Question 1: **Correct**

The development team at an e-commerce company completed the last deployment for their application at a reduced capacity because of the deployment policy. The application took a performance hit because of the traffic spike due to an on-going sale.

Which of the following represents the BEST deployment option for the upcoming application version such that it maintains at least the FULL capacity of the application and MINIMAL impact of failed deployment?

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

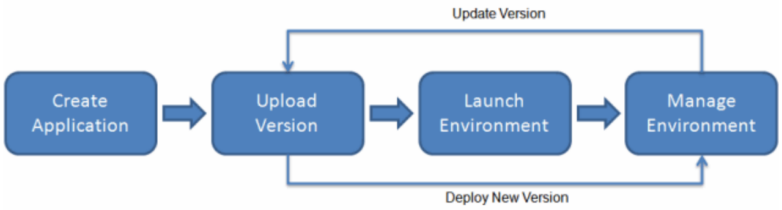
Explanation

Correct option:

Deploy the new application version using 'Immutable' deployment policy

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:

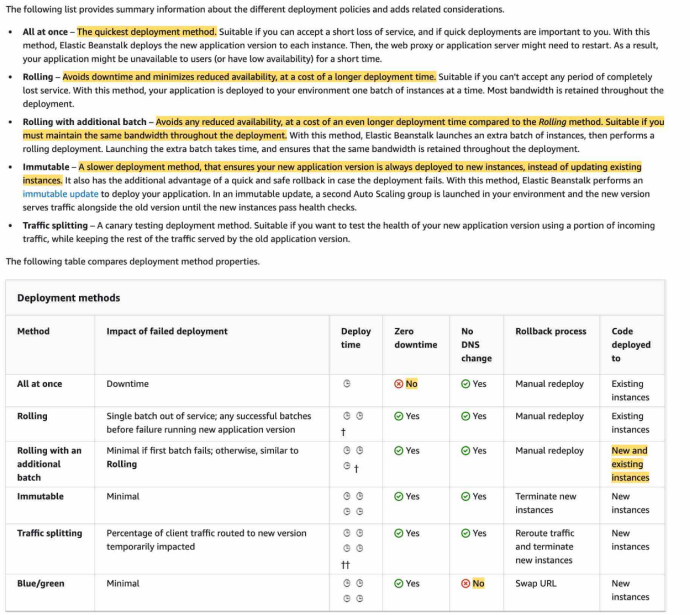


Deploy New Version

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

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. 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. In case of deployment failure, the new instances are terminated, so the impact is minimal.

Overview of Elastic Beanstalk Deployment Policies:



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

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

The following table compares deployment method properties.

Deployment methods

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

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

Incorrect options:

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

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

Deploy the new application version using 'Rolling with additional batch' deployment policy - This policy avoids any reduced availability, at a cost of an even longer deployment time compared to the Rolling method. Suitable if you must maintain the same bandwidth throughout the deployment. However in case of deployment failure, the rollback process is via manual redeploy, so it's not as quick as the Immutable deployment.

References:

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

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

Question 2: **Correct**

Your team lead has asked you to learn AWS CloudFormation to create a collection of related AWS resources and provision them in an orderly fashion. You decide to provide AWS-specific parameter types to catch invalid values.

When specifying parameters which of the following is not a valid Parameter type?

- ○ **String**
- ◉ **DependentParameter** (Correct)
- ○ **AWS::EC2::KeyPair::KeyName**
- ○ **CommaDelimitedList**

Explanation
Correct option:

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

How CloudFormation Works:

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

Parameter types enable CloudFormation to validate inputs earlier in the stack creation process.

CloudFormation currently supports the following parameter types:

```
String – A literal string
Number – An integer or float
List<Number> – An array of integers or floats
CommaDelimitedList – An array of literal strings that are separated by co
mmas
AWS::EC2::KeyPair::KeyName – An Amazon EC2 key pair name
AWS::EC2::SecurityGroup::Id – A security group ID
AWS::EC2::Subnet::Id – A subnet ID
AWS::EC2::VPC::Id – A VPC ID
List<AWS::EC2::VPC::Id> – An array of VPC IDs
List<AWS::EC2::SecurityGroup::Id> – An array of security group IDs
List<AWS::EC2::Subnet::Id> – An array of subnet IDs
```

DependentParameter

In CloudFormation, parameters are all independent and cannot depend on each other. Therefore, this is an invalid parameter type.

Incorrect options:

String

CommaDelimitedList

AWS::EC2::KeyPair::KeyName

As mentioned in the explanation above, these are valid parameter types.

Reference:

https://aws.amazon.com/blogs/devops/using-the-new-cloudformation-parameter-types/

Question 3: **Correct**

A company needs a version control system for their fast development lifecycle with incremental changes, version control, and support to existing Git tools.

Which AWS service will meet these requirements?

- ○ **AWS CodeBuild**
- ○ **Amazon Versioned S3 Bucket**
- ○ **AWS CodePipeline**
- ◉ **AWS CodeCommit** **(Correct)**

Explanation
Correct option:

AWS CodeCommit - AWS CodeCommit is a fully-managed Source Control service that hosts secure Git-based repositories. It makes it easy for teams to collaborate on code in a secure and highly scalable ecosystem. AWS CodeCommit helps you collaborate on code with teammates via pull requests, branching and merging. AWS CodeCommit keeps your repositories close to your build, staging, and production environments in the AWS cloud. You can transfer incremental changes instead of the entire application. AWS CodeCommit supports all Git commands and works with your existing Git tools. You can keep using your preferred development environment plugins, continuous integration/continuous delivery systems, and graphical clients with CodeCommit.

Incorrect options:

Amazon Versioned S3 Bucket - AWS CodeCommit is designed for collaborative software development. It manages batches of changes across multiple files, offers parallel branching, and includes version differencing ("diffing"). In comparison, Amazon S3 versioning supports recovering past versions of individual files but doesn't support tracking batched changes that span multiple files or other features needed for collaborative software development.

AWS CodePipeline - AWS CodePipeline is a fully managed "continuous delivery" service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define.

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

your builds are not left waiting in a queue.

References:

https://aws.amazon.com/codecommit/

https://aws.amazon.com/codepipeline/

https://aws.amazon.com/codebuild/

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

Question 4: **Incorrect**

A media publishing company is using Amazon EC2 instances for running their business-critical applications. Their IT team is looking at reserving capacity apart from savings plans for the critical instances.

As a Developer Associate, which of the following reserved instance types you would select to provide capacity reservations?

- ◯ **Both Regional Reserved Instances and Zonal Reserved Instances**
- ◯ **Neither Regional Reserved Instances nor Zonal Reserved Instances**
- ◉ **Regional Reserved Instances** **(Incorrect)**
- ◯ **Zonal Reserved Instances** **(Correct)**

Explanation
Correct option:

When you purchase a Reserved Instance for a specific Availability Zone, it's referred to as a Zonal Reserved Instance. Zonal Reserved Instances provide capacity reservations as well as discounts.

Zonal Reserved Instances - A zonal Reserved Instance provides a capacity reservation in the specified Availability Zone. Capacity Reservations enable you to reserve capacity for your Amazon EC2 instances in a specific Availability Zone for any duration. This gives you the ability to create and manage Capacity Reservations independently from the billing discounts offered by Savings Plans or regional Reserved Instances.

Regional and Zonal Reserved Instances:

Regional and zonal Reserved Instances (scope)

PDF | Kindle | RSS

When you purchase a Reserved Instance, you determine the scope of the Reserved Instance. The scope is either regional or zonal.

- **Regional**: When you purchase a Reserved Instance for a Region, it's referred to as a *regional* Reserved Instance.
- **Zonal**: When you purchase a Reserved Instance for a specific Availability Zone, it's referred to as a *zonal* Reserved Instance.

Differences between regional and zonal Reserved Instances

The following table highlights some key differences between regional Reserved Instances and zonal Reserved Instances:

	Regional Reserved Instances	Zonal Reserved Instances
Availability Zone flexibility	The Reserved Instance discount applies to instance usage in any Availability Zone in the specified Region.	No Availability Zone flexibility—the Reserved Instance discount applies to instance usage in the specified Availability Zone only.
Capacity reservation	No capacity reservation—a regional Reserved Instance does *not* provide a capacity reservation.	A zonal Reserved Instance provides a capacity reservation in the specified Availability Zone.
Instance size flexibility	The Reserved Instance discount applies to instance usage within the instance family, regardless of size. Only supported on Amazon Linux/Unix Reserved Instances with default tenancy. For more information, see Instance size flexibility determined by normalization factor.	No instance size flexibility—the Reserved Instance discount applies to instance usage for the specified instance type and size only.

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

High Level Overview of EC2 Instance Purchase Options:

On-Demand

With On-Demand instances, you pay for compute capacity by the hour or the second depending on which instances you run. No longer-term commitments or upfront payments are needed. You can increase or decrease your compute capacity depending on the demands of your application and only pay the specified per hourly rates for the instance you use.

On-Demand instances are recommended for:

- Users that prefer the low cost and flexibility of Amazon EC2 without any up-front payment or long-term commitment
- Applications with short-term, spiky, or unpredictable workloads that cannot be interrupted
- Applications being developed or tested on Amazon EC2 for the first time

See On-Demand pricing »

Spot instances

Amazon EC2 Spot instances allow you to request spare Amazon EC2 computing capacity for up to 90% off the On-Demand price. Learn More.

Spot instances are recommended for:

- Applications that have flexible start and end times
- Applications that are only feasible at very low compute prices
- Users with urgent computing needs for large amounts of additional capacity

See Spot pricing »

Savings Plans

Savings Plans are a flexible pricing model that offer low prices on EC2 and Fargate usage, in exchange for a commitment to a consistent amount of usage (measured in $/hour) for a 1 or 3 year term.

Dedicated Hosts

A Dedicated Host is a physical EC2 server dedicated for your use. Dedicated Hosts can help you reduce costs by allowing you to use your existing server-bound software licenses, including Windows Server, SQL Server, and SUSE Linux Enterprise Server (subject to your license terms), and can also help you meet compliance requirements. Learn more.

- Can be purchased On-Demand (hourly).
- Can be purchased as a Reservation for up to 70% off the On-Demand price.

See Dedicated pricing »

Reserved Instances

Reserved Instances provide you with a significant discount (up to 75%) compared to On-Demand instance pricing. In addition, when Reserved Instances are assigned to a specific Availability Zone, they provide a capacity reservation, giving you additional confidence in your ability to launch instances when you need them.

For applications that have steady state or predictable usage, Reserved Instances can provide significant savings compared to using On-Demand instances. See How to Purchase Reserved Instances for more information.

Reserved Instances are recommended for:

- Applications with steady state usage
- Applications that may require reserved capacity
- Customers that can commit to using EC2 over a 1 or 3 year term to reduce their total computing costs

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

Incorrect options:

Regional Reserved Instances - When you purchase a Reserved Instance for a Region, it's referred to as a regional Reserved Instance. A regional Reserved Instance does not provide a capacity reservation.

Both Regional Reserved Instances and Zonal Reserved Instances - As discussed above, only Zonal Reserved Instances provide capacity reservation.

Neither Regional Reserved Instances nor Zonal Reserved Instances - As discussed above, Zonal Reserved Instances provide capacity reservation.

References:

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

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

Question 5: **Incorrect**

The development team at a retail company is gearing up for the upcoming Thanksgiving sale and wants to make sure that the application's serverless backend running via Lambda functions does not hit latency bottlenecks as a result of the traffic spike.

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

- ○ **Add an Application Load Balancer in front of the Lambda functions**
- ○ **Configure Application Auto Scaling to manage Lambda provisioned concurrency on a schedule** **(Correct)**
- ○ **Configure Application Auto Scaling to manage Lambda reserved concurrency on a schedule**
- ◉ **No need to make any special provisions as Lambda is automatically scalable because of its serverless nature** **(Incorrect)**

Explanation
Correct option:

Configure Application Auto Scaling to manage Lambda provisioned concurrency on a schedule

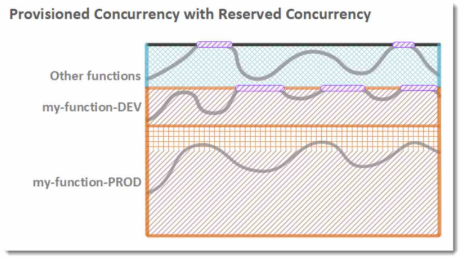
Concurrency is the number of requests that a Lambda function is serving at any given time. If a Lambda function is invoked again while a request is still being processed, another instance is allocated, which increases the function's concurrency.

Due to a spike in traffic, when Lambda functions scale, this causes the portion of requests that are served by new instances to have higher latency than the rest. To enable your function to scale without fluctuations in latency, use provisioned concurrency. By allocating provisioned concurrency before an increase in invocations, you can ensure that all requests are served by initialized instances with very low latency.

You can configure Application Auto Scaling to manage provisioned concurrency on a schedule or based on utilization. Use scheduled scaling to increase provisioned concurrency in anticipation of peak traffic. To increase provisioned concurrency automatically as needed, use the Application Auto Scaling API to register a target and create a scaling policy.

Please see this note for more details on provisioned concurrency:

In the following example, the `my-function-DEV` and `my-function-PROD` functions are configured with both reserved and provisioned concurrency. For `my-function-DEV`, the full pool of reserved concurrency is also provisioned concurrency. In this case, all invocations either run on provisioned concurrency or are throttled. For `my-function-PROD`, a portion of the reserved concurrency pool is standard concurrency. When all provisioned concurrency is in use, the function scales on standard concurrency to serve any additional requests.



Legend

- Function concurrency
- Reserved concurrency
- Provisioned concurrency
- Unreserved concurrency
- Throttling

Provisioned concurrency does not come online immediately after you configure it. Lambda starts allocating provisioned concurrency after a minute or two of preparation. Similar to how functions scale under load, up to 3000 instances of the function can be initialized at once, depending on the Region. After the initial burst, instances are allocated at a steady rate of 500 per minute until the request is fulfilled. When you request provisioned concurrency for multiple functions or versions of a function in the same Region, scaling limits apply across all requests.

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

Incorrect options:

Configure Application Auto Scaling to manage Lambda reserved concurrency on a schedule - To ensure that a function can always reach a certain level of concurrency, you can configure the function with reserved concurrency. When a function has reserved concurrency, no other function can use that concurrency. More importantly, reserved concurrency also limits the maximum concurrency for the function, and applies to the function as a whole, including versions and aliases.

You cannot configure Application Auto Scaling to manage Lambda reserved concurrency on a schedule.

Add an Application Load Balancer in front of the Lambda functions - This is a distractor as just adding the Application Load Balancer will not help in scaling the Lambda functions to address the surge in traffic.

No need to make any special provisions as Lambda is automatically scalable because of its serverless nature - It's true that Lambda is serverless, however, due to the surge in traffic the Lambda functions can still hit the concurrency limits. So this option is incorrect.

