

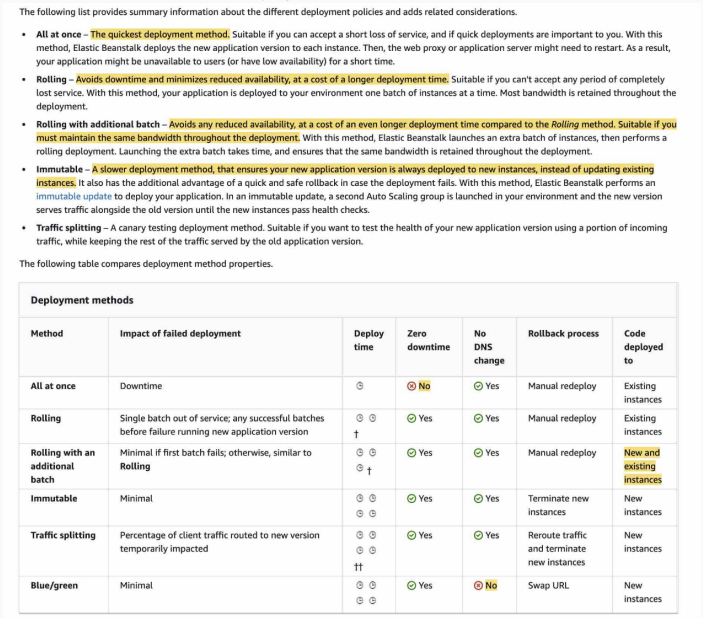
How Elastic BeanStalk Works:



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

The rolling deployment policy deploys the new version in batches. Each batch is taken out of service during the deployment phase, reducing your environment's capacity by the number of instances in a batch. The cost remains the same as the number of EC2 instances does not increase. This policy avoids downtime and minimizes reduced availability, at a cost of a longer deployment time.

Overview of Elastic Beanstalk Deployment Policies:



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

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

The following table compares deployment method properties.

Deployment methods

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

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

Incorrect options:

Immutable - The 'Immutable' deployment policy ensures that your new application version is always deployed to new instances, instead of updating existing instances. It also has the additional advantage of a quick and safe rollback in case the deployment fails.

All at once - This policy deploys the new version to all instances simultaneously. Although 'All at once' is the quickest deployment method, but the application may become unavailable to users (or have low availability) for a short time.

Rolling with additional batches - This policy deploys the new version in batches, but first launches a new batch of instances to ensure full capacity during the deployment process. This policy avoids any reduced availability, at a cost of an even longer deployment time compared to the Rolling method. Suitable if you must maintain the same bandwidth throughout the deployment. These increase the costs as you're adding extra instances during the deployment.

Reference:

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

Question 37: **Incorrect**

Which of the following security credentials can only be created by the AWS Account root user?

- ◯ **CloudFront Key Pairs** (Correct)
- ◯ **EC2 Instance Key Pairs**
- ◯ **IAM User Access Keys**
- ◉ **IAM User passwords** (Incorrect)

Explanation
Correct option:

For Amazon CloudFront, you use key pairs to create signed URLs for private content, such as when you want to distribute restricted content that someone paid for.

CloudFront Key Pairs - IAM users can't create CloudFront key pairs. You must log in using root credentials to create key pairs.

To create signed URLs or signed cookies, you need a signer. A signer is either a trusted key group that you create in CloudFront, or an AWS account that contains a CloudFront key pair. AWS recommends that you use trusted key groups with signed URLs and signed cookies instead of using CloudFront key pairs.

Incorrect options:

EC2 Instance Key Pairs - You use key pairs to access Amazon EC2 instances, such as when you use SSH to log in to a Linux instance. These key pairs can be created from the IAM user login and do not need root user access.

IAM User Access Keys - Access keys consist of two parts: an access key ID and a secret access key. You use access keys to sign programmatic requests that you make to AWS if you use AWS CLI commands (using the SDKs) or using AWS API operations. IAM users can create their own Access Keys, does not need root access.

IAM User passwords - Every IAM user has access to his own credentials and can reset the password whenever they need to.

References:

https://docs.aws.amazon.com/general/latest/gr/aws-sec-cred-types.html#access-keys-and-secret-access-keys

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-trusted-signers.html

Question 38: **Correct**

As a developer, you are working on creating an application using AWS Cloud Development Kit (CDK).

Which of the following represents the correct order of steps to be followed for creating an app using AWS CDK?

- ○ **Create the app from a template provided by AWS CDK -> Add code to the app to create resources within stacks -> Synthesize one or more stacks in the app -> Deploy stack(s) to your AWS account -> Build the app**
- ○ **Create the app from a template provided by AWS CloudFormation -> Add code to the app to create resources within stacks -> Build the app (optional) -> Synthesize one or more stacks in the app -> Deploy stack(s) to your AWS account**
- ◉ **Create the app from a template provided by AWS CDK -> Add code to the app to create resources within stacks -> Build the app (optional) -> Synthesize one or more stacks in the app -> Deploy stack(s) to your AWS account** (Correct)
- ○ **Create the app from a template provided by AWS CloudFormation -> Add code to the app to create resources within stacks -> Synthesize one or more stacks in the app -> Deploy stack(s) to your AWS account -> Build the app**

Explanation

Correct option:

Create the app from a template provided by AWS CDK -> Add code to the app to create resources within stacks -> Build the app (optional) -> Synthesize one or more stacks in the app -> Deploy stack(s) to your AWS account

The standard AWS CDK development workflow is similar to the workflow you're already familiar as a developer. There are a few extra steps:

1. Create the app from a template provided by AWS CDK - Each AWS CDK app should be in its own directory, with its own local module dependencies. Create a new directory for your app. Now initialize the app using the `cdk init` command, specifying the desired template ("app") and programming language. The `cdk init` command creates a number of files and folders inside the created home directory to help you organize the source code for your AWS CDK app.
2. Add code to the app to create resources within stacks - Add custom code as is needed for your application.
3. Build the app (optional) - In most programming environments, after making changes to your code, you'd build (compile) it. This isn't strictly necessary with the AWS CDK—the Toolkit does it for you so you can't forget. But you can still build manually whenever you want to catch syntax and type errors.
4. Synthesize one or more stacks in the app to create an AWS CloudFormation template - Synthesize one or more stacks in the app to create an AWS CloudFormation template. The synthesis step catches logical errors in defining your AWS resources. If your app contains more than one stack, you'd need to specify which stack(s) to synthesize.

5. Deploy one or more stacks to your AWS account - It is optional (though good practice) to synthesize before deploying. The AWS CDK synthesizes your stack before each deployment. If your code has security implications, you'll see a summary of these and need to confirm them before deployment proceeds. `cdk deploy` is used to deploy the stack using CloudFormation templates. This command displays progress information as your stack is deployed. When it's done, the command prompt reappears.

Incorrect options:

Create the app from a template provided by AWS CloudFormation -> Add code to the app to create resources within stacks -> Build the app (optional) -> Synthesize one or more stacks in the app -> Deploy stack(s) to your AWS account

Create the app from a template provided by AWS CloudFormation -> Add code to the app to create resources within stacks -> Synthesize one or more stacks in the app -> Deploy stack(s) to your AWS account -> Build the app

For both these options, you cannot use AWS CloudFormation to create the app. So these options are incorrect.

Create the app from a template provided by AWS CDK -> Add code to the app to create resources within stacks -> Synthesize one or more stacks in the app -> Deploy stack(s) to your AWS account -> Build the app - You cannot have the build step after deployment. So this option is incorrect.

Reference:

https://docs.aws.amazon.com/cdk/latest/guide/hello_world.html

Question 39: **Correct**

You are a developer for a web application written in .NET which uses the AWS SDK. You need to implement an authentication mechanism that returns a JWT (JSON Web Token).

Which AWS service will help you with token handling and management?

- ○ **API Gateway**
- ○ **Cognito Sync**
- ◉ **Cognito User Pools** **(Correct)**
- ○ **Cognito Identity Pools**

Explanation
Correct option:

"Cognito User Pools"

After successful authentication, Amazon Cognito returns user pool tokens to your app. You can use the tokens to grant your users access to your own server-side resources, or to the Amazon API Gateway.

Amazon Cognito user pools implement ID, access, and refresh tokens as defined by the OpenID Connect (OIDC) open standard.

The ID token is a JSON Web Token (JWT) that contains claims about the identity of the authenticated user such as name, email, and phone_number. You can use this identity information inside your application. The ID token can also be used to authenticate users against your resource servers or server applications.

Using Tokens with User Pools

PDF | Kindle

After a successful authentication, Amazon Cognito returns user pool tokens to your app. You can use the tokens to grant your users access to your own server-side resources, or to the Amazon API Gateway. Or, you can exchange them for temporary AWS credentials to access other AWS services. See Common Amazon Cognito Scenarios.

User pool token handling and management for your web or mobile app is provided on the client side through Amazon Cognito SDKs. For information on the SDKs, and sample code for JavaScript, Android, and iOS see Amazon Cognito User Pool SDKs. If you need to manually process tokens for server-side API processing, or if you are using other programming languages, there are many good libraries for decoding and verifying a JWT. See the OpenID Foundation list of libraries for working with JWT tokens;

Amazon Cognito user pools implements ID, access, and refresh tokens as defined by the OpenID Connect (OIDC) open standard:

- The ID Token contains claims about the identity of the authenticated user such as `name`, `email`, and `phone_number`.
- The Access Token contains scopes and groups and is used to grant access to authorized resources.
- The Refresh Token contains the information necessary to obtain a new ID or access token.

via - https://docs.aws.amazon.com/cognito/latest/developerguide/amazon-cognito-user-pools-using-tokens-with-identity-providers.html

Incorrect options:

"API Gateway" - If you are processing tokens server-side and using other programming languages not supported in AWS it may be a good choice. Other than that, go with a service already providing the functionality.

"Cognito Identity Pools" - You can use Identity pools to grant your users access to other AWS services. 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 specific identity providers that you can use to authenticate users for identity pools.

"Cognito Sync" - Amazon Cognito Sync is an AWS service and client library that enables cross-device syncing of application-related user data. You can use it to synchronize user profile data across mobile devices and the web without requiring your own backend.

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.

Question 40: **Correct**

You are a developer working on AWS Lambda functions that are invoked via REST API's using Amazon API Gateway. Currently, when a GET request is invoked by the consumer, the entire data-set returned by the Lambda function is visible. Your team lead asked you to format the data response.

Which feature of the API Gateway can be used to solve this issue?

- ◯ **Use an API Gateway stage variable**
- ◉ **Use API Gateway Mapping Templates** **(Correct)**
- ◯ **Deploy an interceptor shell script**
- ◯ **Use a Lambda custom interceptor**

Explanation

Correct option:

Use API Gateway Mapping Templates - In API Gateway, an API's method request can take a payload in a different format from the corresponding integration request payload, as required in the backend. Similarly, vice versa is also possible. API Gateway lets you use mapping templates to map the payload from a method request to the corresponding integration request and from an integration response to the corresponding method response.

Suppose we have an API for managing fruit and vegetable inventory in the produce department of a supermarket. When a manager queries the backend for the current inventory, the server sends back the following response payload:


```
{
  "department": "produce",
  "categories": [
    "fruit",
    "vegetables"
  ],
  "bins": [
    {
      "category": "fruit",
      "type": "apples",
      "price": 1.99,
      "unit": "pound",
      "quantity": 232
    },
    {
      "category": "fruit",
      "type": "bananas",
      "price": 0.19,
      "unit": "each",
      "quantity": 112
    },
    {
      "category": "vegetables",
      "type": "carrots",
      "price": 1.29,
      "unit": "bag",
      "quantity": 57
    }
  ]
}
```

When the backend returns the query results shown above, the manager of the produce department might be interested in reading them, as follows:

```
{
  "choices": [
    {
      "kind": "apples",
      "suggestedPrice": "1.99 per pound",
      "available": 232
    },
    {
      "kind": "bananas",
      "suggestedPrice": "0.19 per each",
      "available": 112
    },
    {
      "kind": "carrots",
      "suggestedPrice": "1.29 per bag",
      "available": 57
    }
  ]
}
```

To enable this, we need to provide API Gateway with a mapping template to translate the data from the backend format like so:

```
#set($inputRoot = $input.path('$'))
{
  "choices": [
#foreach($elem in $inputRoot.bins)
    {
      "kind": "$elem.type",
      "suggestedPrice": "$elem.price per $elem.unit",
      "available": $elem.quantity
    }#if($foreach.hasNext),#end

#end
  ]
}
```

Incorrect options:

Deploy an interceptor shell script - This option has been added as a distractor.

Use an API Gateway stage variable - Stage variables are name-value pairs that you can define as configuration attributes associated with a deployment stage of a REST API. They act like environment variables and can be used in your API setup and mapping templates. This feature is not useful for the current use case.

Use a Lambda custom interceptor - This is a made-up option. Lambda cannot intercept the response for the given use-case.

References:

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

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

Question 41: **Correct**

The manager at an IT company wants to set up member access to user-specific folders in an Amazon S3 bucket - `bucket-a`. So, user x can only access files in his folder - `bucket-a/user/user-x/` and user y can only access files in her folder - `bucket-a/user/user-y/` and so on.

As a Developer Associate, which of the following IAM constructs would you recommend so that the policy snippet can be made generic for all team members and the manager does not need to create separate IAM policy for each team member?

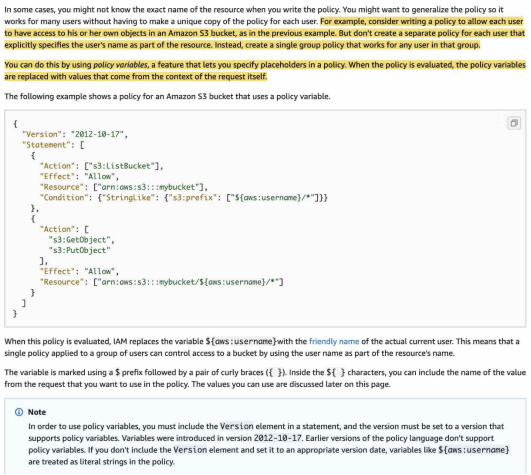
- ◯ **IAM policy principal**
- ◉ **IAM policy variables** **(Correct)**
- ◯ **IAM policy resource**
- ◯ **IAM policy condition**

Explanation
Correct option:

IAM policy variables

Instead of creating individual policies for each user, you can use policy variables and create a single policy that applies to multiple users (a group policy). Policy variables act as placeholders. When you make a request to AWS, the placeholder is replaced by a value from the request when the policy is evaluated.

As an example, the following policy gives each of the users in the group full programmatic access to a user-specific object (their own "home directory") in Amazon S3.



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

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

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

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

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

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

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

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

Incorrect options:

IAM policy principal - You can use the Principal element in a policy to specify the principal that is allowed or denied access to a resource (In IAM, a principal is a person or application that can make a request for an action or operation on an AWS resource. The principal is authenticated as the AWS account root user or an IAM entity to make requests to AWS). You cannot use the Principal element in an IAM identity-based policy. You can use it in the trust policies for IAM roles and in resource-based policies.

IAM policy condition - The Condition element (or Condition block) lets you specify conditions for when a policy is in effect, like so - `"Condition" : { "StringEquals" : { "aws:username" : "johndoe" }}`. This can not be used to address the requirements of the given use-case.

IAM policy resource - The Resource element specifies the object or objects that the statement covers. You specify a resource using an ARN. This can not be used to address the requirements of the given use-case.

Reference:

https://aws.amazon.com/blogs/security/writing-iam-policies-grant-access-to-user-specific-folders-in-an-amazon-s3-bucket/

Question 42: **Correct**

An application is hosted by a 3rd party and exposed at yourapp.3rdparty.com. You would like to have your users access your application using www.mydomain.com, which you own and manage under Route 53.

What Route 53 record should you create?

- ○ **Create an Alias Record**
- ◉ **Create a CNAME record** **(Correct)**
- ○ **Create an A record**
- ○ **Create a PTR record**

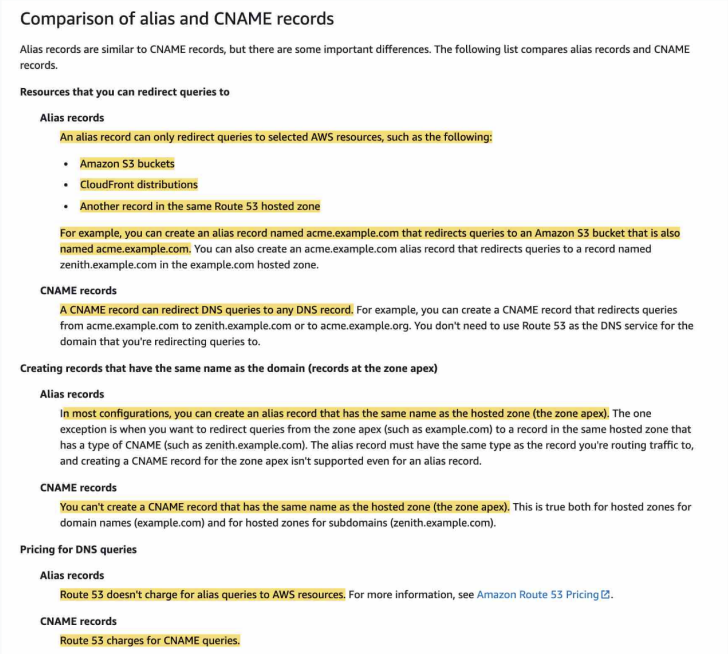
Explanation
Correct option:

Create a CNAME record

A CNAME record maps DNS queries for the name of the current record, such as acme.example.com, to another domain (example.com or example.net) or subdomain (acme.example.com or zenith.example.org).

CNAME records can be used to map one domain name to another. Although you should keep in mind that the DNS protocol does not allow you to create a CNAME record for the top node of a DNS namespace, also known as the zone apex. For example, if you register the DNS name example.com, the zone apex is example.com. You cannot create a CNAME record for example.com, but you can create CNAME records for www.example.com, newproduct.example.com, and so on.

Please review the major differences between CNAME and Alias Records:



Comparison of alias and CNAME records

Alias records are similar to CNAME records, but there are some important differences. The following list compares alias records and CNAME records.

Resources that you can redirect queries to

Alias records

An alias record can only redirect queries to selected AWS resources, such as the following:

- Amazon S3 buckets
- CloudFront distributions
- Another record in the same Route 53 hosted zone

For example, you can create an alias record named acme.example.com that redirects queries to an Amazon S3 bucket that is also named acme.example.com. You can also create an acme.example.com alias record that redirects queries to a record named zenith.example.com in the example.com hosted zone.

CNAME records

A CNAME record can redirect DNS queries to any DNS record. For example, you can create a CNAME record that redirects queries from acme.example.com to zenith.example.com or to acme.example.org. You don't need to use Route 53 as the DNS service for the domain that you're redirecting queries to.

Creating records that have the same name as the domain (records at the zone apex)

Alias records

In most configurations, you can create an alias record that has the same name as the hosted zone (the zone apex). The one exception is when you want to redirect queries from the zone apex (such as example.com) to a record in the same hosted zone that has a type of CNAME (such as zenith.example.com). The alias record must have the same type as the record you're routing traffic to, and creating a CNAME record for the zone apex isn't supported even for an alias record.

CNAME records

You can't create a CNAME record that has the same name as the hosted zone (the zone apex). This is true both for hosted zones for domain names (example.com) and for hosted zones for subdomains (zenith.example.com).

Pricing for DNS queries

Alias records

Route 53 doesn't charge for alias queries to AWS resources. For more information, see Amazon Route 53 Pricing.

CNAME records

Route 53 charges for CNAME queries.

via - https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-choosing-alias-non-alias.html

Incorrect options:

Create an A record - Used to point a domain or subdomain to an IP address. 'A record' cannot be used to map one domain name to another.

Create a PTR record - A Pointer (PTR) record resolves an IP address to a fully-qualified domain name (FQDN) as an opposite to what A record does. PTR records are also called Reverse DNS records. 'PTR record' cannot be used to map one domain name to another.

Create an Alias Record - Alias records let you route traffic to selected AWS resources, such as CloudFront distributions and Amazon S3 buckets. They also let you route traffic from one record in a hosted zone to another record. 3rd party websites do not qualify for these as we have no control over those. 'Alias record' cannot be used to map one domain name to another.

Reference:

https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-choosing-alias-non-alias.html

Question 43: **Correct**

You are running workloads on AWS and have embedded RDS database connection strings within each web server hosting your applications. After failing a security audit, you are looking at a different approach to store your secrets securely and automatically rotate the database credentials.

Which AWS service can you use to address this use-case?

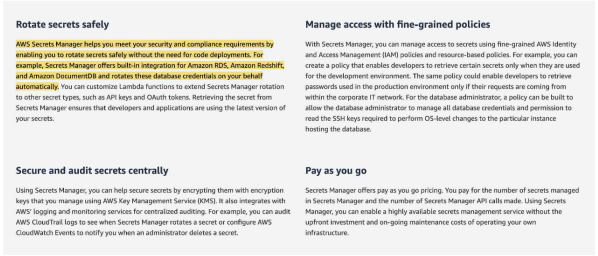
- ◯ **Systems Manager**
- ◉ **Secrets Manager** **(Correct)**
- ◯ **KMS**
- ◯ **SSM Parameter Store**

Explanation
Correct option:

Secrets Manager

AWS Secrets Manager enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Users and applications retrieve secrets with a call to Secrets Manager APIs, eliminating the need to hardcode sensitive information in plain text. Secrets Manager offers secret rotation with built-in integration for Amazon RDS, Amazon Redshift, and Amazon DocumentDB.

Benefits of Secrets Manager:



via - https://aws.amazon.com/secrets-manager/

Incorrect options:

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. SSM Parameter Store cannot be used to automatically rotate the database credentials.

Systems Manager - AWS Systems Manager gives you visibility and control of your infrastructure on AWS. Systems Manager provides a unified user interface so you can view operational data from multiple AWS services and allows you to automate operational tasks across your AWS resources. Systems Manager cannot be used to store your secrets securely and automatically rotate the database credentials.

KMS - AWS Key Management Service (KMS) makes it easy for you to create and manage cryptographic keys and control their use across a wide range of AWS services and in your applications. KMS cannot be used to store your secrets securely and automatically rotate the database credentials.

References:

https://aws.amazon.com/secrets-manager/

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

Question 44: **Correct**

As an AWS Certified Developer Associate, you have configured the AWS CLI on your workstation. Your default region is us-east-1 and your IAM user has permissions to operate commands on services such as EC2, S3 and RDS in any region. You would like to execute a command to stop an EC2 instance in the us-east-2 region.