Reference:

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

Question 6: **Correct**

As an AWS Certified Developer Associate, you have been hired to work with the development team at a company to create a REST API using the serverless architecture.

Which of the following solutions will you choose to move the company to the serverless architecture paradigm?

- (●) **API Gateway exposing Lambda Functionality** **(Correct)**
- ( ) **Fargate with Lambda at the front**
- ( ) **Public-facing Application Load Balancer with ECS on Amazon EC2**
- ( ) **Route 53 with EC2 as backend**

Explanation
Correct option:

API Gateway exposing Lambda Functionality

Amazon API Gateway is a fully managed service that makes it easy for developers to create, publish, maintain, monitor, and secure APIs at any scale. APIs act as the "front door" for applications to access data, business logic, or functionality from your backend services.

How API Gateway Works:

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

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

How Lambda function works:

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

API Gateway can expose Lambda functionality through RESTful APIs. Both are serverless options offered by AWS and hence the right choice for this scenario, considering all the functionality they offer.

Incorrect options:

Fargate with Lambda at the front - Lambda cannot directly handle RESTful API requests. You can invoke a Lambda function over HTTPS by defining a custom RESTful API using Amazon API Gateway. So, Fargate with Lambda as the front-facing service is a wrong combination, though both Fargate and Lambda are serverless.

Public-facing Application Load Balancer with ECS on Amazon EC2 - ECS on Amazon EC2 does not come under serverless and hence cannot be considered for this use case.

Route 53 with EC2 as backend - Amazon EC2 is not a serverless service and hence cannot be considered for this use case.

References:

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

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

Question 7: **Correct**

The development team at a HealthCare company has deployed EC2 instances in AWS Account A. These instances need to access patient data with Personally Identifiable Information (PII) on multiple S3 buckets in another AWS Account B.

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

- ○ **Copy the underlying AMI for the EC2 instances from Account A into Account B. Launch EC2 instances in Account B using this AMI and then access the PII data on Amazon S3 in Account B**
- ○ **Add a bucket policy to all the Amazon S3 buckets in Account B to allow access from EC2 instances in Account A**
- ○ **Create an IAM role (instance profile) in Account A and set Account B as a trusted entity. Attach this role to the EC2 instances in Account A and add an inline policy to this role to access S3 data from Account B**
- ◉ **Create an IAM role with S3 access in Account B and set Account A as a trusted entity. Create another role (instance profile) in Account A and attach it to the EC2 instances in Account A and add an inline policy to this role to assume the role from Account B** (Correct)

Explanation

Correct option:

Create an IAM role with S3 access in Account B and set Account A as a trusted entity. Create another role (instance profile) in Account A and attach it to the EC2 instances in Account A and add an inline policy to this role to assume the role from Account B

You can give EC2 instances in one account ("account A") permissions to assume a role from another account ("account B") to access resources such as S3 buckets. You need to create an IAM role in Account B and set Account A as a trusted entity. Then attach a policy to this IAM role such that it delegates access to Amazon S3 like so -

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::awsexamplebucket1",
                "arn:aws:s3:::awsexamplebucket1/*",
                "arn:aws:s3:::awsexamplebucket2",
                "arn:aws:s3:::awsexamplebucket2/*"
            ]
        }
    ]
}
```

Then you can create another role (instance profile) in Account A and attach it to the EC2 instances in Account A and add an inline policy to this role to assume the role from Account B like so -

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::AccountB_ID:role/ROLENAME"
        }
    ]
}
```

Incorrect options:

Create an IAM role (instance profile) in Account A and set Account B as a trusted entity. Attach this role to the EC2 instances in Account A and add an inline policy to this role to access S3 data from Account B - This option contradicts the explanation provided earlier in the explanation, hence this option is incorrect.

Copy the underlying AMI for the EC2 instances from Account A into Account B. Launch EC2 instances in Account B using this AMI and then access the PII data on Amazon S3 in Account B - Copying the AMI is a distractor as this does not solve the use-case outlined in the problem statement.

Add a bucket policy to all the Amazon S3 buckets in Account B to allow access from EC2 instances in Account A - Just adding a bucket policy in Account B is not enough, as you also need to create an IAM policy in Account A to access S3 objects in Account B.

Please review this reference material for a deep-dive on cross-account access to objects that are in Amazon S3 buckets - https://aws.amazon.com/premiumsupport/knowledge-center/cross-account-access-s3/

References:

https://aws.amazon.com/premiumsupport/knowledge-center/s3-instance-access-bucket/

Question 8: **Incorrect**

A company runs its flagship application on a fleet of Amazon EC2 instances. After misplacing a couple of private keys from the SSH key pairs, they have decided to re-use their SSH key pairs for the different instances across AWS Regions.

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

- ○ **Encrypt the private SSH key and store it in the S3 bucket to be accessed from any AWS Region**
- ○ **Generate a public SSH key from a private SSH key. Then, import the key into each of your AWS Regions** (Correct)
- ○ **Store the public and private SSH key pair in AWS Trusted Advisor and access it across AWS Regions**
- ◉ **It is not possible to reuse SSH key pairs across AWS Regions** (Incorrect)

Explanation
Correct option:

Generate a public SSH key from a private SSH key. Then, import the key into each of your AWS Regions

Here is the correct way of reusing SSH keys in your AWS Regions:

1. Generate a public SSH key (.pub) file from the private SSH key (.pem) file.
2. Set the AWS Region you wish to import to.
3. Import the public SSH key into the new Region.

Incorrect options:

It is not possible to reuse SSH key pairs across AWS Regions - As explained above, it is possible to reuse with manual import.

Store the public and private SSH key pair in AWS Trusted Advisor and access it across AWS Regions - AWS Trusted Advisor is an application that draws upon best practices learned from AWS' aggregated operational history of serving hundreds of thousands of AWS customers. Trusted Advisor inspects your AWS environment and makes recommendations for saving money, improving system performance, or closing security gaps. It does not store key pair credentials.

Encrypt the private SSH key and store it in the S3 bucket to be accessed from any AWS Region - Storing private key to Amazon S3 is possible. But, this will not make the key accessible for all AWS Regions, as is the need in the current use case.

Reference:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html

Question 9: **Correct**

A developer is looking at establishing access control for an API that connects to a Lambda function downstream.

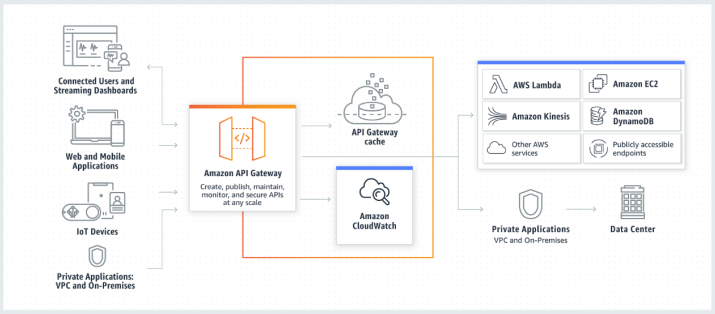
Which of the following represents a mechanism that CANNOT be used for authenticating with the API Gateway?

- (●) **AWS Security Token Service (STS)** **(Correct)**
- ( ) **Cognito User Pools**
- ( ) **Lambda Authorizer**
- ( ) **Standard AWS IAM roles and policies**

Explanation
Correct option:

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

How API Gateway Works:



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

AWS Security Token Service (STS) - AWS Security Token Service (AWS STS) is a web service that enables you to request temporary, limited-privilege credentials for AWS Identity and Access Management (IAM) users or for users that you authenticate (federated users). However, it is not supported by API Gateway.

API Gateway supports the following mechanisms for authentication and authorization:

Controlling and managing access to a REST API in API Gateway

PDF | Kindle | RSS

API Gateway supports multiple mechanisms for controlling and managing access to your API.

You can use the following mechanisms for authentication and authorization:

- **Resource policies** let you create resource-based policies to allow or deny access to your APIs and methods from specified source IP addresses or VPC endpoints. For more information, see Controlling access to an API with API Gateway resource policies.
- **Standard AWS IAM roles and policies** offer flexible and robust access controls that can be applied to an entire API or individual methods. IAM roles and policies can be used for controlling who can create and manage your APIs, as well as who can invoke them. For more information, see Control access to an API with IAM permissions.
- **IAM tags** can be used together with IAM policies to control access. For more information, see Using tags to control access to API Gateway resources.
- **Endpoint policies for interface VPC endpoints** allow you to attach IAM resource policies to interface VPC endpoints to improve the security of your private APIs. For more information, see Use VPC endpoint policies for private APIs in API Gateway.
- **Lambda authorizers** are Lambda functions that control access to REST API methods using bearer token authentication—as well as information described by headers, paths, query strings, stage variables, or context variables request parameters. Lambda authorizers are used to control who can invoke REST API methods. For more information, see Use API Gateway Lambda authorizers.
- **Amazon Cognito user pools** let you create customizable authentication and authorization solutions for your REST APIs. Amazon Cognito user pools are used to control who can invoke REST API methods. For more information, see Control access to a REST API using Amazon Cognito User Pools as authorizer.

via - https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-control-access-to-api.html

Incorrect options:

Standard AWS IAM roles and policies - Standard AWS IAM roles and policies offer flexible and robust access controls that can be applied to an entire API or individual methods. IAM roles and policies can be used for controlling who can create and manage your APIs, as well as who can invoke them.

Lambda Authorizer - Lambda authorizers are Lambda functions that control access to REST API methods using bearer token authentication—as well as information described by headers, paths, query strings, stage variables, or context variables request parameters. Lambda authorizers are used to control who can invoke REST API methods.

Cognito User Pools - Amazon Cognito user pools let you create customizable authentication and authorization solutions for your REST APIs. Amazon Cognito user pools are used to control who can invoke REST API methods.

References:

https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-control-access-to-api.html

https://docs.aws.amazon.com/STS/latest/APIReference/welcome.html

Question 10: **Correct**

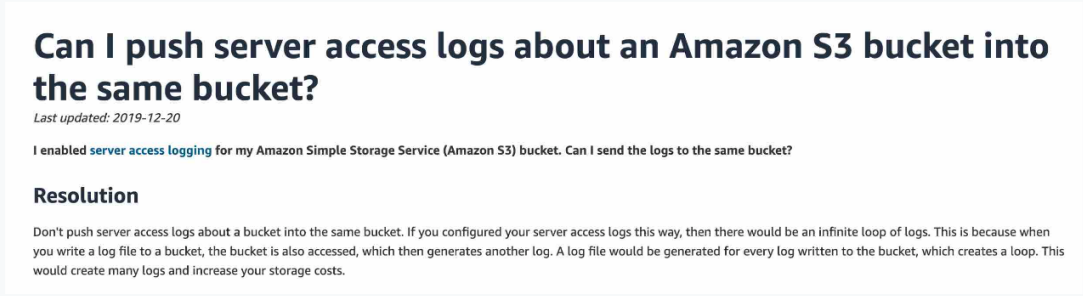
A company has created an Amazon S3 bucket that holds customer data. The team lead has just enabled access logging to this bucket. The bucket size has grown substantially after starting access logging. Since no new files have been added to the bucket, the perplexed team lead is looking for an answer.

Which of the following reasons explains this behavior?

- ○ **Erroneous Bucket policies for batch uploads can sometimes be responsible for the exponential growth of S3 Bucket size**
- ○ **Object Encryption has been enabled and each object is stored twice as part of this configuration**
- ○ **A DDoS attack on your S3 bucket can potentially blow up the size of data in the bucket if the bucket security is compromised during the attack**
- ◉ **S3 access logging is pointing to the same bucket and is responsible for the substantial growth of bucket size** **(Correct)**

Explanation
Correct option:

S3 access logging is pointing to the same bucket and is responsible for the substantial growth of bucket size - When your source bucket and target bucket are the same bucket, additional logs are created for the logs that are written to the bucket. The extra logs about logs might make it harder to find the log that you are looking for. This configuration would drastically increase the size of the S3 bucket.



Can I push server access logs about an Amazon S3 bucket into the same bucket?

Last updated: 2019-12-20

I enabled server access logging for my Amazon Simple Storage Service (Amazon S3) bucket. Can I send the logs to the same bucket?

Resolution

Don't push server access logs about a bucket into the same bucket. If you configured your server access logs this way, then there would be an infinite loop of logs. This is because when you write a log file to a bucket, the bucket is also accessed, which then generates another log. A log file would be generated for every log written to the bucket, which creates a loop. This would create many logs and increase your storage costs.

via - https://aws.amazon.com/premiumsupport/knowledge-center/s3-server-access-logs-same-bucket/

Incorrect options:

Erroneous Bucket policies for batch uploads can sometimes be responsible for the exponential growth of S3 Bucket size - This is an incorrect statement. A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy. You add a bucket policy to a bucket to grant other AWS accounts or IAM users access permissions for the bucket and the objects in it. A bucket policy, for batch processes or normal processes, will not increase the size of the bucket or the objects in it.

A DDOS attack on your S3 bucket can potentially blow up the size of data in the bucket if the bucket security is compromised during the attack - This is an incorrect statement. AWS handles DDoS attacks on all of its managed services. However, a DDoS attack will not increase the size of the bucket.

Object Encryption has been enabled and each object is stored twice as part of this configuration - Encryption does not increase a bucket's size, that too, on daily basis, as if the case in the current scenario

References:

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

https://docs.aws.amazon.com/AmazonS3/latest/user-guide/set-permissions.html

Question 11: **Correct**

A developer working with EC2 Windows instance has installed Kinesis Agent for Windows to stream JSON-formatted log files to Amazon Simple Storage Service (S3) via Amazon Kinesis Data Firehose. The developer wants to understand the sink type capabilities of Kinesis Firehose.

Which of the following sink types is NOT supported by Kinesis Firehose.

- (●) **Amazon ElastiCache with Amazon S3 as backup** (Correct)
- ( ) **Amazon Redshift with Amazon S3**
- ( ) **Amazon Elasticsearch Service (Amazon ES) with optionally backing up data to Amazon S3**
- ( ) **Amazon Simple Storage Service (Amazon S3) as a direct Firehose destination**

Explanation
Correct option:

Amazon Kinesis Data Firehose is a fully managed service for delivering real-time streaming data to destinations such as Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon Elasticsearch Service (Amazon ES), and Splunk. With Kinesis Data Firehose, you don't need to write applications or manage resources. You configure your data producers to send data to Kinesis Data Firehose, and it automatically delivers the data to the destination that you specified.

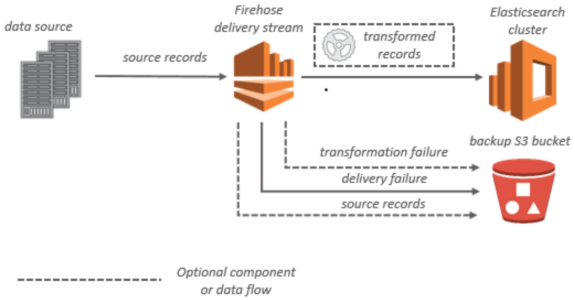
Amazon ElastiCache with Amazon S3 as backup - Amazon ElastiCache is a fully managed in-memory data store, compatible with Redis or Memcached. ElastiCache is NOT a supported destination for Amazon Kinesis Data Firehose.