What of the following is the MOST optimal solution to address this use-case?

- ○ **You should create a new IAM user just for that other region**
- ○ **You need to override the default region by using aws configure**
- ○ **Use boto3 dependency injection**
- ◉ **Use the --region parameter** **(Correct)**

Explanation
Correct option:

Use the --region parameter: If the region parameter is not set, then the CLI command is executed against the default AWS region.

You can also review all general options for AWS CLI:

General Options

The AWS CLI has a few general options:

Variable	Option	Config Entry	Environment Variable	Description
profile	--profile	N/A	AWS_PROFILE	Default profile name
region	--region	region	AWS_DEFAULT_REGION	Default AWS Region
output	--output	output	AWS_DEFAULT_OUTPUT	Default output style
cli_timestamp_format	N/A	cli_timestamp_format	N/A	Output format of timestamps
cli_follow_urlparam	N/A	cli_follow_urlparam	N/A	Fetch URL url parameters
ca_bundle	--ca-bundle	ca_bundle	AWS_CA_BUNDLE	CA Certificate Bundle
parameter_validation	N/A	parameter_validation	N/A	Toggles parameter validation
tcp_keepalive	N/A	tcp_keepalive	N/A	Toggles TCP Keep-Alive
max_attempts	N/A	max_attempts	AWS_MAX_ATTEMPTS	Number of total requests
retry_mode	N/A	retry_mode	AWS_RETRY_MODE	Type of retries performed

via - https://docs.aws.amazon.com/cli/latest/topic/config-vars.html#general-options

Incorrect options:

You need to override the default region by using aws configure - This is not the most optimal way as you will have to change it again to reset the default region.

You should create a new IAM user just for that other region - This is not the most optimal way as you would need to manage two IAM user profiles.

Use boto3 dependency injection - With the CLI you do not use boto3. This option is a distractor.

Reference:

https://docs.aws.amazon.com/cli/latest/topic/config-vars.html#general-options

Question 45: **Correct**

The development team at a company creates serverless solutions using AWS Lambda. Functions are invoked by clients via AWS API Gateway which anyone can access. The team lead would like to control access using a 3rd party authorization mechanism.

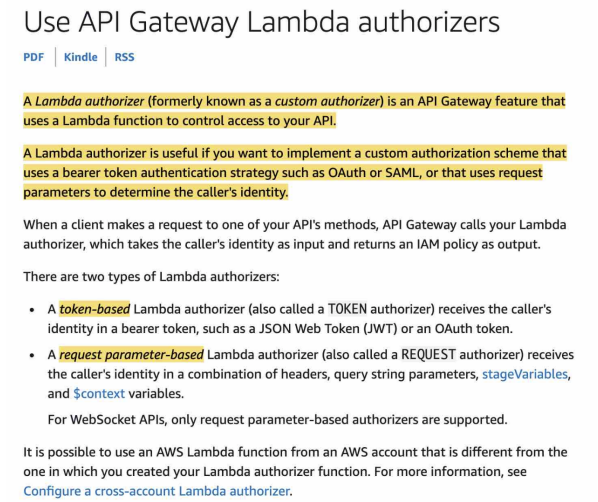
As a Developer Associate, which of the following options would you recommend for the given use-case?

- ◯ **API Gateway User Pools**
- ◯ **Cognito User Pools**
- ◯ **IAM permissions with sigv4**
- ◉ **Lambda Authorizer** **(Correct)**

Explanation
Correct option:

"Lambda Authorizer"

An Amazon API Gateway Lambda authorizer (formerly known as a custom authorizer) is a Lambda function that you provide to control access to your API. A Lambda authorizer uses bearer token authentication strategies, such as OAuth or SAML. Before creating an API Gateway Lambda authorizer, you must first create the AWS Lambda function that implements the logic to authorize and, if necessary, to authenticate the caller.



Use API Gateway Lambda authorizers

PDF | Kindle | RSS

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.

When a client makes a request to one of your API's methods, API Gateway calls your Lambda authorizer, which takes the caller's identity as input and returns an IAM policy as output.

There are two types of Lambda authorizers:

- A *token-based* Lambda authorizer (also called a `TOKEN` authorizer) receives the caller's identity in a bearer token, such as a JSON Web Token (JWT) or an OAuth token.
- A *request parameter-based* Lambda authorizer (also called a `REQUEST` authorizer) receives the caller's identity in a combination of headers, query string parameters, stageVariables, and $context variables.

  For WebSocket APIs, only request parameter-based authorizers are supported.

It is possible to use an AWS Lambda function from an AWS account that is different from the one in which you created your Lambda authorizer function. For more information, see Configure a cross-account Lambda authorizer.

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

Incorrect options:

"IAM permissions with sigv4" - Signature Version 4 is the process to add authentication information to AWS requests sent by HTTP. You will still need to provide permissions but our requirements have a need for 3rd party authentication which is where Lambda Authorizer comes in to play.

"Cognito User Pools" - 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. This is managed by AWS, therefore, does not meet our requirements.

"API Gateway User Pools" - This is a made-up option, added as a distractor.

Reference:

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

Question 46: **Correct**

You are creating a Cloud Formation template to deploy your CMS application running on an EC2 instance within your AWS account. Since the application will be deployed across multiple regions, you need to create a map of all the possible values for the base AMI.

How will you invoke the `!FindInMap` function to fulfill this use case?

- (●) `!FindInMap [ MapName, TopLevelKey, SecondLevelKey ]` **(Correct)**
- ( ) `!FindInMap [ MapName, TopLevelKey, SecondLevelKey, ThirdLevelKey ]`
- ( ) `!FindInMap [ MapName, TopLevelKey ]`
- ( ) `!FindInMap [ MapName ]`

Explanation
Correct option:

`!FindInMap [ MapName, TopLevelKey, SecondLevelKey ]` - The intrinsic function Fn::FindInMap returns the value corresponding to keys in a two-level map that is declared in the Mappings section. YAML Syntax for the full function name: Fn::FindInMap: [MapName, TopLevelKey, SecondLevelKey]

Short form of the above syntax is : !FindInMap [MapName, TopLevelKey, SecondLevelKey]

Where,

MapName - Is the logical name of a mapping declared in the Mappings section that contains the keys and values. TopLevelKey - The top-level key name. Its value is a list of key-value pairs. SecondLevelKey - The second-level key name, which is set to one of the keys from the list assigned to TopLevelKey.

Consider the following YAML template:

```
Mappings:
  RegionMap:
    us-east-1:
      HVM64: "ami-0ff8a91507f77f867"
      HVMG2: "ami-0a584ac55a7631c0c"
    us-west-1:
      HVM64: "ami-0bdb828fd58c52235"
      HVMG2: "ami-066ee5fd4a9ef77f1"
    eu-west-1:
      HVM64: "ami-047bb4163c506cd98"
      HVMG2: "ami-31c2f645"
    ap-southeast-1:
      HVM64: "ami-08569b978cc4dfa10"
      HVMG2: "ami-0be9df32ae9f92309"
    ap-northeast-1:
      HVM64: "ami-06cd52961ce9f0d85"
      HVMG2: "ami-053cdd503598e4a9d"
Resources:
  myEC2Instance:
    Type: "AWS::EC2::Instance"
    Properties:
      ImageId: !FindInMap
        - RegionMap
        - !Ref 'AWS::Region'
        - HVM64
      InstanceType: m1.small
```

The example template contains an AWS::EC2::Instance resource whose ImageId property is set by the FindInMap function.

MapName is set to the map of interest, "RegionMap" in this example. TopLevelKey is set to the region where the stack is created, which is determined by using the "AWS::Region" pseudo parameter. SecondLevelKey is set to the desired architecture, "HVM64" for this example.

FindInMap returns the AMI assigned to FindInMap. For a HVM64 instance in us-east-1, FindInMap would return "ami-0ff8a91507f77f867".

Incorrect options:

`!FindInMap [ MapName, TopLevelKey ]`

`!FindInMap [ MapName ]`

`!FindInMap [ MapName, TopLevelKey, SecondLevelKey, ThirdLevelKey ]`

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

Reference:

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/intrinsic-function-reference-findinmap.html

Question 47: **Correct**

A startup with newly created AWS account is testing different EC2 instances. They have used Burstable performance instance - T2.micro - for 35 seconds and stopped the instance.

At the end of the month, what is the instance usage duration that the company is charged for?

- ○ **35 seconds**
- ○ **30 seconds**
- ○ **60 seconds**
- ◉ **0 seconds** (Correct)

Explanation

Correct option:

Burstable performance instances, which are T3, T3a, and T2 instances, are designed to provide a baseline level of CPU performance with the ability to burst to a higher level when required by your workload. Burstable performance instances are the only instance types that use credits for CPU usage.

0 seconds - AWS states that, if your AWS account is less than 12 months old, you can use a t2.micro instance for free within certain usage limits.

Incorrect options:

35 seconds

60 seconds

30 seconds

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

Reference:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/burstable-performance-instances.html

Question 48: **Correct**

A multi-national company has multiple business units with each unit having its own AWS account. The development team at the company would like to debug and trace data across accounts and visualize it in a centralized account.

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

- ○ **CloudWatch Events**
- ○ **CloudTrail**
- ○ **VPC Flow Logs**
- ◉ **X-Ray** **(Correct)**

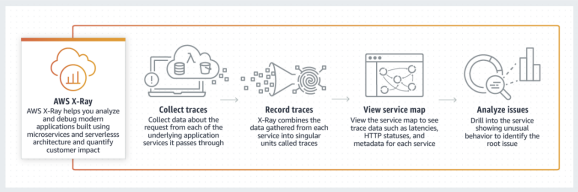
Explanation
Correct option:

X-Ray

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

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

How X-Ray Works:



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

Incorrect options:

VPC Flow Logs: VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC. Flow log data is used to analyze network traces and helps with network security. Flow log data can be published to Amazon CloudWatch Logs or Amazon S3. You cannot use VPC Flow Logs to debug and trace data across accounts.

CloudWatch Events: Amazon CloudWatch Events delivers a near real-time stream of system events that describe changes in Amazon Web Services (AWS) resources. These help to trigger notifications based on changes happening in AWS services. You cannot use CloudWatch Events to debug and trace data across accounts.

CloudTrail: With CloudTrail, you can log, continuously monitor, and retain account activity related to actions across your AWS infrastructure. You can use AWS CloudTrail to answer questions such as - "Who made an API call to modify this resource?". CloudTrail provides event history of your AWS account activity thereby enabling governance, compliance, operational auditing, and risk auditing of your AWS account. You cannot use CloudTrail to debug and trace data across accounts.

Reference:

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

Question 49: **Correct**

A firm runs its technology operations on a fleet of Amazon EC2 instances. The firm needs a certain software to be available on the instances to support their daily workflows. The developer team has been told to use the user data feature of EC2 instances.

Which of the following are true about the user data EC2 configuration? (Select two)

- [ ] **When an instance is running, you can update user data by using root user credentials**
- [ ] **By default, user data is executed every time an EC2 instance is re-started**
- [x] **By default, scripts entered as user data are executed with root user privileges** (Correct)
- [x] **By default, user data runs only during the boot cycle when you first launch an instance** (Correct)
- [ ] **By default, scripts entered as user data do not have root user privileges for executing**

Explanation

Correct options:

User Data is generally used to perform common automated configuration tasks and even run scripts after the instance starts. When you launch an instance in Amazon EC2, you can pass two types of user data - shell scripts and cloud-init directives. You can also pass this data into the launch wizard as plain text or as a file.