Incorrect options:

Amazon Elasticsearch Service (Amazon ES) with optionally backing up data to Amazon S3 - Amazon ES is a supported destination type for Kinesis Firehose. Streaming data is delivered to your Amazon ES cluster, and can optionally be backed up to your S3 bucket concurrently.

Data Flow for ES:

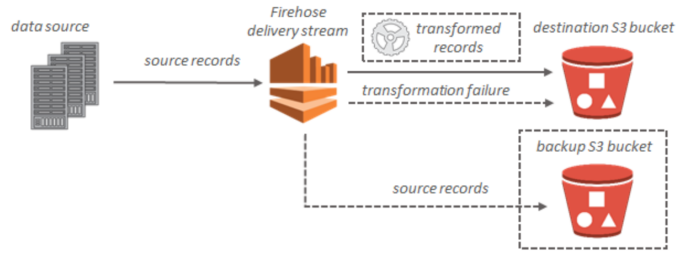
Data Flow for ES:



via - https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html

Amazon Simple Storage Service (Amazon S3) as a direct Firehose destination - For Amazon S3 destinations, streaming data is delivered to your S3 bucket. If data transformation is enabled, you can optionally back up source data to another Amazon S3 bucket.

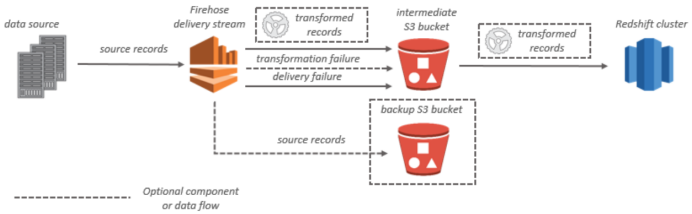
Data Flow for S3:



via - https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html

Amazon Redshift with Amazon S3 - For Amazon Redshift destinations, streaming data is delivered to your S3 bucket first. Kinesis Data Firehose then issues an Amazon Redshift COPY command to load data from your S3 bucket to your Amazon Redshift cluster. If data transformation is enabled, you can optionally back up source data to another Amazon S3 bucket.

Data Flow for Redshift:



via - https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html

Reference:

https://docs.aws.amazon.com/firehose/latest/dev/what-is-this-service.html

Question 12: **Correct**

A company uses Amazon Simple Email Service (SES) to cost-effectively send susbscription emails to the customers. Intermittently, the SES service throws the error: `Throttling – Maximum sending rate exceeded`.

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

- ◯ **Configure Timeout mechanism for each request made to the SES service**
- ◯ **Raise a service request with Amazon to increase the throttling limit for the SES API**
- ◯ **Implement retry mechanism for all 4xx errors to avoid throttling error**
- ◉ **Use Exponential Backoff technique to introduce delay in time before attempting to execute the operation again** **(Correct)**

Explanation

Correct option:

Use Exponential Backoff technique to introduce delay in time before attempting to execute the operation again - A “Throttling – Maximum sending rate exceeded” error is retriable. This error is different than other errors returned by Amazon SES. A request rejected with a “Throttling” error can be retried at a later time and is likely to succeed.

Retries are “selfish.” In other words, when a client retries, it spends more of the server's time to get a higher chance of success. Where failures are rare or transient, that's not a problem. This is because the overall number of retried requests is small, and the tradeoff of increasing apparent availability works well. When failures are caused by overload, retries that increase load can make matters significantly worse. They can even delay recovery by keeping the load high long after the original issue is resolved.

The preferred solution is to use a backoff. Instead of retrying immediately and aggressively, the client waits some amount of time between tries. The most common pattern is an exponential backoff, where the wait time is increased exponentially after every attempt.

A variety of factors can affect your send rate, e.g. message size, network performance or Amazon SES availability. The advantage of the exponential backoff approach is that your application will self-tune and it will call Amazon SES at close to the maximum allowed rate.

Incorrect options:

Configure Timeout mechanism for each request made to the SES service - Requests are configured to timeout if they do not complete successfully in a given time. This helps free up the database, application and any other resource that could potentially keep on waiting to eventually succeed. But, if errors are caused by load, retries can be ineffective if all clients retry at the same time. Throttling error signifies that load is high on SES and it does not make sense to keep retrying.

Raise a service request with Amazon to increase the throttling limit for the SES API - If throttling error is persistent, then it indicates a high load on the system consistently and increasing the throttling limit will be the right solution for the problem. But, the error is only intermittent here, signifying that decreasing the rate of requests will handle the error.

Implement retry mechanism for all 4xx errors to avoid throttling error - 4xx status codes indicate that there was a problem with the client request. Common client request errors include providing invalid credentials and omitting required parameters. When you get a 4xx error, you need to correct the problem and resubmit a properly formed client request. Throttling is a server error and not a client error, hence retry on 4xx errors does not make sense here.

References:

https://aws.amazon.com/builders-library/timeouts-retries-and-backoff-with-jitter/

https://aws.amazon.com/blogs/messaging-and-targeting/how-to-handle-a-throttling-maximum-sending-rate-exceeded-error/

Question 13: **Correct**

A business has purchased one m4.xlarge Reserved Instance but it has used three m4.xlarge instances concurrently for an hour.

As a Developer, explain how the instances are charged?

- ○ **All instances are charged at one hour of Reserved Instance usage**
- ◉ **One instance is charged at one hour of Reserved Instance usage and the other two instances are charged at two hours of On-Demand usage** **(Correct)**
- ○ **All instances are charged at one hour of On-Demand Instance usage**
- ○ **One instance is charged at one hour of On-Demand usage and the other two instances are charged at two hours of Reserved Instance usage**

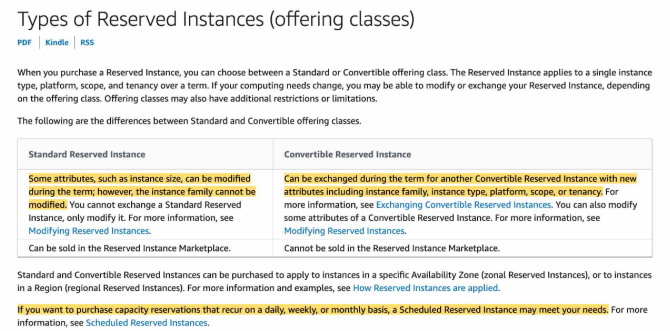
Explanation
Correct option:

All Reserved Instances provide you with a discount compared to On-Demand pricing.

One instance is charged at one hour of Reserved Instance usage and the other two instances are charged at two hours of On-Demand usage

A Reserved Instance billing benefit can apply to a maximum of 3600 seconds (one hour) of instance usage per clock-hour. You can run multiple instances concurrently, but can only receive the benefit of the Reserved Instance discount for a total of 3600 seconds per clock-hour; instance usage that exceeds 3600 seconds in a clock-hour is billed at the On-Demand rate.

Please review this note on the EC2 Reserved Instance types:



Types of Reserved Instances (offering classes)

PDF | Kindle | RSS

When you purchase a Reserved Instance, you can choose between a Standard or Convertible offering class. The Reserved Instance applies to a single instance type, platform, scope, and tenancy over a term. If your computing needs change, you may be able to modify or exchange your Reserved Instance, depending on the offering class. Offering classes may also have additional restrictions or limitations.

The following are the differences between Standard and Convertible offering classes.

Standard Reserved Instance	Convertible Reserved Instance
Some attributes, such as instance size, can be modified during the term; however, the instance family cannot be modified. You cannot exchange a Standard Reserved Instance, only modify it. For more information, see Modifying Reserved Instances.	Can be exchanged during the term for another Convertible Reserved Instance with new attributes including instance family, instance type, platform, scope, or tenancy. For more information, see Exchanging Convertible Reserved Instances. You can also modify some attributes of a Convertible Reserved Instance. For more information, see Modifying Reserved Instances.
Can be sold in the Reserved Instance Marketplace.	Cannot be sold in the Reserved Instance Marketplace.

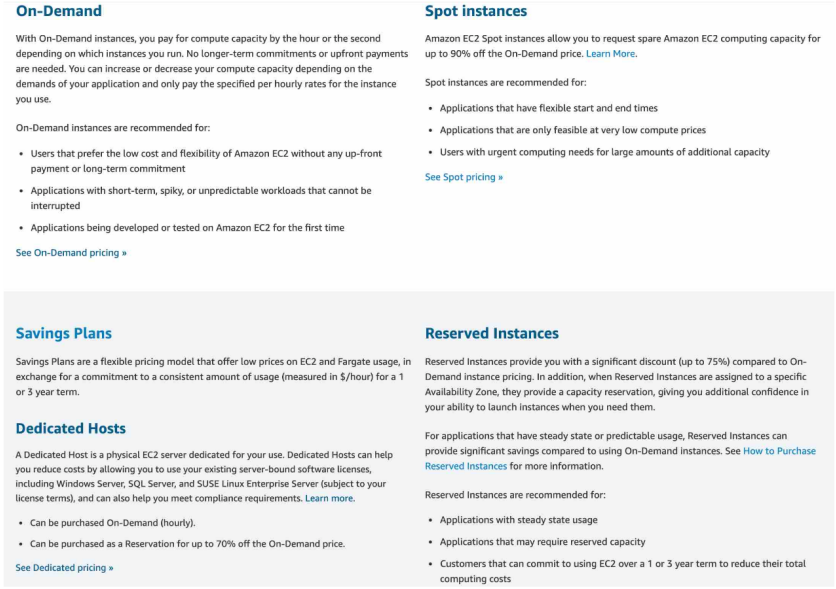
Standard and Convertible Reserved Instances can be purchased to apply to instances in a specific Availability Zone (zonal Reserved Instances), or to instances in a Region (regional Reserved Instances). For more information and examples, see How Reserved Instances are applied.

If you want to purchase capacity reservations that recur on a daily, weekly, or monthly basis, a Scheduled Reserved Instance may meet your needs. For more information, see Scheduled Reserved Instances.

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

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

High Level Overview of EC2 Instance Purchase Options:



On-Demand

With On-Demand instances, you pay for compute capacity by the hour or the second depending on which instances you run. No longer-term commitments or upfront payments are needed. You can increase or decrease your compute capacity depending on the demands of your application and only pay the specified per hourly rates for the instance you use.

On-Demand instances are recommended for:

- Users that prefer the low cost and flexibility of Amazon EC2 without any up-front payment or long-term commitment
- Applications with short-term, spiky, or unpredictable workloads that cannot be interrupted
- Applications being developed or tested on Amazon EC2 for the first time

See On-Demand pricing »

Spot instances

Amazon EC2 Spot instances allow you to request spare Amazon EC2 computing capacity for up to 90% off the On-Demand price. Learn More.

Spot instances are recommended for:

- Applications that have flexible start and end times
- Applications that are only feasible at very low compute prices
- Users with urgent computing needs for large amounts of additional capacity

See Spot pricing »

Savings Plans

Savings Plans are a flexible pricing model that offer low prices on EC2 and Fargate usage, in exchange for a commitment to a consistent amount of usage (measured in $/hour) for a 1 or 3 year term.

Dedicated Hosts

A Dedicated Host is a physical EC2 server dedicated for your use. Dedicated Hosts can help you reduce costs by allowing you to use your existing server-bound software licenses, including Windows Server, SQL Server, and SUSE Linux Enterprise Server (subject to your license terms), and can also help you meet compliance requirements. Learn more.

- Can be purchased On-Demand (hourly).
- Can be purchased as a Reservation for up to 70% off the On-Demand price.

See Dedicated pricing »

Reserved Instances

Reserved Instances provide you with a significant discount (up to 75%) compared to On-Demand instance pricing. In addition, when Reserved Instances are assigned to a specific Availability Zone, they provide a capacity reservation, giving you additional confidence in your ability to launch instances when you need them.

For applications that have steady state or predictable usage, Reserved Instances can provide significant savings compared to using On-Demand instances. See How to Purchase Reserved Instances for more information.

Reserved Instances are recommended for:

- Applications with steady state usage
- Applications that may require reserved capacity
- Customers that can commit to using EC2 over a 1 or 3 year term to reduce their total computing costs

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

Incorrect options:

All instances are charged at one hour of Reserved Instance usage - This is incorrect.

All instances are charged at one hour of On-Demand Instance usage - This is incorrect.

One instance is charged at one hour of On-Demand usage and the other two instances are charged at two hours of Reserved Instance usage - This is incorrect. If multiple eligible instances are running concurrently, the Reserved Instance billing benefit is applied to all the instances at the same time up to a maximum of 3600 seconds in a clock-hour; thereafter, On-Demand rates apply.

Reference:

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

Question 14: **Correct**

You are a development team lead setting permissions for other IAM users with limited permissions. On the AWS Management Console, you created a dev group where new developers will be added, and on your workstation, you configured a developer profile. You would like to test that this user cannot terminate instances.

Which of the following options would you execute?

- ◯ **Retrieve the policy using the EC2 metadata service and use the IAM policy simulator**
- ◯ **Using the CLI, create a dummy EC2 and delete it using another CLI call**
- ◉ **Use the AWS CLI --dry-run option** **(Correct)**
- ◯ **Use the AWS CLI --test option**

Explanation
Correct option:

Use the AWS CLI --dry-run option: The --dry-run option checks whether you have the required permissions for the action, without actually making the request, and provides an error response. If you have the required permissions, the error response is DryRunOperation, otherwise, it is UnauthorizedOperation.

Incorrect options:

Use the AWS CLI --test option - This is a made-up option and has been added as a distractor.

Retrieve the policy using the EC2 metadata service and use the IAM policy simulator - EC2 metadata service is used to retrieve dynamic information such as instance-id, local-hostname, public-hostname. This cannot be used to check whether you have the required permissions for the action.

Using the CLI, create a dummy EC2 and delete it using another CLI call - That would not work as the current EC2 may have permissions that the dummy instance does not have. If permissions were the same it can work but it's not as elegant as using the dry-run option.

References:

https://docs.aws.amazon.com/AWSEC2/latest/APIReference/API_RunInstances.html

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

Question 15: **Correct**

After a code review, a developer has been asked to make his publicly accessible S3 buckets private, and enable access to objects with a time-bound constraint.

Which of the following options will address the given use-case?