By default, scripts entered as user data are executed with root user privileges - Scripts entered as user data are executed as the root user, hence do not need the sudo command in the script. Any files you create will be owned by root; if you need non-root users to have file access, you should modify the permissions accordingly in the script.

By default, user data runs only during the boot cycle when you first launch an instance - By default, user data scripts and cloud-init directives run only during the boot cycle when you first launch an instance. You can update your configuration to ensure that your user data scripts and cloud-init directives run every time you restart your instance.

Incorrect options:

By default, user data is executed every time an EC2 instance is re-started - As discussed above, this is not a default configuration of the system. But, can be achieved by explicitly configuring the instance.

When an instance is running, you can update user data by using root user credentials - You can't change the user data if the instance is running (even by using root user credentials), but you can view it.

By default, scripts entered as user data do not have root user privileges for executing - Scripts entered as user data are executed as the root user, hence do not need the sudo command in the script.

Reference:

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

Question 50: **Incorrect**

An IT company is configuring Auto Scaling for its Amazon EC2 instances spread across different AZs and Regions.

Which of the following scenarios are NOT correct about EC2 Auto Scaling? (Select two)

- [ ] **Auto Scaling groups that span across multiple Regions need to be enabled for all the Regions specified** (Correct)
- [x] **An Auto Scaling group can contain EC2 instances in one or more Availability Zones within the same Region** (Incorrect)
- [x] **An Auto Scaling group can contain EC2 instances in only one Availability Zone of a Region** (Correct)
- [ ] **Amazon EC2 Auto Scaling attempts to distribute instances evenly between the Availability Zones that are enabled for your Auto Scaling group**
- [ ] **For Auto Scaling groups in a VPC, the EC2 instances are launched in subnets**

Explanation
Correct options:

Auto Scaling groups that span across multiple Regions need to be enabled for all the Regions specified - This is not valid for Auto Scaling groups. Auto Scaling groups cannot span across multiple Regions.

An Auto Scaling group can contain EC2 instances in only one Availability Zone of a Region - This is not valid for Auto Scaling groups. An Auto Scaling group can contain EC2 instances in one or more Availability Zones within the same Region.

Amazon EC2 Auto Scaling Overview:

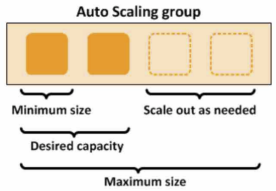
Amazon EC2 Auto Scaling Overview:

What Is Amazon EC2 Auto Scaling?

PDF | Kindle | RSS

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

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



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

Incorrect options:

An Auto Scaling group can contain EC2 instances in one or more Availability Zones within the same Region - This is a valid statement. Auto Scaling groups can span across the availability Zones of a Region.

Amazon EC2 Auto Scaling attempts to distribute instances evenly between the Availability Zones that are enabled for your Auto Scaling group - When one Availability Zone becomes unhealthy or unavailable, Auto Scaling launches new instances in an unaffected Availability Zone. When the unhealthy Availability Zone returns to a healthy state, Auto Scaling automatically redistributes the application instances evenly across all of the designated Availability Zones.

For Auto Scaling groups in a VPC, the EC2 instances are launched in subnets - For Auto Scaling groups in a VPC, the EC2 instances are launched in subnets. Customers can select the subnets for your EC2 instances when you create or update the Auto Scaling group.

References:

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

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

Question 51: **Correct**

To enable HTTPS connections for his web application deployed on the AWS Cloud, a developer is in the process of creating server certificate.

Which AWS entities can be used to deploy SSL/TLS server certificates? (Select two)

- [ ] **AWS CloudFormation**
- [ ] **AWS Systems Manager**
- [x] **AWS Certificate Manager** (Correct)
- [x] **IAM** (Correct)
- [ ] **AWS Secrets Manager**

Explanation

Correct options:

AWS Certificate Manager - AWS Certificate Manager (ACM) is the preferred tool to provision, manage, and deploy server certificates. With ACM you can request a certificate or deploy an existing ACM or external certificate to AWS resources. Certificates provided by ACM are free and automatically renew. In a supported Region, you can use ACM to manage server certificates from the console or programmatically.

IAM - IAM is used as a certificate manager only when you must support HTTPS connections in a Region that is not supported by ACM. IAM securely encrypts your private keys and stores the encrypted version in IAM SSL certificate storage. IAM supports deploying server certificates in all Regions, but you must obtain your certificate from an external provider for use with AWS. You cannot upload an ACM certificate to IAM. Additionally, you cannot manage your certificates from the IAM Console.

Incorrect options:

AWS 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. Users and applications retrieve secrets with a call to Secrets Manager APIs, eliminating the need to hardcode sensitive information in plain text. It cannot be used to discover and protect your sensitive data in AWS.

AWS Systems Manager - AWS Systems Manager gives you visibility and control of your infrastructure on AWS. Systems Manager provides a unified user interface so you can view operational data from multiple AWS services and allows you to automate operational tasks such as running commands, managing patches, and configuring servers across AWS Cloud as well as on-premises infrastructure.

AWS CloudFormation - AWS CloudFormation allows you to use programming languages or a simple text file to model and provision, in an automated and secure manner, all the resources needed for your applications across all Regions and accounts. Think infrastructure as code; think CloudFormation. You cannot use CloudFormation for running commands or managing patches on servers.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_server-certs.html

Question 52: **Incorrect**

CodeCommit is a managed version control service that hosts private Git repositories in the AWS cloud.

Which of the following credential types is NOT supported by IAM for CodeCommit?

- (●) **SSH Keys** **(Incorrect)**
- ( ) **IAM username and password** **(Correct)**
- ( ) **AWS Access Keys**
- ( ) **Git credentials**

Explanation
Correct option:

IAM username and password - IAM username and password credentials cannot be used to access CodeCommit.

Incorrect options:

Git credentials - These are IAM -generated user name and password pair you can use to communicate with CodeCommit repositories over HTTPS.

SSH Keys - Are locally generated public-private key pair that you can associate with your IAM user to communicate with CodeCommit repositories over SSH.

AWS access keys - You can use these keys with the credential helper included with the AWS CLI to communicate with CodeCommit repositories over HTTPS.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_credentials_ssh-keys.html

Question 53: **Correct**

A developer is configuring a bucket policy that denies upload object permission to any requests that do not include the `x-amz-server-side-encryption` header requesting server-side encryption with SSE-KMS for an Amazon S3 bucket - `examplebucket`.

Which of the following policies is the right fit for the given requirement?

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

(Correct)

( )
```
{
   "Version":"2012-10-17",
   "Id":"PutObjectPolicy",
   "Statement":[{
         "Sid":"DenyUnEncryptedObjectUploads",
         "Effect":"Deny",
         "Principal":"*",
         "Action":"s3:GetObject",
         "Resource":"arn:aws:s3:::examplebucket/*",
         "Condition":{
            "StringNotEquals":{
               "s3:x-amz-server-side-encryption":"aws:AES256"
            }
         }
      }
   ]
}
```

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

○

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

Explanation
Correct option:

{ "Version":"2012-10-17", "Id":"PutObjectPolicy", "Statement":[{ "Sid":"DenyUnEncryptedObjectUploads", "Effect":"Deny", "Principal":"*", *"Action":"s3:PutObject", "Resource":"arn:aws:s3:::examplebucket/*",* "Condition":{ "StringNotEquals":{ "s3:x-amz-server-side-encryption":"aws:kms" } } }] } - This bucket policy denies upload object (s3:PutObject) permission if the request does not include the x-amz-server-side-encryption header requesting server-side encryption with SSE-KMS. To ensure that a particular AWS KMS CMK be used to encrypt the objects in a bucket, you can use the `s3:x-amz-server-side-encryption-aws-kms-key-id` condition key. To specify the AWS KMS CMK, you must use a key Amazon Resource Name (ARN) that is in the "arn:aws:kms:region:acct-id:key/key-id" format.

When you upload an object, you can specify the AWS KMS CMK using the `x-amz-server-side-encryption-aws-kms-key-id` header. If the header is not present in the request, Amazon S3 assumes the AWS-managed CMK.

Incorrect options:

{ "Version":"2012-10-17", "Id":"PutObjectPolicy", "Statement":[{ "Sid":"DenyUnEncryptedObjectUploads", "Effect":"Deny", "Principal":"*", *"Action":"s3:PutObject", "Resource":"arn:aws:s3:::examplebucket/*",* "Condition":{ "StringEquals":{ "s3:x-amz-server-side-encryption":"aws:kms" } } }] } - The condition is incorrect in this policy. The condition should use StringNotEquals.

{ "Version":"2012-10-17", "Id":"PutObjectPolicy", "Statement":[{ "Sid":"DenyUnEncryptedObjectUploads", "Effect":"Deny", "Principal":"*", *"Action":"s3:GetObject", "Resource":"arn:aws:s3:::examplebucket/*",* "Condition":{ "StringNotEquals":{ "s3:x-amz-server-side-encryption":"aws:AES256" } } }] } - AES256 is used for Amazon S3-managed encryption keys (SSE-S3). Amazon S3 server-side encryption uses one of the strongest block ciphers available to encrypt your data, 256-bit Advanced Encryption Standard (AES-256).

{ "Version":"2012-10-17", "Id":"PutObjectPolicy", "Statement":[{ "Sid":"DenyUnEncryptedObjectUploads", "Effect":"Deny", "Principal":"*", *"Action":"s3:PutObject", "Resource":"arn:aws:s3:::examplebucket/*",* "Condition":{ "StringNotEquals":{ "s3:x-amz-server-side-encryption":"false" } } }] } - The condition is incorrect in this policy. The condition should use `"s3:x-amz-server-side-encryption":"aws:kms"`.

Reference:

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

Question 54: **Correct**

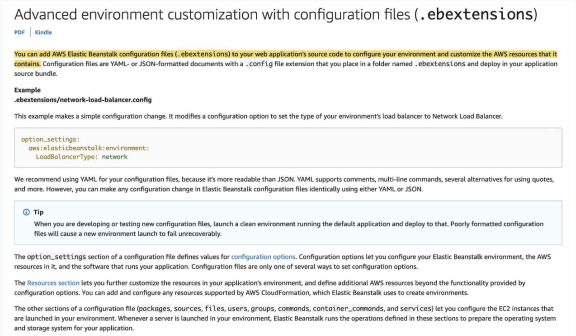
You have created a Java application that uses RDS for its main data storage and ElastiCache for user session storage. The application needs to be deployed using Elastic Beanstalk and every new deployment should allow the application servers to reuse the RDS database. On the other hand, user session data stored in ElastiCache can be re-created for every deployment.

Which of the following configurations will allow you to achieve this? (Select two)

- [ ] **ElastiCache database defined externally and referenced through environment variables**
- [x] **RDS database defined externally and referenced through environment variables** **(Correct)**
- [ ] **ElastiCache bundled with the application source code**
- [x] **ElastiCache defined in `.ebextensions/`** **(Correct)**
- [ ] **RDS database defined in `.ebextensions/`**

Explanation
Correct option:

ElastiCache defined in `.ebextensions/` - Any resources created as part of your `.ebextensions` is part of your Elastic Beanstalk template and will get deleted if the environment is terminated.