- ○ **Use Routing policies to re-route unintended access**
- ○ **Use Bucket policy to block the unintended access**
- ◉ **Share pre-signed URLs with resources that need access** **(Correct)**
- ○ **It is not possible to implement time constraints on Amazon S3 Bucket access**

Explanation

Correct option:

Share pre-signed URLs with resources that need access - All objects by default are private, with the object owner having permission to access the objects. However, the object owner can optionally share objects with others by creating a pre-signed URL, using their own security credentials, to grant time-limited permission to download the objects. When you create a pre-signed URL for your object, you must provide your security credentials, specify a bucket name, an object key, specify the HTTP method (GET to download the object), and expiration date and time. The pre-signed URLs are valid only for the specified duration.

Incorrect options:

Use Bucket policy to block the unintended access - A bucket policy is a resource-based AWS Identity and Access Management (IAM) policy. You add a bucket policy to a bucket to grant other AWS accounts or IAM users access permissions for the bucket and the objects in it. Bucket policy can be used to block off unintended access, but it's not possible to provide time-based access, as is the case in the current use case.

Use Routing policies to re-route unintended access - There is no such facility directly available with Amazon S3.

It is not possible to implement time constraints on Amazon S3 Bucket access - This is an incorrect statement. As explained above, it is possible to give time-bound access permissions on S3 buckets and objects.

References:

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

https://docs.aws.amazon.com/AmazonS3/latest/user-guide/add-bucket-policy.html

Question 16: **Correct**

Consider an application that enables users to store their mobile phone images in the cloud and supports tens of thousands of users. The application should utilize an Amazon API Gateway REST API that leverages AWS Lambda functions for photo processing while storing photo details in Amazon DynamoDB. The application should allow users to create an account, upload images, and retrieve previously uploaded images, with images ranging in size from 500 KB to 5 MB.

How will you design the application with the least operational overhead?

- (●) **Leverage Cognito user pools to manage user accounts and set up an Amazon Cognito user pool authorizer in API Gateway to control access to the API. Set up a Lambda function to store the images in Amazon S3 and save the image object's S3 key as part of the photo details in a DynamoDB table. Have the Lambda function retrieve previously uploaded images by querying DynamoDB for the S3 key** (Correct)

- ( ) **Use Cognito identity pools to manage user accounts and set up an Amazon Cognito identity pool authorizer in API Gateway to control access to the API. Set up a Lambda function to store the images in Amazon S3 and save the image object's S3 key as part of the photo details in a DynamoDB table. Have the Lambda function retrieve previously uploaded images by querying DynamoDB for the S3 key**

- ( ) **Leverage Cognito user pools to manage user accounts and set up an Amazon Cognito user pool authorizer in API Gateway to control access to the API. Set up a Lambda function to store the images as well as the image metadata in a DynamoDB table. Have the Lambda function retrieve previously uploaded images from DynamoDB**

- ( ) **Use Cognito identity pools to create an IAM user for each user of the application during the sign-up process. Leverage IAM authentication in API Gateway to control access to the API. Set up a Lambda function to store the images in Amazon S3 and save the image object's S3 key as part of the photo details in a DynamoDB table. Have the Lambda function retrieve previously uploaded images by querying DynamoDB for the S3 key**

Explanation
Correct option:

Leverage Cognito user pools to manage user accounts and set up an Amazon Cognito user pool authorizer in API Gateway to control access to the API. Set up a Lambda function to store the images in Amazon S3 and save the image object's S3 key as part of the photo details in a DynamoDB table. Have the Lambda function retrieve previously uploaded images by querying DynamoDB for the S3 key

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. Your users can also sign in through social identity providers like Google, Facebook, Amazon, or Apple, and SAML identity providers. 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 a Software Development Kit (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, as well as sign-in with SAML 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.

To use an Amazon Cognito user pool with your Amazon API Gateway API, you must first create an authorizer of the COGNITO_USER_POOLS type and then configure an API method to use that authorizer. After the API is deployed, the client must first sign the user into the user pool, obtain an identity or access token for the user, and then call the API method with one of the tokens, which are typically set to the request's Authorization header.

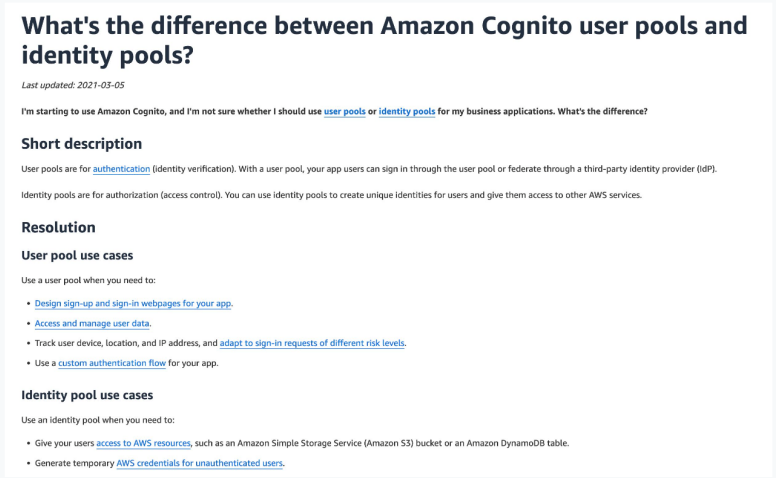
For the given use case, you can use a Cognito user pool to manage user accounts and configure an Amazon Cognito user pool authorizer in API Gateway to control access to the API. You should use a Lambda function to store the actual images on S3 and the image metadata on DynamoDB. Finally, you can get the images using the Lambda function that leverages the metadata stored in DynamoDB.

Incorrect options:

Use Cognito identity pools to manage user accounts and set up an Amazon Cognito identity pool authorizer in API Gateway to control access to the API. Set up a Lambda function to store the images in Amazon S3 and save the image object's S3 key as part of the photo details in a DynamoDB table. Have the Lambda function retrieve previously uploaded images by querying DynamoDB for the S3 key

Use Cognito identity pools to create an IAM user for each user of the application during the sign-up process. Leverage IAM authentication in API Gateway to control access to the API. Set up a Lambda function to store the images in Amazon S3 and save the image object's S3 key as part of the photo details in a DynamoDB table. Have the Lambda function retrieve previously uploaded images by querying DynamoDB for the S3 key

Amazon Cognito identity pools (federated identities) enable you to create unique identities for your users and federate them with identity providers. With an identity pool, you can obtain temporary, limited-privilege AWS credentials to access other AWS services. You cannot use identity pools to manage users or to create IAM users. So both of these options are incorrect.



What's the difference between Amazon Cognito user pools and identity pools?

Last updated: 2021-03-05

I'm starting to use Amazon Cognito, and I'm not sure whether I should use user pools or identity pools for my business applications. What's the difference?

Short description

User pools are for authentication (identity verification). With a user pool, your app users can sign in through the user pool or federate through a third-party identity provider (IdP).

Identity pools are for authorization (access control). You can use identity pools to create unique identities for users and give them access to other AWS services.

Resolution

User pool use cases

Use a user pool when you need to:

- Design sign-up and sign-in webpages for your app.
- Access and manage user data.
- Track user device, location, and IP address, and adapt to sign-in requests of different risk levels.
- Use a custom authentication flow for your app.

Identity pool use cases

Use an identity pool when you need to:

- Give your users access to AWS resources, such as an Amazon Simple Storage Service (Amazon S3) bucket or an Amazon DynamoDB table.
- Generate temporary AWS credentials for unauthenticated users.

via - https://aws.amazon.com/premiumsupport/knowledge-center/cognito-user-pools-identity-pools/

Leverage Cognito user pools to manage user accounts and set up an Amazon Cognito user pool authorizer in API Gateway to control access to the API. Set up a Lambda function to store the images as well as the image metadata in a DynamoDB table. Have the Lambda function retrieve previously uploaded images from DynamoDB - You cannot use DynamoDB to store images as the maximum allowed item size is 400KB and the images range in size from 500KB to 5MB. You should also note that storing images on DynamoDB is an anti-pattern. So this option is incorrect.

References:

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

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

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

https://aws.amazon.com/premiumsupport/knowledge-center/cognito-user-pools-identity-pools/