Advanced environment customization with configuration files (`.ebextensions`)

PDF | Kindle

You can add AWS Elastic Beanstalk configuration files (`.ebextensions`) to your web application's source code to configure your environment and customize the AWS resources that it contains. Configuration files are YAML- or JSON-formatted documents with a `.config` file extension that you place in a folder named `.ebextensions` and deploy in your application source bundle.

Example
`.ebextensions/network-load-balancer.config`

This example makes a simple configuration change. It modifies a configuration option to set the type of your environment's load balancer to Network Load Balancer.

```
option_settings:
  aws:elasticbeanstalk:environment:
    LoadBalancerType: network
```

We recommend using YAML for your configuration files, because it's more readable than JSON. YAML supports comments, multi-line commands, several alternatives for using quotes, and more. However, you can make any configuration change in Elastic Beanstalk configuration files identically using either YAML or JSON.

Tip
When you are developing or testing new configuration files, launch a clean environment running the default application and deploy to that. Poorly formatted configuration files will cause a new environment launch to fail unrecoverably.

The `option_settings` section of a configuration file defines values for configuration options. Configuration options let you configure your Elastic Beanstalk environment, the AWS resources in it, and the software that runs your application. Configuration files are only one of several ways to set configuration options.

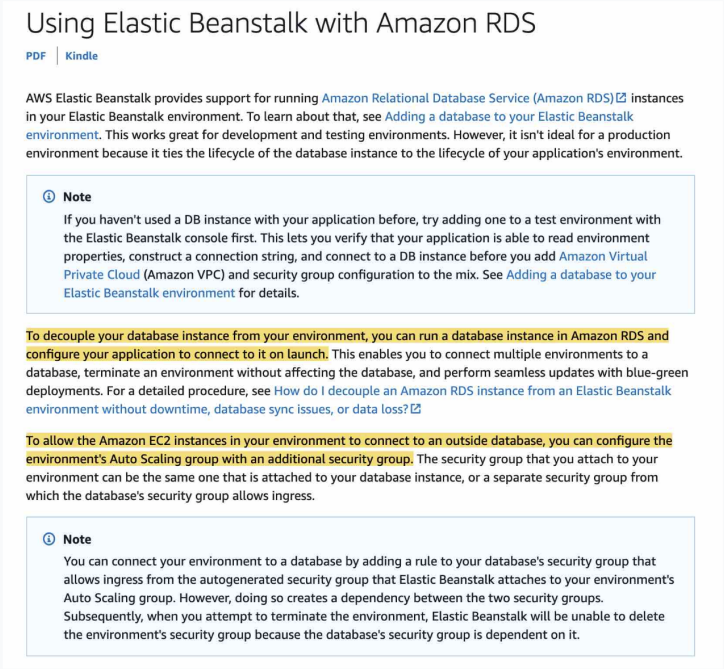
The Resources section lets you further customize the resources in your application's environment, and define additional AWS resources beyond the functionality provided by configuration options. You can add and configure any resources supported by AWS CloudFormation, which Elastic Beanstalk uses to create environments.

The other sections of a configuration file (`packages`, `sources`, `files`, `users`, `groups`, `commands`, `container_commands`, and `services`) let you configure the EC2 instances that are launched in your environment. Whenever a server is launched in your environment, Elastic Beanstalk runs the operations defined in these sections to prepare the operating system and storage system for your application.

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

RDS database defined externally and referenced through environment variables - To decouple your database instance from your environment, you can run a database instance in Amazon RDS and configure your application to connect to it on launch. This enables you to connect multiple environments to a database, terminate an environment without affecting the database, and perform seamless updates with blue-green deployments. To allow the Amazon EC2 instances in your environment to connect to an outside database, you can configure the environment's Auto Scaling group with an additional security group.

Using Elastic Beanstalk with Amazon RDS:



Using Elastic Beanstalk with Amazon RDS

PDF | Kindle

AWS Elastic Beanstalk provides support for running Amazon Relational Database Service (Amazon RDS) instances in your Elastic Beanstalk environment. To learn about that, see Adding a database to your Elastic Beanstalk environment. This works great for development and testing environments. However, it isn't ideal for a production environment because it ties the lifecycle of the database instance to the lifecycle of your application's environment.

Note

If you haven't used a DB instance with your application before, try adding one to a test environment with the Elastic Beanstalk console first. This lets you verify that your application is able to read environment properties, construct a connection string, and connect to a DB instance before you add Amazon Virtual Private Cloud (Amazon VPC) and security group configuration to the mix. See Adding a database to your Elastic Beanstalk environment for details.

To decouple your database instance from your environment, you can run a database instance in Amazon RDS and configure your application to connect to it on launch. This enables you to connect multiple environments to a database, terminate an environment without affecting the database, and perform seamless updates with blue-green deployments. For a detailed procedure, see How do I decouple an Amazon RDS instance from an Elastic Beanstalk environment without downtime, database sync issues, or data loss?

To allow the Amazon EC2 instances in your environment to connect to an outside database, you can configure the environment's Auto Scaling group with an additional security group. The security group that you attach to your environment can be the same one that is attached to your database instance, or a separate security group from which the database's security group allows ingress.

Note

You can connect your environment to a database by adding a rule to your database's security group that allows ingress from the autogenerated security group that Elastic Beanstalk attaches to your environment's Auto Scaling group. However, doing so creates a dependency between the two security groups. Subsequently, when you attempt to terminate the environment, Elastic Beanstalk will be unable to delete the environment's security group because the database's security group is dependent on it.

via - https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/AWSHowTo.RDS.html

Incorrect options:

ElastiCache bundled with the application source code - ElastiCache is an AWS service and cannot be bundled with the source code.

RDS database defined in `.ebextensions/` - The lifetime of the RDS instance gets tied to the lifetime of the Elastic Beanstalk environment, so this option is incorrect.

ElastiCache database defined externally and referenced through environment variables - For the given use-case, the client is fine with losing user session data and hence defining it in .ebextensions/ is more appropriate.

References:

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/AWSHowTo.RDS.html

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

https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/customize-environment-resources-elasticache.html

Question 55: **Correct**

A company has built its technology stack on AWS serverless architecture for managing all its business functions. To expedite development for a new business requirement, the company is looking at using pre-built serverless applications.

Which AWS service represents the easiest solution to address this use-case?

- (●) **AWS Serverless Application Repository (SAR)** (Correct)
- ( ) **AWS AppSync**
- ( ) **AWS Service Catalog**
- ( ) **AWS Marketplace**

Explanation

Correct option:

AWS Serverless Application Repository (SAR) - The AWS Serverless Application Repository is a managed repository for serverless applications. It enables teams, organizations, and individual developers to store and share reusable applications, and easily assemble and deploy serverless architectures in powerful new ways. Using the Serverless Application Repository, you don't need to clone, build, package, or publish source code to AWS before deploying it. Instead, you can use pre-built applications from the Serverless Application Repository in your serverless architectures, helping you and your teams reduce duplicated work, ensure organizational best practices, and get to market faster. Integration with AWS Identity and Access Management (IAM) provides resource-level control of each application, enabling you to publicly share applications with everyone or privately share them with specific AWS accounts.

Deploying applications using SAR:

via - https://aws.amazon.com/serverless/serverlessrepo/

Incorrect options:

AWS Marketplace - The AWS Marketplace enables qualified partners to market and sell their software to AWS Customers. AWS Marketplace is an online software store that helps customers find, buy, and immediately start using the software and services that run on AWS. AWS Marketplace is designed for Independent Software Vendors (ISVs), Value-Added Resellers (VARs), and Systems Integrators (SIs) who have software products they want to offer to customers in the cloud.

AWS AppSync - AWS AppSync is a fully managed service that makes it easy to develop GraphQL APIs by handling the heavy lifting of securely connecting to data sources like AWS DynamoDB, Lambda, and more. Organizations choose to build APIs with GraphQL because it helps them develop applications faster, by giving front-end developers the ability to query multiple databases, microservices, and APIs with a single GraphQL endpoint.

AWS Service Catalog - AWS Service Catalog allows organizations to create and manage catalogs of IT services that are approved for use on AWS. These IT services can include everything from virtual machine images, servers, software, and databases to complete multi-tier application architectures. AWS Service Catalog allows you to centrally manage deployed IT services and your applications, resources, and metadata. This helps you achieve consistent governance and meet your compliance requirements while enabling users to quickly deploy only the approved IT services they need.

Reference:

https://aws.amazon.com/serverless/serverlessrepo/

Question 56: **Correct**

An E-commerce business, has its applications built on a fleet of Amazon EC2 instances, spread across various Regions and AZs. The technical team has suggested using Elastic Load Balancers for better architectural design.

What characteristics of an Elastic Load Balancer make it a winning choice? (Select two)

- [ ] **Improve vertical scalability of the system**
- [x] **Separate public traffic from private traffic** (Correct)
- [ ] **Deploy EC2 instances across multiple AWS Regions**
- [ ] **The Load Balancer communicates with the underlying EC2 instances using their public IPs**
- [x] **Build a highly available system** (Correct)

Explanation
Correct options:

A load balancer accepts incoming traffic from clients and routes requests to its registered targets (such as EC2 instances) in one or more Availability Zones. The load balancer also monitors the health of its registered targets and ensures that it routes traffic only to healthy targets. When the load balancer detects an unhealthy target, it stops routing traffic to that target. It then resumes routing traffic to that target when it detects that the target is healthy again.

Elastic Load Balancing supports three types of load balancers:

Application Load Balancers

Network Load Balancers

Classic Load Balancers

Separate public traffic from private traffic - The nodes of an internet-facing load balancer have public IP addresses. Load balancers route requests to your targets using private IP addresses. Therefore, your targets do not need public IP addresses to receive requests from users over the internet.

Build a highly available system - Elastic Load Balancing provides fault tolerance for your applications by automatically balancing traffic across targets – Amazon EC2 instances, containers, IP addresses, and Lambda functions – in multiple Availability Zones while ensuring only healthy targets receive traffic.

Incorrect options:

The Load Balancer communicates with the underlying EC2 instances using their public IPs - This is an incorrect statement. The Load Balancer communicates with the underlying EC2 instances using their private IPs.

Improve vertical scalability of the system - This is an incorrect statement. Elastic Load Balancers can connect with Auto Scaling groups to provide horizontal scaling.

Deploy EC2 instances across multiple AWS Regions - A Load Balancer can target EC2 instances only within an AWS Region.

References:

https://aws.amazon.com/elasticloadbalancing/

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

Question 57: **Correct**

A developer has been asked to create a web application to be deployed on EC2 instances. The developer just wants to focus on writing application code without worrying about server provisioning, configuration and deployment.

As a Developer Associate, which AWS service would you recommend for the given use-case?

- ○ **CodeDeploy**
- ○ **CloudFormation**
- ◉ **Elastic Beanstalk** **(Correct)**
- ○ **Serverless Application Model**

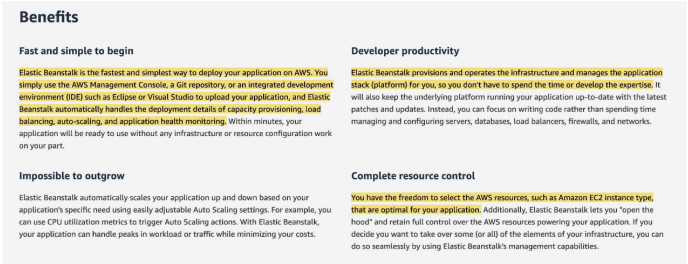
Explanation
Correct option:

Elastic Beanstalk

AWS Elastic Beanstalk provides an environment to easily deploy and run applications in the cloud. It is integrated with developer tools and provides a one-stop experience for you to manage the lifecycle of your applications.

AWS Elastic Beanstalk lets you manage all of the resources that run your application as environments where each environment runs only a single application version at a time. When an environment is being created, Elastic Beanstalk provisions all the required resources needed to run the application version. You don't need to worry about server provisioning, configuration, and deployment as that's taken care of by Beanstalk.

Benefits of Elastic Beanstalk:



Benefits

Fast and simple to begin

Elastic Beanstalk is the fastest and simplest way to deploy your application on AWS. You simply use the AWS Management Console, a Git repository, or an integrated development environment (IDE) such as Eclipse or Visual Studio to upload your application, and Elastic Beanstalk automatically handles the deployment details of capacity provisioning, load balancing, auto-scaling, and application health monitoring. Within minutes, your application will be ready to use without any infrastructure or resource configuration work on your part.

Developer productivity

Elastic Beanstalk provisions and operates the infrastructure and manages the application stack (platform) for you, so you don't have to spend the time or develop the expertise. It will also keep the underlying platform running your application up-to-date with the latest patches and updates. Instead, you can focus on writing code rather than spending time managing and configuring servers, databases, load balancers, firewalls, and networks.

Impossible to outgrow

Elastic Beanstalk automatically scales your application up and down based on your application's specific need using easily adjustable Auto Scaling settings. For example, you can use CPU utilization metrics to trigger Auto Scaling actions. With Elastic Beanstalk, your application can handle peaks in workload or traffic while minimizing your costs.

Complete resource control

You have the freedom to select the AWS resources, such as Amazon EC2 instance type, that are optimal for your application. Additionally, Elastic Beanstalk lets you "open the hood" and retain full control over the AWS resources powering your application. If you decide you want to take over some (or all) of the elements of your infrastructure, you can do so seamlessly by using Elastic Beanstalk's management capabilities.

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

Incorrect options:

CloudFormation - AWS CloudFormation is a service that gives developers and businesses an easy way to create a collection of related AWS and third-party resources and provision them in an orderly and predictable fashion. With CloudFormation, you still need to create a template to specify the type of resources you need, hence this option is not correct.

How CloudFormation Works:

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

CodeDeploy - AWS CodeDeploy is a fully managed deployment service that automates software deployments to a variety of compute services such as Amazon EC2, AWS Fargate, AWS Lambda, and your on-premises servers. AWS CodeDeploy makes it easier for you to rapidly release new features, helps you avoid downtime during application deployment, and handles the complexity of updating your applications. It can deploy an application to an instance but it cannot provision the instance.

Serverless Application Model - The AWS Serverless Application Model (AWS SAM) is an open-source framework for building serverless applications. It provides shorthand syntax to express functions, APIs, databases, and event source mappings. You define the application you want with just a few lines per resource and model it using YAML. As the web application needs to be deployed on EC2 instances, so this option is ruled out.

References:

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

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

Question 58: **Correct**

Amazon Simple Queue Service (SQS) has a set of APIs for various actions supported by the service.

As a developer associate, which of the following would you identify as correct regarding the `CreateQueue` API? (Select two)

- [ ] **Queue tags are case insensitive. A new tag with a key identical to that of an existing tag overwrites the existing tag**
- [ ] **The dead-letter queue of a FIFO queue must also be a FIFO queue. Whereas, the dead-letter queue of a standard queue can be a standard queue or a FIFO queue**
- [x] **You can't change the queue type after you create it** (Correct)
- [x] **The visibility timeout value for the queue is in seconds, which defaults to 30 seconds** (Correct)
- [ ] **The length of time, in seconds, for which the delivery of all messages in the queue is delayed is configured using `MessageRetentionPeriod` attribute**

Explanation

Correct options:

You can't change the queue type after you create it - You can't change the queue type after you create it and you can't convert an existing standard queue into a FIFO queue. You must either create a new FIFO queue for your application or delete your existing standard queue and recreate it as a FIFO queue.

The visibility timeout value for the queue is in seconds, which defaults to 30 seconds - The visibility timeout for the queue is in seconds. Valid values are: An integer from 0 to 43,200 (12 hours), the Default value is 30.

Incorrect options:

The dead-letter queue of a FIFO queue must also be a FIFO queue. Whereas, the dead-letter queue of a standard queue can be a standard queue or a FIFO queue - The dead-letter queue of a FIFO queue must also be a FIFO queue. Similarly, the dead-letter queue of a standard queue must also be a standard queue.

The length of time, in seconds, for which the delivery of all messages in the queue is delayed is configured using `MessageRetentionPeriod` attribute - The length of time, in seconds, for which the delivery of all messages in the queue is delayed is configured using `DelaySeconds` attribute. `MessageRetentionPeriod` attribute controls the length of time, in seconds, for which Amazon SQS retains a message.

Queue tags are case insensitive. A new tag with a key identical to that of an existing tag overwrites the existing tag - Queue tags are case-sensitive. A new tag with a key identical to that of an existing tag overwrites the existing tag. To be able to tag a queue on creation, you must have the `sqs:CreateQueue` and `sqs:TagQueue` permissions.

Reference:

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

Question 59: **Incorrect**

A global e-commerce company wants to perform geographic load testing of its order processing API. The company must deploy resources to multiple AWS Regions to support the load testing of the API.

How can the company address these requirements without additional application code?

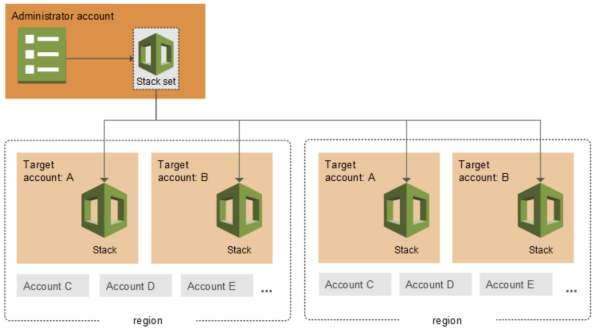
- ○ **Set up an AWS Organizations template that defines the load test resources across the organization. Leverage the AWS CLI create-stack-set command to create a stack set in the desired Regions**
- ○ **Set up an AWS CloudFormation template that defines the load test resources. Leverage the AWS CLI create-stack-set command to create a stack set in the desired Regions** (Correct)
- ○ **Set up an AWS CloudFormation template that defines the load test resources. Develop region-specific Lambda functions to create a stack from the AWS CloudFormation template in each Region when the respective function is invoked**
- ◉ **Set up an AWS Cloud Development Kit (CDK) ToolKit that defines the load test resources. Leverage the CDK CLI to create a stack from the template in each Region** (Incorrect)

Explanation

Correct option:

Set up an AWS CloudFormation template that defines the load test resources. Leverage the AWS CLI create-stack-set command to create a stack set in the desired Regions

AWS CloudFormation StackSets extends the capability of stacks by enabling you to create, update, or delete stacks across multiple accounts and AWS Regions with a single operation. Using an administrator account, you define and manage an AWS CloudFormation template, and use the template as the basis for provisioning stacks into selected target accounts across specified AWS Regions.



via - https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/what-is-cfnstacksets.html

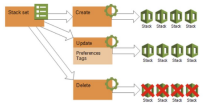
A stack set lets you create stacks in AWS accounts across regions by using a single CloudFormation template. A stack set's CloudFormation template defines all the resources in each stack. As you create the stack set, specify the template to use, in addition to any parameters and capabilities that the template requires.

After you've defined a stack set, you can create, update, or delete stacks in the target accounts and AWS Regions you specify.

Stack instances

A *stack instance* is a reference to a stack in a target account within a Region. A stack instance can exist without a stack. For example, if the stack couldn't be created for some reason, the stack instance shows the reason for stack creation failure. A stack instance associates with only one stack set.

The following figure shows the logical relationships between stack sets, stack operations, and stacks. When you update a stack set, all associated stack instances update throughout all accounts and Regions.



Stack set operations

You can perform the following operations on stack sets.

Create stack set

Creating a new stack set includes specifying a CloudFormation template that you want to use to create stacks, specifying the target accounts in which you want to create stacks, and identifying the AWS Regions in which you want to deploy stacks in your target accounts. A stack set ensures consistent deployment of the same stack resources, with the same settings, to all specified target accounts within the Regions you choose.

Update stack set

When you update a stack set, you push changes out to stacks in your stack set. You can update a stack set in one of the following ways. Your template updates always affect all stacks; you can't selectively update the template for some stacks in the stack set, but not others.

- Change existing settings in the template or add new resources, such as updating parameter settings for a specific service, or adding new Amazon EC2 instances.
- Replace the template with a different template.
- Add stacks in existing or additional target accounts, across existing or additional Regions.

Delete stacks

When you delete stacks, you are removing a stack and all its associated resources from the target accounts you specify, within the Regions you specify. You can delete stacks in the following ways.

- Delete stacks from some target accounts, while leaving other stacks in other target accounts running.
- Delete stacks from some Regions, while leaving stacks in other Regions running.
- Delete stacks from your stack set, but save them so they continue to run independently of your stack set by choosing the **Retain Stacks** option. You can then manage retained stacks outside of your stack set in AWS CloudFormation.
- Delete all stacks in your stack set, in preparation for deleting your entire stack set.

Delete stack set

You can delete your stack set only when there are no stack instances in it.

via - https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacksets-concepts.html

Incorrect options:

Set up an AWS CloudFormation template that defines the load test resources. Develop region-specific Lambda functions to create a stack from the AWS CloudFormation template in each Region when the respective function is invoked - If you do not use a stack set, then you need to define the CloudFormation templates in each region as well as develop lambda functions in each region to create a stack from the corresponding CloudFormation template. This is unnecessary bloat that can be avoided by simply using the CloudFormation StackSets.

Set up an AWS Cloud Development Kit (CDK) ToolKit that defines the load test resources. Leverage the CDK CLI to create a stack from the template in each Region - AWS CDK is a framework for defining cloud infrastructure in code and provisioning it through AWS CloudFormation. The CDK Toolkit again poses regional limitations and is not the right fit for the given use case.

Set up an AWS Organizations template that defines the load test resources across the organization. Leverage the AWS CLI create-stack-set command to create a stack set in the desired Regions - This option acts as a distractor. AWS Organizations cannot be used to create templates for provisioning AWS infrastructure. AWS Organizations is an account management service that enables you to consolidate multiple AWS accounts into an organization that you create and centrally manage.

References:

https://docs.aws.amazon.com/organizations/latest/userguide/orgs_introduction.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacksets-concepts.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/what-is-cfnstacksets.html

Question 60: **Correct**

A development team lead is responsible for managing access for her IAM principals. At the start of the cycle, she has granted excess privileges to users to keep them motivated for trying new things. She now wants to ensure that the team has only the minimum permissions required to finish their work.

Which of the following will help her identify unused IAM roles and remove them without disrupting any service?

- (●) **Access Advisor feature on IAM console** (Correct)
- ( ) **AWS Trusted Advisor**
- ( ) **Amazon Inspector**
- ( ) **IAM Access Analyzer**

Explanation

Correct option:

Access Advisor feature on IAM console- To help identify the unused roles, IAM reports the last-used timestamp that represents when a role was last used to make an AWS request. Your security team can use this information to identify, analyze, and then confidently remove unused roles. This helps improve the security posture of your AWS environments. Additionally, by removing unused roles, you can simplify your monitoring and auditing efforts by focusing only on roles that are in use.