Question 17: **Incorrect**

An Accounting firm extensively uses Amazon EBS volumes for persistent storage of application data of Amazon EC2 instances. The volumes are encrypted to protect the critical data of the clients. As part of managing the security credentials, the project manager has come across a policy snippet that looks like the following:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Allow for use of this Key",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/UserRole"
            },
            "Action": [
                "kms:GenerateDataKeyWithoutPlaintext",
                "kms:Decrypt"
            ],
            "Resource": "*"
        },
        {
            "Sid": "Allow for EC2 Use",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/UserRole"
            },
            "Action": [
                "kms:CreateGrant",
                "kms:ListGrants",
                "kms:RevokeGrant"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                "kms:ViaService": "ec2.us-west-2.amazonaws.com"
            }
        }
    ]
}
```

- ○ **The second statement in this policy provides the security group (mentioned in first statement of the policy), the ability to create, list, and revoke grants for Amazon EC2**
- ○ **The first statement provides the security group the ability to generate a data key and decrypt that data key from the CMK when necessary**
- ○ **The first statement provides a specified IAM principal the ability to generate a data key and decrypt that data key from the CMK when necessary** (Correct)
- ◉ **The second statement in the policy mentions that all the resources stated in the first statement can take the specified role which will provide the ability to create, list, and revoke grants for Amazon EC2** (Incorrect)

Explanation
Correct option:

The first statement provides a specified IAM principal the ability to generate a data key and decrypt that data key from the CMK when necessary - To create and use an encrypted Amazon Elastic Block Store (EBS) volume, you need permissions to use Amazon EBS. The key policy associated with the CMK would need to include these. The above policy is an example of one such policy.

In this CMK policy, the first statement provides a specified IAM principal the ability to generate a data key and decrypt that data key from the CMK when necessary. These two APIs are necessary to encrypt the EBS volume while it's attached to an Amazon Elastic Compute Cloud (EC2) instance.

The second statement in this policy provides the specified IAM principal the ability to create, list, and revoke grants for Amazon EC2. Grants are used to delegate a subset of permissions to AWS services, or other principals, so that they can use your keys on your behalf. In this case, the condition policy explicitly ensures that only Amazon EC2 can use the grants. Amazon EC2 will use them to re-attach an encrypted EBS volume back to an instance if the volume gets detached due to a planned or unplanned outage. These events will be recorded within AWS CloudTrail when, and if, they do occur for your auditing.

Incorrect options:

The first statement provides the security group the ability to generate a data key and decrypt that data key from the CMK when necessary

The second statement in this policy provides the security group (mentioned in the first statement of the policy), the ability to create, list, and revoke grants for Amazon EC2

The second statement in the policy mentions that all the resources stated in the first statement can take the specified role which will provide the ability to create, list, and revoke grants for Amazon EC2

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

Reference:

https://d0.awsstatic.com/whitepapers/aws-kms-best-practices.pdf

Question 18: **Correct**

A company uses AWS CodeDeploy to deploy applications from GitHub to EC2 instances running Amazon Linux. The deployment process uses a file called appspec.yml for specifying deployment hooks. A final lifecycle event should be specified to verify the deployment success.

Which of the following hook events should be used to verify the success of the deployment?

- ○ **AfterInstall**
- ◉ **ValidateService** **(Correct)**
- ○ **ApplicationStart**
- ○ **AllowTraffic**

Explanation

Correct option:

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.

An EC2/On-Premises deployment hook is executed once per deployment to an instance. You can specify one or more scripts to run in a hook.

Lifecycle Event Hook Availability

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

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

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

ValidateService: ValidateService is the last deployment lifecycle event. It is used to verify the deployment was completed successfully.

Incorrect options:

AfterInstall - You can use this deployment lifecycle event for tasks such as configuring your application or changing file permissions

ApplicationStart - You typically use this deployment lifecycle event to restart services that were stopped during ApplicationStop

AllowTraffic - During this deployment lifecycle event, internet traffic is allowed to access instances after a deployment. This event is reserved for the AWS CodeDeploy agent and cannot be used to run scripts

Reference:

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

Question 19: **Correct**

Recently in your organization, the AWS X-Ray SDK was bundled into each Lambda function to record outgoing calls for tracing purposes. When your team leader goes to the X-Ray service in the AWS Management Console to get an overview of the information collected, they discover that no data is available.

What is the most likely reason for this issue?

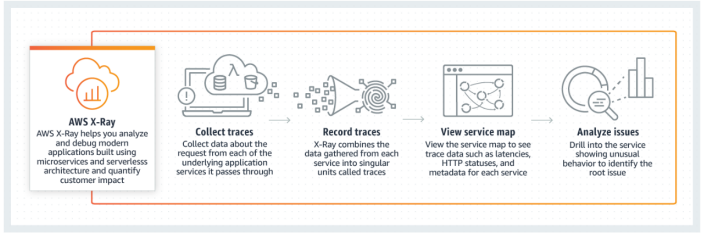
- ◯ **X-Ray only works with AWS Lambda aliases**
- ◉ **Fix the IAM Role** **(Correct)**
- ◯ **Enable X-Ray sampling**
- ◯ **Change the security group rules**

Explanation

Correct option:

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

How X-Ray Works:



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

Fix the IAM Role

Create an IAM role with write permissions and assign it to the resources running your application. You can use AWS Identity and Access Management (IAM) to grant X-Ray permissions to users and compute resources in your account. This should be one of the first places you start by checking that your permissions are properly configured before exploring other troubleshooting options.

Here is an example of X-Ray Read-Only permissions via an IAM policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries",
                "xray:BatchGetTraces",
                "xray:GetServiceGraph",
                "xray:GetTraceGraph",
                "xray:GetTraceSummaries",
                "xray:GetGroups",
                "xray:GetGroup"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Another example of write permissions for using X-Ray via an IAM policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "xray:PutTraceSegments",
                "xray:PutTelemetryRecords",
                "xray:GetSamplingRules",
                "xray:GetSamplingTargets",
                "xray:GetSamplingStatisticSummaries"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

Incorrect options:

Enable X-Ray sampling - If permissions are not configured correctly sampling will not work, so this option is not correct.

X-Ray only works with AWS Lambda aliases - This is not true, aliases are pointers to specific Lambda function versions. To use the X-Ray SDK on Lambda, bundle it with your function code each time you create a new version.

Change the security group rules - You grant permissions to your Lambda function to access other resources using an IAM role and not via security groups.

Question 20: **Correct**

You are a developer working on a web application written in Java and would like to use AWS Elastic Beanstalk for deployment because it would handle deployment, capacity provisioning, load balancing, auto-scaling, and application health monitoring. In the past, you connected to your provisioned instances through SSH to issue configuration commands. Now, you would like a configuration mechanism that automatically applies settings for you.

Which of the following options would help do this?

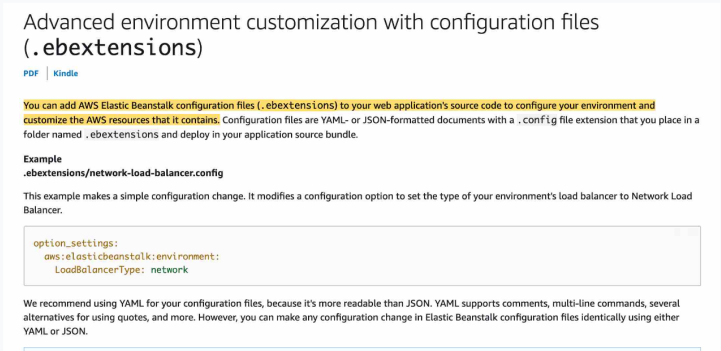
- (●) **Include config files in .ebextensions/ at the root of your source code** **(Correct)**
- ( ) **Use an AWS Lambda hook**
- ( ) **Deploy a CloudFormation wrapper**
- ( ) **Use SSM parameter store as an input to your Elastic Beanstalk Configurations**

Explanation

Correct option:

Include config files in .ebextensions/ at the root of your source code

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.



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

PDF | Kindle

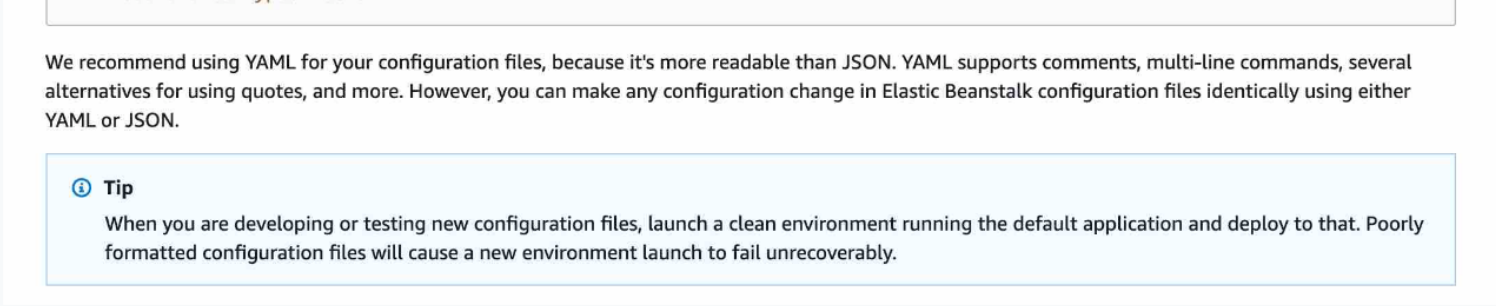
You can add AWS Elastic Beanstalk configuration files (`.ebextensions`) to your web application's source code to configure your environment and customize the AWS resources that it contains. Configuration files are YAML- or JSON-formatted documents with a `.config` file extension that you place in a folder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



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

Incorrect options:

Deploy a CloudFormation wrapper - This is a made-up option. This has been added as a distractor.

Use SSM parameter store as an input to your Elastic Beanstalk Configurations - SSM parameter is still not supported for Elastic Beanstalk. So this option is incorrect.

Use an AWS Lambda hook - Lambda functions are not the best-fit to trigger these configuration changes as it would involve significant development effort.

References:

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

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

Question 21: **Incorrect**

A serverless application built on AWS processes customer orders 24/7 using an AWS Lambda function and communicates with an external vendor's HTTP API for payment processing. The development team wants to notify the support team in near real-time using an existing Amazon Simple Notification Service (Amazon SNS) topic, but only when the external API error rate exceeds 5% of the total transactions processed in an hour.

As an AWS Certified Developer Associate, which option will you suggest as the most efficient solution?

- ○ **Log the results of payment processing API calls to Amazon CloudWatch. Leverage Amazon CloudWatch Metric Filter to look at the CloudWatch logs. Set up the Lambda function to check the output from CloudWatch Metric Filter on a schedule and send notification via the existing SNS topic when the error rate exceeds the specified rate**

- ○ **Configure CloudWatch metrics with detailed monitoring for the external payment processing API calls. Create a CloudWatch alarm that sends a notification via the existing SNS topic when the error rate exceeds the specified rate**

- ◉ **Log the results of payment processing API calls to Amazon CloudWatch. Leverage Amazon CloudWatch Logs Insights to query the CloudWatch logs. Set up the Lambda function to check the output from CloudWatch Logs Insights on a schedule and send notification via the existing SNS topic when the error rate exceeds the specified rate** **(Incorrect)**

- ○ **Configure and push high-resolution custom metrics to CloudWatch that record the failures of the external payment processing API calls. Create a CloudWatch alarm that sends a notification via the existing SNS topic when the error rate exceeds the specified rate** **(Correct)**

Explanation
Correct option:

Configure and push high-resolution custom metrics to CloudWatch that record the failures of the external payment processing API calls. Create a CloudWatch alarm that sends a notification via the existing SNS topic when the error rate exceeds the specified rate

You can publish your own metrics, known as custom metrics, to CloudWatch using the AWS CLI or an API.

Each metric is one of the following:

Standard resolution, with data having a one-minute granularity

High resolution, with data at a granularity of one second

Metrics produced by AWS services are standard resolution by default. When you publish

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

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

You can create metric and composite alarms in Amazon CloudWatch. For the given use case, you can set up a CloudWatch metric alarm that watches the custom metric that captures the API errors and then triggers the alarm when the API error rate exceeds the 5% threshold. The alarm then sends a notification via the existing SNS topic.

Incorrect options:

Configure CloudWatch metrics with detailed monitoring for the external payment processing API calls. Create a CloudWatch alarm that sends a notification via the existing SNS topic when the error rate exceeds the specified rate - CloudWatch provides two categories of monitoring: basic monitoring and detailed monitoring. Detailed monitoring options differ based on the services that offer it. For example, Amazon EC2 detailed monitoring provides more frequent metrics, published at one-minute intervals, instead of the five-minute intervals used in Amazon EC2 basic monitoring. Detailed monitoring is offered by only some services. As explained above, you need to use custom metrics to capture data for the external payment processing API calls since detailed monitoring for the standard CloudWatch metrics cannot be used for this scenario.

Log the results of payment processing API calls to Amazon CloudWatch. Leverage Amazon CloudWatch Logs Insights to query the CloudWatch logs. Set up the Lambda function to check the output from CloudWatch Logs Insights on a schedule and send notification via the existing SNS topic when the error rate exceeds the specified rate - CloudWatch Logs Insights enables you to interactively search and analyze your log data in Amazon CloudWatch Logs. You can perform queries to help you more efficiently and effectively respond to operational issues. This option is not the right fit for the given use case since Lambda cannot monitor the output of the CloudWatch Logs Insights on a real-time basis since it is being invoked on a schedule. Also, it is not an efficient solution since Lambda will need significant custom code to parse and compute the external API error rate from the CloudWatch Logs Insights data.

Log the results of payment processing API calls to Amazon CloudWatch. Leverage Amazon CloudWatch Metric Filter to look at the CloudWatch logs. Set up the Lambda function to check the output from CloudWatch Metric Filter on a schedule and send notification via the existing SNS topic when the error rate exceeds the specified rate - You can search and filter the log data coming into CloudWatch Logs by creating one or more metric filters. Metric filters define the terms and patterns to look for in log data as it is sent to CloudWatch Logs. CloudWatch Logs uses these metric filters to turn log data into numerical CloudWatch metrics that you can graph or set an alarm on. This option is not the best fit for the given use case since Lambda cannot monitor the output of the CloudWatch Metric Filter on a real-time basis since it is being invoked on a

schedule. Also, it is not an efficient solution since Lambda will need significant custom code to parse and compute the external API error rate from the CloudWatch Metric Filter data.

References:

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

https://aws.amazon.com/premiumsupport/knowledge-center/cloudwatch-push-custom-metrics/

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

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

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

Question 22: **Correct**

A business has their test environment built on Amazon EC2 configured on General purpose SSD volume.

At which gp2 volume size will their test environment hit the max IOPS?

- (●) **5.3 TiB** **(Correct)**
- ( ) **16 TiB**
- ( ) **2.7 TiB**
- ( ) **10.6 TiB**

Explanation
Correct option:

The performance of gp2 volumes is tied to volume size, which determines the baseline performance level of the volume and how quickly it accumulates I/O credits; larger volumes have higher baseline performance levels and accumulate I/O credits faster.

5.3 TiB - General Purpose SSD (gp2) volumes offer cost-effective storage that is ideal for a broad range of workloads. These volumes deliver single-digit millisecond latencies and the ability to burst to 3,000 IOPS for extended periods of time. Between a minimum of 100 IOPS (at 33.33 GiB and below) and a maximum of 16,000 IOPS (at 5,334 GiB and above), baseline performance scales linearly at 3 IOPS per GiB of volume size.

Maximum IOPS vs Volume Size for General Purpose SSD (gp2) volumes:  via - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html

Incorrect options:

10.6 TiB - As explained above, this is an incorrect option.

16 TiB - As explained above, this is an incorrect option.

2.7 TiB - As explained above, this is an incorrect option.

Reference:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html

Question 23: **Correct**

As an AWS certified developer associate, you are working on an AWS CloudFormation template that will create resources for a company's cloud infrastructure. Your template is composed of three stacks which are Stack-A, Stack-B, and Stack-C. Stack-A will provision a VPC, a security group, and subnets for public web applications that will be referenced in Stack-B and Stack-C.

After running the stacks you decide to delete them, in which order should you do it?

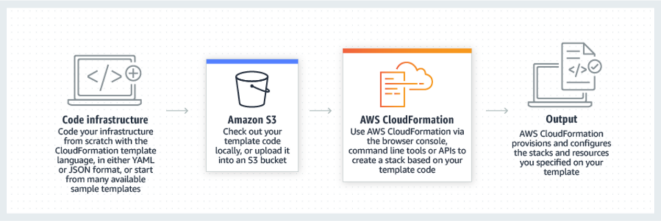
- ○ **Stack A, then Stack B, then Stack C**
- ○ **Stack A, Stack C then Stack B**
- ◉ **Stack B, then Stack C, then Stack A** (Correct)
- ○ **Stack C then Stack A then Stack B**

Explanation

Correct option:

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

How CloudFormation Works:



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

Stack B, then Stack C, then Stack A

All of the imports must be removed before you can delete the exporting stack or modify the output value. In this case, you must delete Stack B as well as Stack C, before you delete Stack A.

Exporting stack output values

PDF | Kindle | RSS

To share information between stacks, export a stack's output values. Other stacks that are in the same AWS account and region can import the exported values. For example, you might have a single networking stack that exports the IDs of a subnet and security group for public web servers. Stacks with a public web server can easily import those networking resources. You don't need to hard code resource IDs in the stack's template or pass IDs as input parameters.

To export a stack's output value, use the `Export` field in the Output section of the stack's template. To import those values, use the Fn::ImportValue function in the template for the other stacks. For a walkthrough and sample templates, see Walkthrough: Refer to resource outputs in another AWS CloudFormation stack.

Note

After another stack imports an output value, you can't delete the stack that is exporting the output value or modify the exported output value. All of the imports must be removed before you can delete the exporting stack or modify the output value.

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

Incorrect options:

Stack A, then Stack B, then Stack C - All of the imports must be removed before you can delete the exporting stack or modify the output value. In this case, you cannot delete Stack A first because that's being referenced in the other Stacks.

Stack A, Stack C then Stack B - All of the imports must be removed before you can delete the exporting stack or modify the output value. In this case, you cannot delete Stack A first because that's being referenced in the other Stacks.

Stack C then Stack A then Stack B - Stack C is fine but you should delete Stack B before Stack A because all of the imports must be removed before you can delete the exporting stack or modify the output value.

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

Question 24: **Correct**

The development team at an analytics company is using SQS queues for decoupling the various components of application architecture. As the consumers need additional time to process SQS messages, the development team wants to postpone the delivery of new messages to the queue for a few seconds.

As a Developer Associate, which of the following solutions would you recommend to the development team?

- ◯ **Use visibility timeout to postpone the delivery of new messages to the queue for a few seconds**
- ◉ **Use delay queues to postpone the delivery of new messages to the queue for a few seconds** **(Correct)**
- ◯ **Use FIFO queues to postpone the delivery of new messages to the queue for a few seconds**
- ◯ **Use dead-letter queues to postpone the delivery of new messages to the queue for a few seconds**

Explanation
Correct option:

Use delay queues to postpone the delivery of new messages to the queue for a few seconds

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

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

Amazon SQS delay queues

PDF | Kindle | RSS

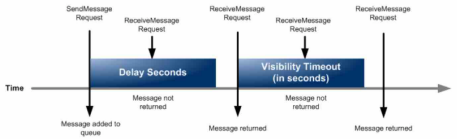
Delay queues let you postpone the delivery of new messages to a queue for a number of seconds, for example, when your consumer application needs additional time to process messages. If you create a delay queue, any messages that you send to the queue remain invisible to consumers for the duration of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

Note

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

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

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



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

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

Incorrect options:

Use FIFO queues to postpone the delivery of new messages to the queue for a few seconds - SQS FIFO queues are designed to guarantee that messages are processed exactly once, in the exact order that they are sent. You cannot use FIFO queues to postpone the delivery of new messages to the queue for a few seconds.

Use dead-letter queues to postpone the delivery of new messages to the queue for a few seconds - Dead-letter queues can be used by other queues (source queues) as a target for messages that can't be processed (consumed) successfully. Dead-letter queues are useful for debugging your application or messaging system because they let you isolate problematic messages to determine why their processing doesn't succeed. You cannot use dead-letter queues to postpone the delivery of new messages to the queue for a few seconds.

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

Reference:

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

Question 25: **Correct**

A company wants to share information with a third party via an HTTP API endpoint managed by the third party. The company has the necessary API key to access the endpoint and the integration of the API key with the company's application code must not impact the application's performance.

What is the most secure approach?

- ○ **Keep the API credentials in a local code variable and use the local code variable at runtime to make the API call**
- ◉ **Keep the API credentials in AWS Secrets Manager and use the credentials to make the API call by fetching the API credentials at runtime by using the AWS SDK** **(Correct)**
- ○ **Keep the API credentials in an encrypted table in MySQL RDS and use the credentials to make the API call by fetching the API credentials from RDS at runtime by using the AWS SDK**
- ○ **Keep the API credentials in an encrypted file in S3 and use the credentials to make the API call by fetching the API credentials from S3 at runtime by using the AWS SDK**

Explanation

Correct option:

Keep the API credentials in AWS Secrets Manager and use the credentials to make the API call by fetching the API credentials at runtime by using the AWS SDK

Secrets Manager enables you to replace hardcoded credentials in your code, including passwords, with an API call to Secrets Manager to retrieve the secret programmatically. This helps ensure the secret can't be compromised by someone examining your code, because the secret no longer exists in the code. Also, you can configure Secrets Manager to automatically rotate the secret for you according to a specified schedule. This enables you to replace long-term secrets with short-term ones, significantly reducing the risk of compromise.

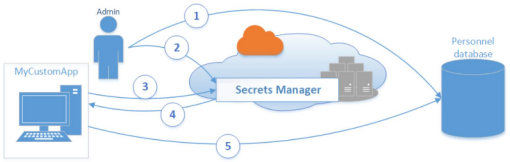
Basic AWS Secrets Manager scenario

The following diagram illustrates the most basic scenario. The diagram displays you can store credentials for a database in Secrets Manager, and then use those credentials in an application to access the database.

1. The database administrator creates a set of credentials on the Personnel database for use by an application called MyCustomApp. The administrator also configures those credentials with the permissions required for the application to access the Personnel database.
2. The database administrator stores the credentials as a secret in Secrets Manager named *MyCustomAppCreds*. Then, Secrets Manager encrypts and stores the credentials within the secret as the *protected secret text*.
3. When MyCustomApp accesses the database, the application queries Secrets Manager for the secret named *MyCustomAppCreds*.
4. Secrets Manager retrieves the secret, decrypts the protected secret text, and returns the secret to the client app over a secured (HTTPS with TLS) channel.

Basic AWS Secrets Manager scenario

The following diagram illustrates the most basic scenario. The diagram displays you can store credentials for a database in Secrets Manager, and then use those credentials in an application to access the database.



1. The database administrator creates a set of credentials on the Personnel database for use by an application called MyCustomApp. The administrator also configures those credentials with the permissions required for the application to access the Personnel database.
2. The database administrator stores the credentials as a secret in Secrets Manager named `MyCustomAppCreds`. Then, Secrets Manager encrypts and stores the credentials within the secret as the *protected secret text*.
3. When MyCustomApp accesses the database, the application queries Secrets Manager for the secret named `MyCustomAppCreds`.
4. Secrets Manager retrieves the secret, decrypts the protected secret text, and returns the secret to the client app over a secured (HTTPS with TLS) channel.
5. The client application parses the credentials, connection string, and any other required information from the response and then uses the information to access the database server.

Note

Secrets Manager supports many types of secrets. However, Secrets Manager can *natively* rotate credentials for supported AWS databases without any additional programming. However, rotating the secrets for other databases or services requires creating a custom Lambda function to define how Secrets Manager interacts with the database or service. You need some programming skill to create the function. For more information, see Rotate AWS Secrets Manager secrets.

via - https://docs.aws.amazon.com/secretsmanager/latest/userguide/intro.html

In the past, when you created a custom application to retrieve information from a database, you typically embedded the credentials, the secret, for accessing the database directly in the application. When the time came to rotate the credentials, you had to do more than just create new credentials. You had to invest time to update the application to use the new credentials. Then you distributed the updated application. If you had multiple applications with shared credentials and you missed updating one of them, the application failed. Because of this risk, many customers choose not to regularly rotate credentials, which effectively substitutes one risk for another. You can also use caching with Secrets Manager to significantly improve the availability and latency of applications.

Incorrect options:

Keep the API credentials in an encrypted table in MySQL RDS and use the credentials to make the API call by fetching the API credentials from RDS at runtime by using the AWS SDK

Keep the API credentials in an encrypted file in S3 and use the credentials to make the API call by fetching the API credentials from S3 at runtime by using the AWS SDK

Keep the API credentials in a local code variable and use the local code variable at runtime to make the API call

It is considered a security bad practice to keep sensitive access credentials in code, database, or a flat file on a file system or object storage. Therefore, all three options are incorrect.

Question 26: **Correct**

The development team at a multi-national retail company wants to support trusted third-party authenticated users from the supplier organizations to create and update records in specific DynamoDB tables in the company's AWS account.

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

- ○ **Create a new IAM group in the company's AWS account for each of the third-party authenticated users from the supplier organizations. The users can then use the IAM group credentials to access DynamoDB**
- ○ **Create a new IAM user in the company's AWS account for each of the third-party authenticated users from the supplier organizations. The users can then use the IAM user credentials to access DynamoDB**
- ◉ **Use Cognito Identity pools to enable trusted third-party authenticated users to access DynamoDB** **(Correct)**
- ○ **Use Cognito User pools to enable trusted third-party authenticated users to access DynamoDB**

Explanation

Correct option:

Use Cognito Identity pools to enable trusted third-party authenticated users to access DynamoDB

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

Public providers: Login with Amazon (Identity Pools), Facebook (Identity Pools), Google (Identity Pools), Sign in with Apple (Identity Pools).

Amazon Cognito User Pools

Open ID Connect Providers (Identity Pools)

SAML Identity Providers (Identity Pools)

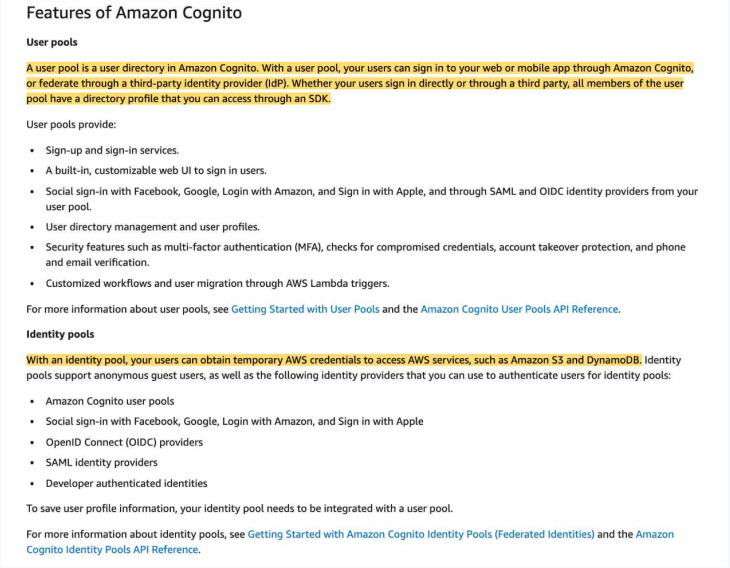
Developer Authenticated Identities (Identity Pools)

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



Features of Amazon Cognito

User pools

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

User pools provide:

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

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

Identity pools

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

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

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

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

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

Incorrect options:

Use Cognito User pools to enable trusted third-party authenticated users to access DynamoDB - 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). Cognito User Pools cannot be used to obtain temporary AWS credentials to access AWS services, such as Amazon S3 and DynamoDB.

Create a new IAM user in the company's AWS account for each of the third-party authenticated users from the supplier organizations. The users can then use the IAM user credentials to access DynamoDB

Create a new IAM group in the company's AWS account for each of the third-party authenticated users from the supplier organizations. The users can then use the IAM group credentials to access DynamoDB

Both these options involve setting up IAM resources such as IAM users or IAM groups just to provide access to DynamoDB tables. As the users are already trusted third-party authenticated users, Cognito Identity Pool can address this use-case in an elegant way.

Reference:

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

Question 27: **Correct**

A developer wants to securely store and retrieve various types of variables, such as remote API authentication information, API URL, and related credentials across different environments of an application deployed on Amazon Elastic Container Service (Amazon ECS).

What would be the best approach that needs minimal modifications in the application code?

- ○ **Configure the application to fetch the variables from an encrypted file that is stored with the application by storing the API URL and credentials in unique files for each environment**
- ○ **Configure the application to fetch the variables from AWS KMS by storing the API URL and credentials as unique keys in KMS for each environment**
- ◉ **Configure the application to fetch the variables and credentials from AWS Systems Manager Parameter Store by leveraging hierarchical unique paths in Parameter Store for each variable in each environment** **(Correct)**
- ○ **Configure the application to fetch the variables from each of the deployed environments by defining the authentication information and API URL in the ECS task definition as unique names during the deployment process**

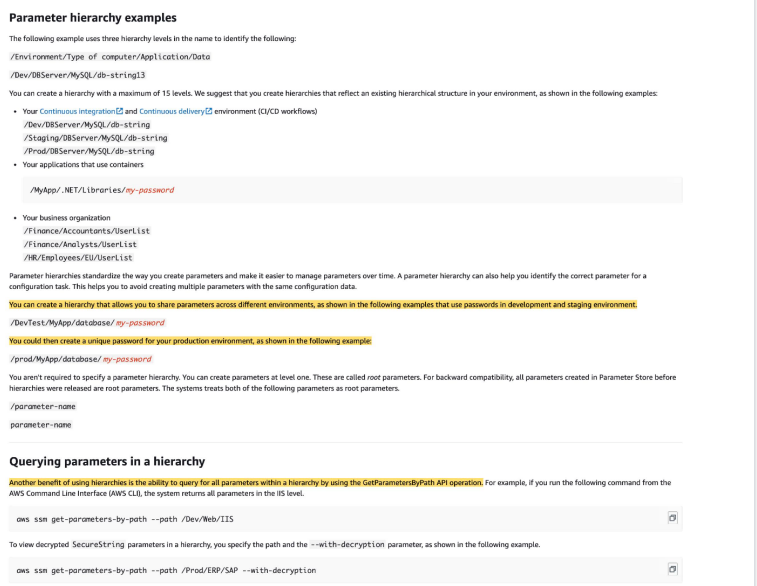
Explanation
Correct option:

Configure the application to fetch the variables and credentials from AWS Systems Manager Parameter Store by leveraging hierarchical unique paths in Parameter Store for each variable in each environment

Parameter Stores is a capability of AWS Systems Manager that 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. You can reference Systems Manager parameters in your scripts, commands, SSM documents, and configuration and automation workflows by using the unique name that you specified when you created the parameter.

Managing dozens or hundreds of parameters as a flat list is time-consuming and prone to errors. It can also be difficult to identify the correct parameter for a task. This means you might accidentally use the wrong parameter, or you might create multiple parameters that use the same configuration data.

You can use parameter hierarchies to help you organize and manage parameters. A hierarchy is a parameter name that includes a path that you define by using forward slashes (/).



Parameter hierarchy examples

The following example uses three hierarchy levels in the name to identify the following:

`/Environment/Type of computer/Application/Data`

`/Dev/DBServer/MySQL/db-string13`

You can create a hierarchy with a maximum of 15 levels. We suggest that you create hierarchies that reflect an existing hierarchical structure in your environment, as shown in the following examples:

- Your Continuous integration and Continuous delivery environment (CI/CD workflows)
  `/Dev/DBServer/MySQL/db-string`
  `/Staging/DBServer/MySQL/db-string`
  `/Prod/DBServer/MySQL/db-string`
- Your applications that use containers