Incorrect options:

AWS Trusted Advisor - AWS Trusted Advisor is an online tool that provides you real-time guidance to help you provision your resources following AWS best practices on cost optimization, security, fault tolerance, service limits, and performance improvement.

IAM Access Analyzer - AWS IAM Access Analyzer helps you identify the resources in your organization and accounts, such as Amazon S3 buckets or IAM roles, that are shared with an external entity. This lets you identify unintended access to your resources and data, which is a security risk.

Amazon Inspector - Amazon Inspector is an automated security assessment service that helps improve the security and compliance of applications deployed on AWS. Amazon Inspector automatically assesses applications for exposure, vulnerabilities, and deviations from best practices.

Reference:

https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_access-advisor-view-data.html

Question 61: **Correct**

A media company has created a video streaming application and it would like their Brazilian users to be served by the company's Brazilian servers. Other users around the globe should not be able to access the servers through DNS queries.

Which Route 53 routing policy meets this requirement?

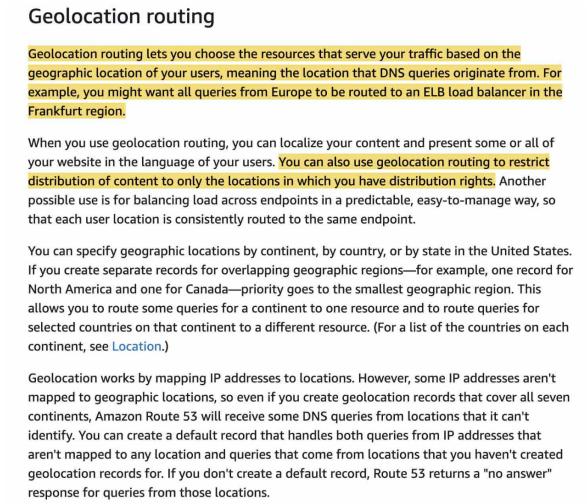
- ○ **Weighted**
- ○ **Failover**
- ○ **Latency**
- ◉ **Geolocation** (Correct)

Explanation
Correct option:

Geolocation

Geolocation routing lets you choose the resources that serve your traffic based on the geographic location of your users, meaning the location that DNS queries originate from. For example, you might want all queries from Europe to be routed to an ELB load balancer in the Frankfurt region. You can also use geolocation routing to restrict distribution of content to only the locations in which you have distribution rights

You can create a default record that handles both queries from IP addresses that aren't mapped to any location and queries that come from locations that you haven't created geolocation records for. If you don't create a default record, Route 53 returns a "no answer" response for queries from those locations.



Geolocation routing

Geolocation routing lets you choose the resources that serve your traffic based on the geographic location of your users, meaning the location that DNS queries originate from. For example, you might want all queries from Europe to be routed to an ELB load balancer in the Frankfurt region.

When you use geolocation routing, you can localize your content and present some or all of your website in the language of your users. You can also use geolocation routing to restrict distribution of content to only the locations in which you have distribution rights. Another possible use is for balancing load across endpoints in a predictable, easy-to-manage way, so that each user location is consistently routed to the same endpoint.

You can specify geographic locations by continent, by country, or by state in the United States. If you create separate records for overlapping geographic regions—for example, one record for North America and one for Canada—priority goes to the smallest geographic region. This allows you to route some queries for a continent to one resource and to route queries for selected countries on that continent to a different resource. (For a list of the countries on each continent, see Location.)

Geolocation works by mapping IP addresses to locations. However, some IP addresses aren't mapped to geographic locations, so even if you create geolocation records that cover all seven continents, Amazon Route 53 will receive some DNS queries from locations that it can't identify. You can create a default record that handles both queries from IP addresses that aren't mapped to any location and queries that come from locations that you haven't created geolocation records for. If you don't create a default record, Route 53 returns a "no answer" response for queries from those locations.

Choosing a routing policy

PDF | Kindle | RSS

When you create a record, you choose a routing policy, which determines how Amazon Route 53 responds to queries:

- **Simple routing policy** – Use for a single resource that performs a given function for your domain, for example, a web server that serves content for the example.com website.
- **Failover routing policy** – Use when you want to configure active-passive failover.
- **Geolocation routing policy** – Use when you want to route traffic based on the location of your users.
- **Geoproximity routing policy** – Use when you want to route traffic based on the location of your resources and, optionally, shift traffic from resources in one location to resources in another.
- **Latency routing policy** – Use when you have resources in multiple AWS Regions and you want to route traffic to the region that provides the best latency.
- **Multivalue answer routing policy** – Use when you want Route 53 to respond to DNS queries with up to eight healthy records selected at random.
- **Weighted routing policy** – Use to route traffic to multiple resources in proportions that you specify.

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

Incorrect options:

Failover - Failover routing lets you route traffic to a resource when the resource is healthy or to a different resource when the first resource is unhealthy.

Latency - If your application is hosted in multiple AWS Regions, you can improve performance for your users by serving their requests from the AWS Region that provides the lowest latency.

Weighted - Use this policy to route traffic to multiple resources in proportions that you specify.

Reference:

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

Question 62: **Correct**

A Developer at a company is working on a CloudFormation template to set up resources. Resources will be defined using code and provisioned based on certain conditions defined in the `Conditions` section.

Which section of a CloudFormation template cannot be associated with `Condition`?

- ◯ **Conditions**
- ◯ **Outputs**
- ◯ **Resources**
- ◉ **Parameters** (Correct)

Explanation
Correct option:

Parameters

Parameters enable you to input custom values to your CloudFormation template each time you create or update a stack. Please see this note to understand how to define a parameter in a template:

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.
```

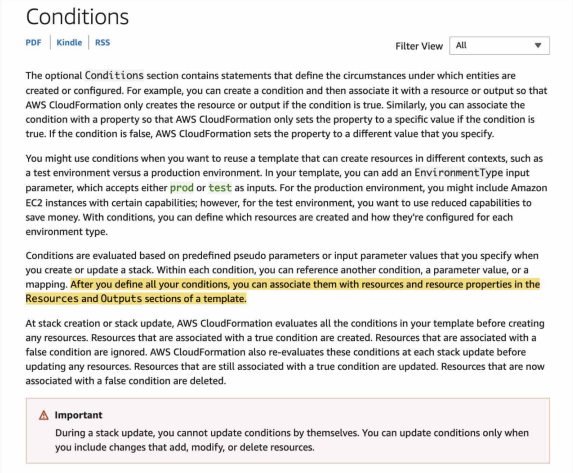
via - https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html

The optional Conditions section contains statements that define the circumstances under which entities are created or configured. For example, you can create a condition and then associate it with a resource or output so that AWS CloudFormation only creates the resource or output if the condition is true.

You might use conditions when you want to reuse a template that can create resources in different contexts, such as a test environment versus a production environment. In your template, you can add an EnvironmentType input parameter, which accepts either prod or test as inputs. For the production environment, you might include Amazon EC2 instances with certain capabilities; however, for the test environment, you want to use reduced capabilities to save money.

Conditions cannot be used within the Parameters section. After you define all your conditions, you can associate them with resources and resource properties only in the Resources and Outputs sections of a template.

Please review this note for more details:



Conditions

PDF | Kindle | RSS

Filter View: All

The optional `Conditions` section contains statements that define the circumstances under which entities are created or configured. For example, you can create a condition and then associate it with a resource or output so that AWS CloudFormation only creates the resource or output if the condition is true. Similarly, you can associate the condition with a property so that AWS CloudFormation only sets the property to a specific value if the condition is true. If the condition is false, AWS CloudFormation sets the property to a different value that you specify.

You might use conditions when you want to reuse a template that can create resources in different contexts, such as a test environment versus a production environment. In your template, you can add an `EnvironmentType` input parameter, which accepts either `prod` or `test` as inputs. For the production environment, you might include Amazon EC2 instances with certain capabilities; however, for the test environment, you want to use reduced capabilities to save money. With conditions, you can define which resources are created and how they're configured for each environment type.

Conditions are evaluated based on predefined pseudo parameters or input parameter values that you specify when you create or update a stack. Within each condition, you can reference another condition, a parameter value, or a mapping. After you define all your conditions, you can associate them with resources and resource properties in the `Resources` and `Outputs` sections of a template.

At stack creation or stack update, AWS CloudFormation evaluates all the conditions in your template before creating any resources. Resources that are associated with a true condition are created. Resources that are associated with a false condition are ignored. AWS CloudFormation also re-evaluates these conditions at each stack update before updating any resources. Resources that are still associated with a true condition are updated. Resources that are now associated with a false condition are deleted.

Important

During a stack update, you cannot update conditions by themselves. You can update conditions only when you include changes that add, modify, or delete resources.

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

Please visit https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/parameters-section-structure.html for more information on the parameter structure.

Incorrect options:

Resources - Resources section describes the resources that you want to provision in your AWS CloudFormation stacks. You can associate conditions with the resources that you want to conditionally create.

Conditions - You actually define conditions in this section of the CloudFormation template

Outputs - The optional Outputs section declares output values that you can import into other stacks (to create cross-stack references), return in response (to describe stack calls), or view on the AWS CloudFormation console. For example, you can output the S3 bucket name for a stack to make the bucket easier to find. You can associate conditions with the outputs that you want to conditionally create.

References:

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

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

Question 63: **Incorrect**

A gaming company wants to store information about all the games that the company has released. Each game has a name, version number, and category (such as sports, puzzles, strategy, etc). The game information also can include additional properties about the supported platforms and technical specifications. This additional information is inconsistent across games.

You have been hired as an AWS Certified Developer Associate to build a solution that addresses the following use cases:

For a given name and version number, get all details about the game that has that name and version number.

For a given name, get all details about all games that have that name.

For a given category, get all details about all games in that category.

What will you recommend as the most efficient solution?

- (●) **Set up an Amazon DynamoDB table with a primary key that consists of the category as the partition key and the version number as the sort key. Create a global secondary index that has the name as the partition key** **(Incorrect)**
- ( ) **Permanently store the name, version number, and category information about the games in an Amazon Elasticache for Memcached instance**
- ( ) **Set up an Amazon DynamoDB table with a primary key that consists of the name as the partition key and the version number as the sort key. Create a global secondary index that has the category as the partition key and the name as the sort key** **(Correct)**
- ( ) **Set up an Amazon RDS MySQL instance having a `games` table that contains columns for name, version number, and category. Configure the name column as the primary key**

Explanation
Correct option:

Set up an Amazon DynamoDB table with a primary key that consists of the name as the partition key and the version number as the sort key. Create a global secondary index that has the category as the partition key and the name as the sort key

When you create a DynamoDB table, in addition to the table name, you must specify the primary key of the table. The primary key uniquely identifies each item in the table, so that no two items can have the same key. You can create one or more secondary indexes on a table. A secondary index lets you query the data in the table using an alternate key, in addition to queries against the primary key. DynamoDB doesn't require that you use indexes, but they give your applications more flexibility when querying your data.

Primary key

When you create a table, in addition to the table name, you must specify the primary key of the table. The primary key uniquely identifies each item in the table, so that no two items can have the same key.

DynamoDB supports two different kinds of primary keys:

- **Partition key** – A simple primary key, composed of one attribute known as the *partition key*.

  DynamoDB uses the partition key's value as input to an internal hash function. The output from the hash function determines the partition (physical storage internal to DynamoDB) in which the item will be stored.

  In a table that has only a partition key, no two items can have the same partition key value.

  The *People* table described in Tables, items, and attributes is an example of a table with a simple primary key (*PersonID*). You can access any item in the *People* table directly by providing the *PersonId* value for that item.
- **Partition key and sort key** – Referred to as a *composite primary key*, this type of key is composed of two attributes. The first attribute is the *partition key*, and the second attribute is the *sort key*.

  DynamoDB uses the partition key value as input to an internal hash function. The output from the hash function determines the partition (physical storage internal to DynamoDB) in which the item will be stored. All items with the same partition key value are stored together, in sorted order by sort key value.

  In a table that has a partition key and a sort key, it's possible for multiple items to have the same partition key value. However, those items must have different sort key values.

  The *Music* table described in Tables, items, and attributes is an example of a table with a composite primary key (*Artist* and *SongTitle*). You can access any item in the *Music* table directly, if you provide the *Artist* and *SongTitle* values for that item.

  A composite primary key gives you additional flexibility when querying data. For example, if you provide only the value for *Artist*, DynamoDB retrieves all of the songs by that artist. To retrieve only a subset of songs by a particular artist, you can provide a value for *Artist* along with a range of values for *SongTitle*.

> **Note**
>
> The partition key of an item is also known as its *hash attribute*. The term *hash attribute* derives from the use of an internal hash function in DynamoDB that evenly distributes data items across partitions, based on their partition key values.
>
> The sort key of an item is also known as its *range attribute*. The term *range attribute* derives from the way DynamoDB stores items with the same partition key physically close together, in sorted order by the sort key value.

Each primary key attribute must be a scalar (meaning that it can hold only a single value). The only data types allowed for primary key attributes are string, number, or binary. There are no such restrictions for other, non-key attributes.

Secondary indexes

You can create one or more secondary indexes on a table. A *secondary index* lets you query the data in the table using an alternate key, in addition to queries against the primary key. DynamoDB doesn't require that you use indexes, but they give your applications more flexibility when querying your data. After you create a secondary index on a table, you can read data from the index in much the same way as you do from the table.

DynamoDB supports two kinds of indexes:

- Global secondary index – An index with a partition key and sort key that can be different from those on the table.
- Local secondary index – An index that has the same partition key as the table, but a different sort key.

Each table in DynamoDB has a quota of 20 global secondary indexes (default quota) and 5 local secondary indexes.

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

You should note that a global secondary index (GSI) contains a selection of attributes from the base table, but they are organized by a primary key that is different from that of the table. The Global secondary indexes allow you to perform queries on attributes that are not part of the table's primary key.

Scenario: Using a Global Secondary Index

To illustrate, consider a table named `GameScores` that tracks users and scores for a mobile gaming application. Each item in `GameScores` is identified by a partition key (`UserId`) and a sort key (`GameTitle`). The following diagram shows how the items in the table would be organized. (Not all of the attributes are shown.)

GameScores

UserId	GameTitle	TopScore	TopScoreDateTime	Wins	Losses	...
"101"	"Galaxy Invaders"	5842	"2015-09-15:17:24:31"	21	72	...
"101"	"Meteor Blasters"	1000	"2015-10-22:23:18:01"	12	3	...
"101"	"Starship X"	24	"2015-08-31:13:14:21"	4	9	...
"102"	"Alien Adventure"	192	"2015-07-12:11:07:56"	32	192	...
"102"	"Galaxy Invaders"	0	"2015-09-18:07:33:42"	0	5	...
"103"	"Attack Ships"	3	"2015-10-19:01:13:24"	1	8	...
"103"	"Galaxy Invaders"	2317	"2015-09-11:06:53:00"	40	3	...
"103"	"Meteor Blasters"	723	"2015-10-19:01:13:24"	22	12	...
"103"	"Starship X"	42	"2015-07-11:06:53:00"	4	19	...
...	...	...	...	...	...	...

Now suppose that you wanted to write a leaderboard application to display top scores for each game. A query that specified the key attributes (`UserId` and `GameTitle`) would be very efficient. However, if the application needed to retrieve data from `GameScores` based on `GameTitle` only, it would need to use a `Scan` operation. As more items are added to the table, scans of all the data would become slow and inefficient. This makes it difficult to answer questions such as the following:

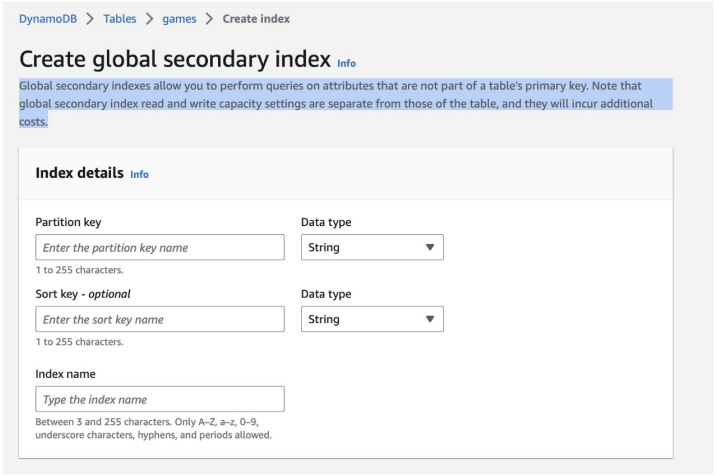
- What is the top score ever recorded for the game Meteor Blasters?
- Which user had the highest score for Galaxy Invaders?
- What was the highest ratio of wins vs. losses?

To speed up queries on non-key attributes, you can create a global secondary index. A global secondary index contains a selection of attributes from the base table, but they are organized by a primary key that is different from that of the table. The index key does not need to have any of the key attributes from the table. It doesn't even need to have the same key schema as a table.

For example, you could create a global secondary index named `GameTitleIndex`, with a partition key of `GameTitle` and a sort key of `TopScore`. The base table's primary key attributes are always projected into an index, so the `UserId` attribute is also present. The following diagram shows what `GameTitleIndex` index would look like.

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

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



Incorrect options:

Set up an Amazon DynamoDB table with a primary key that consists of the category as the partition key and the version number as the sort key. Create a global secondary index that has the name as the partition key - The DynamoDB table for this option has the primary key and GSI that do not solve for the condition - "For a given name and version number, get all details about the game that has that name and version number". This option does not allow for efficient querying of a specific game by its name and version number as you need multiple queries which would be less efficient than the single query allowed by the correct option.

Set up an Amazon RDS MySQL instance having a `games` table that contains columns for name, version number, and category. Configure the name column as the primary key - This option is not the right fit as it does not allow you to efficiently query on the version number and category columns.

Permanently store the name, version number, and category information about the games in an Amazon Elasticache for Memcached instance - You cannot use Elasticache for Memcached to permanently store values meant to be persisted in a database (relational or NoSQL). Elasticache is a caching layer. So this option is incorrect.

References:

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

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

https://aws.amazon.com/premiumsupport/knowledge-center/primary-key-dynamodb-table/

Question 64: **Correct**

A multi-national company has just moved to AWS Cloud and it has configured forecast-based AWS Budgets alerts for cost management. However, no alerts have been received even though the account and the budgets have been created almost three weeks ago.

What could be the issue with the AWS Budgets configuration?

- ○ **Budget forecast has been created from an account that does not have enough privileges**
- ◉ **AWS requires approximately 5 weeks of usage data to generate budget forecasts** **(Correct)**
- ○ **Account has to be part of AWS Organizations to receive AWS Budgets alerts**
- ○ **Amazon CloudWatch could be down and hence alerts are not being sent**

Explanation

Correct option:

AWS Budgets lets customers set custom budgets and receive alerts if their costs or usage exceed (or are forecasted to exceed) their budgeted amount.

AWS requires approximately 5 weeks of usage data to generate budget forecasts - AWS requires approximately 5 weeks of usage data to generate budget forecasts. If you set a budget to alert based on a forecasted amount, this budget alert isn't triggered until you have enough historical usage information.

Incorrect options:

Budget forecast has been created from an account that does not have enough privileges - This is an incorrect statement. If the user account does not have enough privileges, the user will not be able to create the budget at all.

Amazon CloudWatch could be down and hence alerts are not being sent - Amazon CloudWatch is fully managed by AWS, this option has been added as a distractor.

Account has to be part of AWS Organizations to receive AWS Budget alerts - This is an incorrect statement. Stand-alone accounts too can create budgets and being part of an Organization is not mandatory to use AWS Budgets.

Reference:

https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/budgets-best-practices.html

Question 65: **Correct**

Your global organization has an IT infrastructure that is deployed using CloudFormation on AWS Cloud. One employee, in us-east-1 Region, has created a stack 'Application1' and made an exported output with the name 'ELBDNSName'. Another employee has created a stack for a different application 'Application2' in us-east-2 Region and also exported an output with the name 'ELBDNSName'. The first employee wanted to deploy the CloudFormation stack 'Application1' in us-east-2, but it got an error. What is the cause of the error?

- ○ **Exported Output Values in CloudFormation must have unique names across all Regions**
- ◉ **Exported Output Values in CloudFormation must have unique names within a single Region** **(Correct)**
- ○ **Output Values in CloudFormation must have unique names across all Regions**
- ○ **Output Values in CloudFormation must have unique names within a single Region**

Explanation
Correct option:

"Exported Output Values in CloudFormation must have unique names within a single Region"

Using CloudFormation, you can create a template that describes all the AWS resources that you want (like Amazon EC2 instances or Amazon RDS DB instances), and AWS CloudFormation takes care of provisioning and configuring those resources for you.

A CloudFormation template has an optional Outputs section which declares output values that you can import into other stacks (to create cross-stack references), return in response (to describe stack calls), or view on the AWS CloudFormation console. For example, you can output the S3 bucket name for a stack to make the bucket easier to find.

You can use the Export Output Values to export the name of the resource output for a cross-stack reference. For each AWS account, export names must be unique within a region. In this case, we would have a conflict within us-east-2.

Export (optional)

The name of the resource output to be exported for a cross-stack reference.

Note

The following restrictions apply to cross-stack references:

- For each AWS account, `Export` names must be unique within a region.
- You can't create cross-stack references across regions. You can use the intrinsic function `Fn::ImportValue` to import only values that have been exported within the same region.
- For outputs, the value of the `Name` property of an `Export` can't use `Ref` or `GetAtt` functions that depend on a resource.

  Similarly, the `ImportValue` function can't include `Ref` or `GetAtt` functions that depend on a resource.
- You can't delete a stack if another stack references one of its outputs.
- You can't modify or remove an output value that is referenced by another stack.

You can use intrinsic functions to customize the `Name` value of an export. The following examples use the `Fn::Join` function.

JSON

```
"Export" : {
  "Name" : {
    "Fn::Join" : [ ":", [ { "Ref" : "AWS::StackName" }, "AccountVPC" ] ]
  }
}
```

YAML

```
Export:
  Name: !Join [ ":", [ !Ref "AWS::StackName", AccountVPC ] ]
```

To associate a condition with an output, define the condition in the `Conditions` section of the template.

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

Incorrect options:

"Output Values in CloudFormation must have unique names across all Regions"

"Exported Output Values in CloudFormation must have unique names across all Regions"

"Output Values in CloudFormation must have unique names within a single Region"

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

Reference:

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