```
/MyApp/.NET/Libraries/my-password
```

- Your business organization
  `/Finance/Accountants/UserList`
  `/Finance/Analysts/UserList`
  `/HR/Employees/EU/UserList`

Parameter hierarchies standardize the way you create parameters and make it easier to manage parameters over time. A parameter hierarchy can also help you identify the correct parameter for a configuration task. This helps you to avoid creating multiple parameters with the same configuration data.

You can create a hierarchy that allows you to share parameters across different environments, as shown in the following examples that use passwords in development and staging environment.

`/DevTest/MyApp/database/my-password`

You could then create a unique password for your production environment, as shown in the following example:

`/prod/MyApp/database/my-password`

You aren't required to specify a parameter hierarchy. You can create parameters at level one. These are called *root* parameters. For backward compatibility, all parameters created in Parameter Store before hierarchies were released are root parameters. The systems treats both of the following parameters as root parameters.

`/parameter-name`

`parameter-name`

Querying parameters in a hierarchy

Another benefit of using hierarchies is the ability to query for all parameters within a hierarchy by using the GetParametersByPath API operation. For example, if you run the following command from the AWS Command Line Interface (AWS CLI), the system returns all parameters in the IIS level.

```
aws ssm get-parameters-by-path --path /Dev/Web/IIS
```

To view decrypted `SecureString` parameters in a hierarchy, you specify the path and the `--with-decryption` parameter, as shown in the following example.

```
aws ssm get-parameters-by-path --path /Prod/ERP/SAP --with-decryption
```

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

Incorrect options:

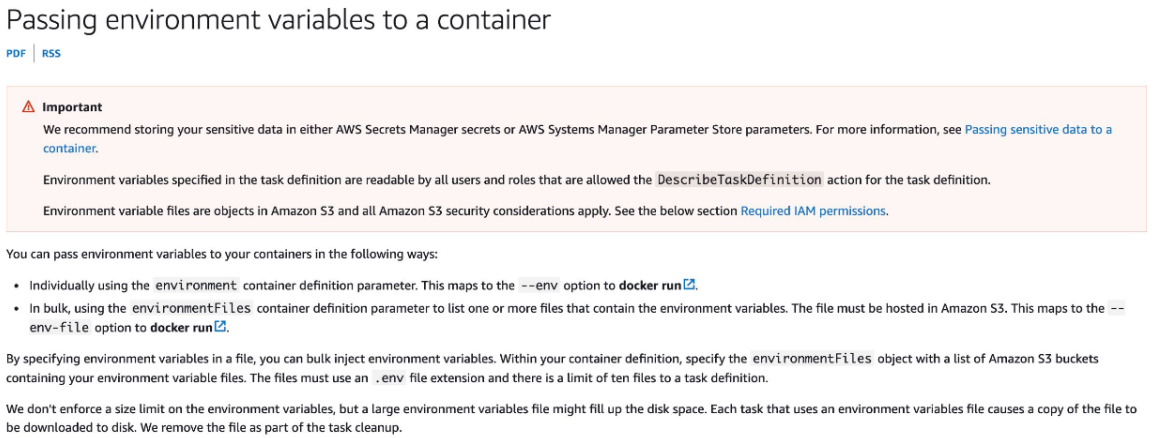
Configure the application to fetch the variables from AWS KMS by storing the API URL and credentials as unique keys in KMS for each environment - AWS KMS lets you create, manage, and control cryptographic keys across your applications and AWS services. KMS is not a key-value service that can be used for the given use case.

Configure the application to fetch the variables from an encrypted file that is stored with the application by storing the API URL and credentials in unique files for each environment - It is not considered a security best practice to store sensitive data and credentials in an encrypted file with the application. So this option is incorrect.

Configure the application to fetch the variables from each of the deployed environments by defining the authentication information and API URL in the ECS task definition as unique names during the deployment process - ECS task definition can be thought of as a blueprint for your application. Task definitions specify various parameters for your application. Examples of task definition parameters are which containers to use, which launch type to use, which ports should be opened for your application, and what data volumes should be used with the containers in the task. The specific parameters available for the task definition depend on which launch type you are using. The task definition is a text file, in JSON format, that describes one or more containers, up to a maximum of ten, that form your application. A task is the instantiation of a task definition within a cluster. After you create a task definition for your application within Amazon ECS, you can specify the number of tasks to run on your cluster.

within Amazon ECS, you can specify the number of tasks to run on your cluster.

AWS recommends storing your sensitive data in either AWS Secrets Manager secrets or AWS Systems Manager Parameter Store parameters. Environment variables specified in the task definition are readable by all users and roles that are allowed the DescribeTaskDefinition action for the task definition. So this option is incorrect.



Passing environment variables to a container

PDF | RSS

Important

We recommend storing your sensitive data in either AWS Secrets Manager secrets or AWS Systems Manager Parameter Store parameters. For more information, see Passing sensitive data to a container.

Environment variables specified in the task definition are readable by all users and roles that are allowed the `DescribeTaskDefinition` action for the task definition.

Environment variable files are objects in Amazon S3 and all Amazon S3 security considerations apply. See the below section Required IAM permissions.

You can pass environment variables to your containers in the following ways:

- Individually using the `environment` container definition parameter. This maps to the `--env` option to **docker run**.
- In bulk, using the `environmentFiles` container definition parameter to list one or more files that contain the environment variables. The file must be hosted in Amazon S3. This maps to the `--env-file` option to **docker run**.

By specifying environment variables in a file, you can bulk inject environment variables. Within your container definition, specify the `environmentFiles` object with a list of Amazon S3 buckets containing your environment variable files. The files must use an `.env` file extension and there is a limit of ten files to a task definition.

We don't enforce a size limit on the environment variables, but a large environment variables file might fill up the disk space. Each task that uses an environment variables file causes a copy of the file to be downloaded to disk. We remove the file as part of the task cleanup.

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

References:

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

https://aws.amazon.com/kms/

https://ecsworkshop.com/introduction/ecs_basics/task_definition/

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/taskdef-envfiles.html

Question 28: **Correct**

As a Developer, 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?

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

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.

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, the presence of the `Transform` section indicates, the document is a SAM template.

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, similar to below code:

References:

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/transform-aws-serverless.html

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

Question 29: **Correct**

What steps can a developer take to optimize the performance of a CPU-bound AWS Lambda function and ensure fast response time?

- ○ **Increase the function's CPU**
- ◉ **Increase the function's memory** **(Correct)**
- ○ **Increase the function's timeout**
- ○ **Increase the function's provisioned concurrency**

Explanation

Correct option:

Increase the function's memory

Memory is the principal lever available to Lambda developers for controlling the performance of a function. You can configure the amount of memory allocated to a Lambda function, between 128 MB and 10,240 MB. The Lambda console defaults new functions to the smallest setting and many developers also choose 128 MB for their functions.

The amount of memory also determines the amount of virtual CPU available to a function. Adding more memory proportionally increases the amount of CPU, increasing the overall computational power available. If a function is CPU-, network- or memory-bound, then changing the memory setting can dramatically improve its performance.

Incorrect options:

Increase the function's provisioned concurrency

Increase the function's reserved concurrency

In Lambda, concurrency is the number of requests your function can handle at the same time. There are two types of concurrency controls available:

Reserved concurrency – Reserved concurrency guarantees the maximum number of concurrent instances for the function. When a function has reserved concurrency, no other function can use that concurrency. There is no charge for configuring reserved concurrency for a function.

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

Neither reserved concurrency nor provisioned concurrency has any impact on the CPU available to a function, so both these options are incorrect

Increase the function's CPU - This is a distractor as you cannot directly increase the CPU available to a function.

References:

https://docs.aws.amazon.com/lambda/latest/operatorguide/computing-power.html

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

Question 30: **Correct**

A social gaming application supports the transfer of gift vouchers between users. When a user hits a certain milestone on the leaderboard, they earn a gift voucher that can be redeemed or transferred to another user. The development team wants to ensure that this transfer is captured in the database such that the records for both users are either written successfully with the new gift vouchers or the status quo is maintained.

Which of the following solutions represent the best-fit options to meet the requirements for the given use-case? (Select two)

- [x] **Complete both operations on RDS MySQL in a single transaction block** (Correct)
- [x] **Use the DynamoDB transactional read and write APIs on the table items as a single, all-or-nothing operation** (Correct)
- [ ] **Perform DynamoDB read and write operations with ConsistentRead parameter set to true**
- [ ] **Use the Amazon Athena transactional read and write APIs on the table items as a single, all-or-nothing operation**
- [ ] **Complete both operations on Amazon RedShift in a single transaction block**

Explanation
Correct option:

Use the DynamoDB transactional read and write APIs on the table items as a single, all-or-nothing operation

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

Question 30: **Correct**

A social gaming application supports the transfer of gift vouchers between users. When a user hits a certain milestone on the leaderboard, they earn a gift voucher that can be redeemed or transferred to another user. The development team wants to ensure that this transfer is captured in the database such that the records for both users are either written successfully with the new gift vouchers or the status quo is maintained.

Which of the following solutions represent the best-fit options to meet the requirements for the given use-case? (Select two)

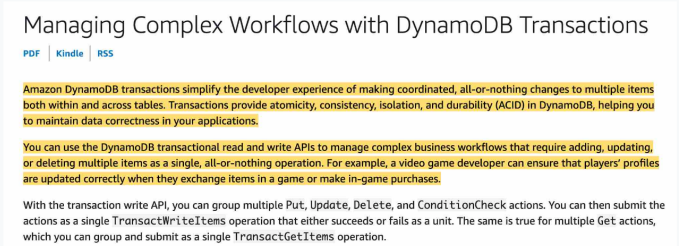
- [x] **Complete both operations on RDS MySQL in a single transaction block** **(Correct)**
- [x] **Use the DynamoDB transactional read and write APIs on the table items as a single, all-or-nothing operation** **(Correct)**
- [ ] **Perform DynamoDB read and write operations with ConsistentRead parameter set to true**
- [ ] **Use the Amazon Athena transactional read and write APIs on the table items as a single, all-or-nothing operation**
- [ ] **Complete both operations on Amazon RedShift in a single transaction block**

Explanation
Correct option:

Use the DynamoDB transactional read and write APIs on the table items as a single, all-or-nothing operation

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

DynamoDB Transactions Overview:



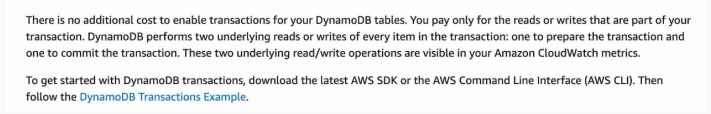
Managing Complex Workflows with DynamoDB Transactions

PDF | Kindle | RSS

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

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

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



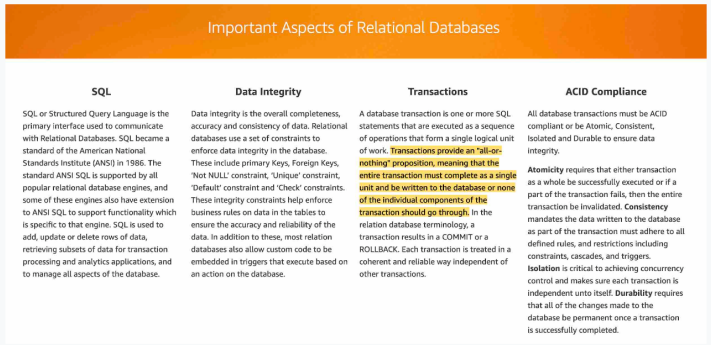
There is no additional cost to enable transactions for your DynamoDB tables. You pay only for the reads or writes that are part of your transaction. DynamoDB performs two underlying reads or writes of every item in the transaction: one to prepare the transaction and one to commit the transaction. These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

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

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

Complete both operations on RDS MySQL in a single transaction block

Amazon Relational Database Service (Amazon RDS) makes it easy to set up, operate, and scale a relational database with support for transactions in the cloud. A relational database is a collection of data items with pre-defined relationships between them. RDS supports the most demanding database applications. You can choose between two SSD-backed storage options: one optimized for high-performance Online Transaction Processing (OLTP) applications, and the other for cost-effective general-purpose use.



via - https://aws.amazon.com/relational-database/

Incorrect options:

Perform DynamoDB read and write operations with ConsistentRead parameter set to true - DynamoDB uses eventually consistent reads unless you specify otherwise. Read operations (such as GetItem, Query, and Scan) provide a ConsistentRead parameter. If you set this parameter to true, DynamoDB uses strongly consistent reads during the operation. Read consistency does not facilitate DynamoDB transactions and this option has been added as a distractor.

Complete both operations on Amazon RedShift in a single transaction block - Amazon Redshift is a fully-managed petabyte-scale cloud-based data warehouse product designed for large scale data set storage and analysis. It cannot be used to manage database transactions.

Use the Amazon Athena transactional read and write APIs on the table items as a single, all-or-nothing operation - Amazon Athena is an interactive query service that makes it easy to analyze data in Amazon S3 using standard SQL. Athena is serverless, so there is no infrastructure to manage, and you pay only for the queries that you run. It cannot be used to manage database transactions.

Question 31: **Correct**

A startup has been experimenting with DynamoDB in its new test environment. The development team has discovered that some of the write operations have been overwriting existing items that have the specified primary key. This has messed up their data, leading to data discrepancies.

Which DynamoDB write option should be selected to prevent this kind of overwriting?

- ○ **Use Scan operation**
- ○ **Atomic Counters**
- ◉ **Conditional writes** **(Correct)**
- ○ **Batch writes**

Explanation
Correct option:

Conditional writes - DynamoDB optionally supports conditional writes for write operations (PutItem, UpdateItem, DeleteItem). A conditional write succeeds only if the item attributes meet one or more expected conditions. Otherwise, it returns an error.

For example, you might want a PutItem operation to succeed only if there is not already an item with the same primary key. Or you could prevent an UpdateItem operation from modifying an item if one of its attributes has a certain value. Conditional writes are helpful in cases where multiple users attempt to modify the same item. This is the right choice for the current scenario.

Incorrect options:

Batch writes - Bath operations (read and write) help reduce the number of network round trips from your application to DynamoDB. In addition, DynamoDB performs the individual read or write operations in parallel. Applications benefit from this parallelism without having to manage concurrency or threading. But, this is of no use in the current scenario of overwriting changes.

Atomic Counters - Atomic Counters is a numeric attribute that is incremented, unconditionally, without interfering with other write requests. You might use an atomic counter to track the number of visitors to a website. This functionality is not useful for the current scenario.

Use Scan operation - A Scan operation in Amazon DynamoDB reads every item in a table or a secondary index. By default, a Scan operation returns all of the data attributes for every item in the table or index. This is given as a distractor and not related to DynamoDB item updates.

Reference:

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/WorkingWithItems.html#WorkingWithItems.ConditionalUpdate

Question 32: **Correct**

A data analytics company is processing real-time Internet-of-Things (IoT) data via Kinesis Producer Library (KPL) and sending the data to a Kinesis Data Streams driven application. The application has halted data processing because of a ProvisionedThroughputExceeded exception.

Which of the following actions would help in addressing this issue? (Select two)

- [x] **Configure the data producer to retry with an exponential backoff** (Correct)
- [ ] **Use Amazon Kinesis Agent instead of Kinesis Producer Library (KPL) for sending data to Kinesis Data Streams**
- [ ] **Use Kinesis enhanced fan-out for Kinesis Data Streams**
- [ ] **Use Amazon SQS instead of Kinesis Data Streams**
- [x] **Increase the number of shards within your data streams to provide enough capacity** (Correct)

Explanation

Correct option:

Configure the data producer to retry with an exponential backoff

Increase the number of shards within your data streams to provide enough capacity

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

How Kinesis Data Streams Work

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

The capacity limits of an Amazon Kinesis data stream are defined by the number of shards within the data stream. The limits can be exceeded by either data throughput or the number of PUT records. While the capacity limits are exceeded, the put data call will be rejected with a ProvisionedThroughputExceeded exception.

If this is due to a temporary rise of the data stream's input data rate, retry (with exponential backoff) by the data producer will eventually lead to the completion of the requests.

If this is due to a sustained rise of the data stream's input data rate, you should increase the number of shards within your data stream to provide enough capacity for the put data calls to consistently succeed.

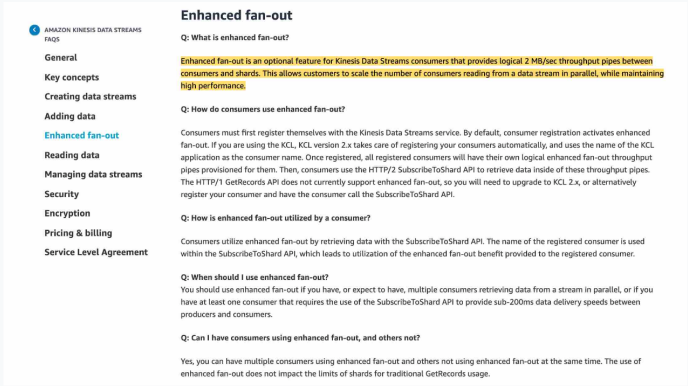
Incorrect options:

Use Amazon Kinesis Agent instead of Kinesis Producer Library (KPL) for sending data to Kinesis Data Streams - Kinesis Agent works with data producers. Using Kinesis Agent instead of KPL will not help as the constraint is the capacity limit of the Kinesis Data Stream.

Use Amazon SQS instead of Kinesis Data Streams - This is a distractor as using SQS will not help address the ProvisionedThroughputExceeded exception for the Kinesis Data Stream. This option does not address the issues in the use-case.

Use Kinesis enhanced fan-out for Kinesis Data Streams - You should use enhanced fan-out if you have, or expect to have, multiple consumers retrieving data from a stream in parallel. Therefore, using enhanced fan-out will not help address the ProvisionedThroughputExceeded exception as the constraint is the capacity limit of the Kinesis Data Stream.

Please review this note for more details on enhanced fan-out for Kinesis Data Streams:



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

References:

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

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

Question 33: **Incorrect**

A development team is building a game where players can buy items with virtual coins. For every virtual coin bought by a user, both the players table as well as the items table in DynamodDB need to be updated simultaneously using an all-or-nothing operation.

As a developer associate, how will you implement this functionality?

- (●) **Use `BatchWriteItem` API to update multiple tables simultaneously** **(Incorrect)**
- ( ) **Capture the transactions in the players table using DynamoDB streams and then sync with the items table**
- ( ) **Capture the transactions in the items table using DynamoDB streams and then sync with the players table**
- ( ) **Use `TransactWriteItems` API of DynamoDB Transactions** **(Correct)**

Explanation

Correct option: **Use `TransactWriteItems` API of DynamoDB Transactions**

With Amazon DynamoDB transactions, you can group multiple actions together and submit them as a single all-or-nothing `TransactWriteItems` or `TransactGetItems` operation.

`TransactWriteItems` is a synchronous and idempotent write operation that groups up to 25 write actions in a single all-or-nothing operation. These actions can target up to 25 distinct items in one or more DynamoDB tables within the same AWS account and in the same Region. The aggregate size of the items in the transaction cannot exceed 4 MB. The actions are completed atomically so that either all of them succeed or none of them succeeds.

You can optionally include a client token when you make a TransactWriteItems call to ensure that the request is idempotent. Making your transactions idempotent helps prevent application errors if the same operation is submitted multiple times due to a connection time-out or other connectivity issue.

Incorrect options:

Use `BatchWriteItem` API to update multiple tables simultaneously - A `TransactWriteItems` operation differs from a `BatchWriteItem` operation in that all the actions it contains must be completed successfully, or no changes are made at all. With a `BatchWriteItem` operation, it is possible that only some of the actions in the batch succeed while the others do not.

Capture the transactions in the players table using DynamoDB streams and then sync with the items table

Capture the transactions in the items table using DynamoDB streams and then

Explanation
Correct option: **Use `TransactWriteItems` API of DynamoDB Transactions**

With Amazon DynamoDB transactions, you can group multiple actions together and submit them as a single all-or-nothing `TransactWriteItems` or `TransactGetItems` operation.

`TransactWriteItems` is a synchronous and idempotent write operation that groups up to 25 write actions in a single all-or-nothing operation. These actions can target up to 25 distinct items in one or more DynamoDB tables within the same AWS account and in the same Region. The aggregate size of the items in the transaction cannot exceed 4 MB. The actions are completed atomically so that either all of them succeed or none of them succeeds.

You can optionally include a client token when you make a TransactWriteItems call to ensure that the request is idempotent. Making your transactions idempotent helps prevent application errors if the same operation is submitted multiple times due to a connection time-out or other connectivity issue.

Incorrect options:

Use `BatchWriteItem` API to update multiple tables simultaneously - A `TransactWriteItems` operation differs from a `BatchWriteItem` operation in that all the actions it contains must be completed successfully, or no changes are made at all. With a `BatchWriteItem` operation, it is possible that only some of the actions in the batch succeed while the others do not.

Capture the transactions in the players table using DynamoDB streams and then sync with the items table

Capture the transactions in the items table using DynamoDB streams and then sync with the players table

Many applications benefit from capturing changes to items stored in a DynamoDB table, at the point in time when such changes occur. DynamoDB supports streaming of item-level change data capture records in near-real-time. You can build applications that consume these streams and take action based on the contents. DynamoDB streams cannot be used to capture transactions in DynamoDB, therefore both these options are incorrect.

Reference:

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/transaction-apis.html#transaction-apis-txwriteitems

Question 34: **Correct**

While defining a business workflow as state machine on AWS Step Functions, a developer has configured several states.

Which of the following would you identify as the state that represents a single unit of work performed by a state machine?

○
```
"No-op": {
  "Type": "Task",
  "Result": {
    "x-datum": 0.381018,
    "y-datum": 622.2269926397355
  },
  "ResultPath": "$.coords",
  "Next": "End"
}
```

○
```
"FailState": {
  "Type": "Fail",
  "Cause": "Invalid response.",
  "Error": "ErrorA"
}
```

◉
```
"HelloWorld": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:HelloFunction",
  "Next": "AfterHelloWorldState",
  "Comment": "Run the HelloWorld Lambda function"
}
```
(Correct)

○
```
"wait_until" : {
  "Type": "Wait",
  "Timestamp": "2016-03-14T01:59:00Z",
  "Next": "NextState"
}
```

Explanation

Correct option:

```
"HelloWorld": {
  "Type": "Task",
  "Resource": "arn:aws:lambda:us-east-1:123456789012:function:HelloFunction",
  "Next": "AfterHelloWorldState",
  "Comment": "Run the HelloWorld Lambda function"
}
```

A Task state ("Type": "Task") represents a single unit of work performed by a state machine.

All work in your state machine is done by tasks. A task performs work by using an activity or an AWS Lambda function, or by passing parameters to the API actions of other services.

AWS Step Functions can invoke Lambda functions directly from a task state. A Lambda function is a cloud-native task that runs on AWS Lambda. You can write Lambda functions in a variety of programming languages, using the AWS Management Console or by uploading code to Lambda.

Incorrect options:

```
"wait_until" : {
  "Type": "Wait",
  "Timestamp": "2016-03-14T01:59:00Z",
  "Next": "NextState"
}
```

- A Wait state ("Type": "Wait") delays the state machine from continuing for a specified time.

```
"No-op": {
  "Type": "Task",
  "Result": {
    "x-datum": 0.381018,
    "y-datum": 622.2269926397355
  },
  "ResultPath": "$.coords",
  "Next": "End"
}
```

- `Resource` field is a required parameter for `Task` state. This definition is not of a `Task` but of type `Pass`.

```
"FailState": {
  "Type": "Fail",
  "Cause": "Invalid response.",
  "Error": "ErrorA"
}
```

- A Fail state ("Type": "Fail") stops the execution of the state machine and marks it as a failure unless it is caught by a Catch block. Because Fail states always exit the state machine, they have no Next field and don't require an End field.

Reference:

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

Question 35: **Correct**

You create an Auto Scaling group to work with an Application Load Balancer. The scaling group is configured with a minimum size value of 5, a maximum value of 20, and the desired capacity value of 10. One of the 10 EC2 instances has been reported as unhealthy.

Which of the following actions will take place?

- ○ **The ASG will detach the EC2 instance from the group, and leave it running**
- ◉ **The ASG will terminate the EC2 Instance** (Correct)
- ○ **The ASG will keep the instance running and re-start the application**
- ○ **The ASG will format the root EBS drive on the EC2 instance and run the User Data again**

Explanation
Correct option:

The ASG will terminate the EC2 Instance

To maintain the same number of instances, Amazon EC2 Auto Scaling performs a periodic health check on running instances within an Auto Scaling group. When it finds that an instance is unhealthy, it terminates that instance and launches a new one. Amazon EC2 Auto Scaling creates a new scaling activity for terminating the unhealthy instance and then terminates it. Later, another scaling activity launches a new instance to replace the terminated instance.

Incorrect options:

The ASG will detach the EC2 instance from the group, and leave it running - The goal of the auto-scaling group is to get rid of the bad instance and replace it

The ASG will keep the instance running and re-start the application - The ASG does not have control of your application

The ASG will format the root EBS drive on the EC2 instance and run the User Data again - This will not happen, the ASG cannot assume the format of your EBS drive, and User Data only runs once at instance first boot.

References:

https://aws.amazon.com/premiumsupport/knowledge-center/auto-scaling-terminate-instance

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

Question 36: **Correct**

As a senior architect, you are responsible for the development, support, maintenance, and implementation of all database applications written using NoSQL technology. A new project demands a throughput requirement of 10 strongly consistent reads per second of 6KB in size each.

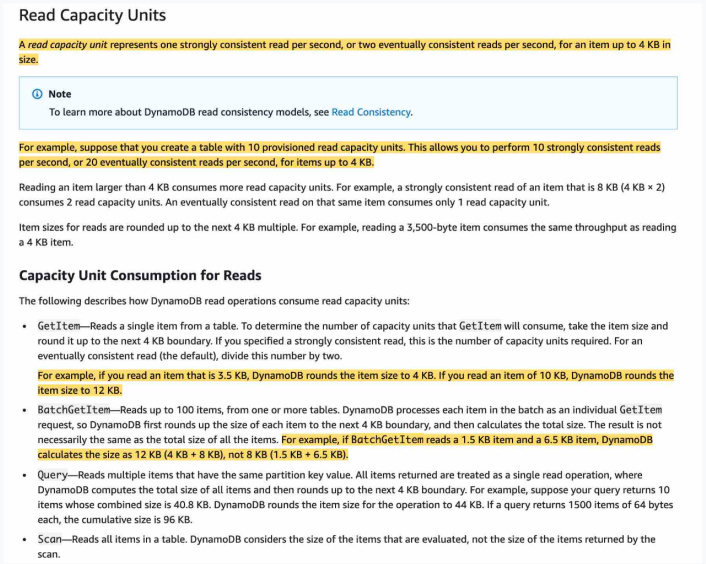
How many read capacity units will you need when configuring your DynamoDB table?

- ○ **60**
- ◉ **20** **(Correct)**
- ○ **30**
- ○ **10**

Explanation

Correct option:

Before proceeding with the calculations, please review the following:



Read Capacity Units

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

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

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

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

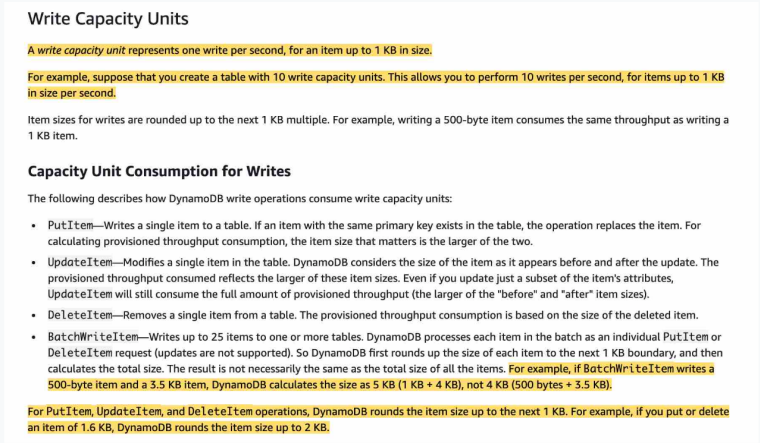
Item sizes for reads are rounded up to the next 4 KB multiple. For example, reading a 3,500-byte item consumes the same throughput as reading a 4 KB item.

Capacity Unit Consumption for Reads

The following describes how DynamoDB read operations consume read capacity units:

- `GetItem`—Reads a single item from a table. To determine the number of capacity units that `GetItem` will consume, take the item size and round it up to the next 4 KB boundary. If you specified a strongly consistent read, this is the number of capacity units required. For an eventually consistent read (the default), divide this number by two.

  For example, if you read an item that is 3.5 KB, DynamoDB rounds the item size to 4 KB. If you read an item of 10 KB, DynamoDB rounds the item size to 12 KB.
- `BatchGetItem`—Reads up to 100 items, from one or more tables. DynamoDB processes each item in the batch as an individual `GetItem` request, so DynamoDB first rounds up the size of each item to the next 4 KB boundary, and then calculates the total size. The result is not necessarily the same as the total size of all the items. For example, if `BatchGetItem` reads a 1.5 KB item and a 6.5 KB item, DynamoDB calculates the size as 12 KB (4 KB + 8 KB), not 8 KB (1.5 KB + 6.5 KB).
- `Query`—Reads multiple items that have the same partition key value. All items returned are treated as a single read operation, where DynamoDB computes the total size of all items and then rounds up to the next 4 KB boundary. For example, suppose your query returns 10 items whose combined size is 40.8 KB. DynamoDB rounds the item size for the operation to 44 KB. If a query returns 1500 items of 64 bytes each, the cumulative size is 96 KB.
- `Scan`—Reads all items in a table. DynamoDB considers the size of the items that are evaluated, not the size of the items returned by the scan.



Write Capacity Units

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

For example, suppose that you create a table with 10 write capacity units. This allows you to perform 10 writes per second, for items up to 1 KB in size per second.

Item sizes for writes are rounded up to the next 1 KB multiple. For example, writing a 500-byte item consumes the same throughput as writing a 1 KB item.

Capacity Unit Consumption for Writes

The following describes how DynamoDB write operations consume write capacity units:

- `PutItem`—Writes a single item to a table. If an item with the same primary key exists in the table, the operation replaces the item. For calculating provisioned throughput consumption, the item size that matters is the larger of the two.
- `UpdateItem`—Modifies a single item in the table. DynamoDB considers the size of the item as it appears before and after the update. The provisioned throughput consumed reflects the larger of these item sizes. Even if you update just a subset of the item's attributes, `UpdateItem` will still consume the full amount of provisioned throughput (the larger of the "before" and "after" item sizes).
- `DeleteItem`—Removes a single item from a table. The provisioned throughput consumption is based on the size of the deleted item.
- `BatchWriteItem`—Writes up to 25 items to one or more tables. DynamoDB processes each item in the batch as an individual `PutItem` or `DeleteItem` request (updates are not supported). So DynamoDB first rounds up the size of each item to the next 1 KB boundary, and then calculates the total size. The result is not necessarily the same as the total size of all the items. For example, if `BatchWriteItem` writes a 500-byte item and a 3.5 KB item, DynamoDB calculates the size as 5 KB (1 KB + 4 KB), not 4 KB (500 bytes + 3.5 KB).

For `PutItem`, `UpdateItem`, and `DeleteItem` operations, DynamoDB rounds the item size up to the next 1 KB. For example, if you put or delete an item of 1.6 KB, DynamoDB rounds the item size up to 2 KB.

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

20

One read capacity unit represents one strongly consistent read per second for an item up to 4 KB in size. If you need to read an item that is larger than 4 KB, DynamoDB will need to consume additional read capacity units.

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

So, in the above case, 6KB / 4 KB = 1.5 or 2 read capacity units.

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

So, in the above case, 2 x 10 = 20 read capacity units.

Incorrect options:

60

30

10

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

Reference:

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

Question 37: **Correct**

Your company has embraced cloud-native microservices architectures. New applications must be dockerized and stored in a registry service offered by AWS. The architecture should support dynamic port mapping and support multiple tasks from a single service on the same container instance. All services should run on the same EC2 instance.

Which of the following options offers the best-fit solution for the given use-case?

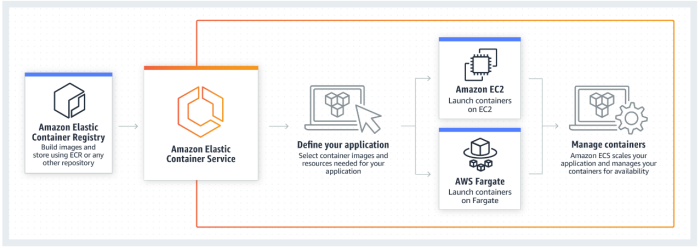
- ○ **Classic Load Balancer + ECS**
- ○ **Classic Load Balancer + Beanstalk**
- ○ **Application Load Balancer + Beanstalk**
- ◉ **Application Load Balancer + ECS** **(Correct)**

Explanation

Correct option:

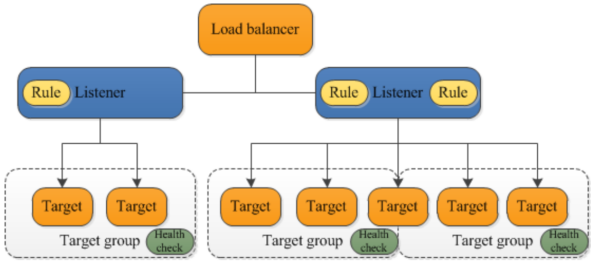
Application Load Balancer + ECS

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



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

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



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

When you deploy your services using Amazon Elastic Container Service (Amazon ECS), you can use dynamic port mapping to support multiple tasks from a single service on the same container instance. Amazon ECS manages updates to your services by automatically registering and deregistering containers with your target group using the instance ID and port for each container.

How do I set up dynamic port mapping for Amazon ECS?

Last updated: 2020-05-12

I want to set up dynamic port mapping for my container instance in Amazon Elastic Container Service (Amazon ECS).

Short Description

The Classic Load Balancer doesn't allow you to run multiple copies of a task on the same instance. Instead, with the Classic Load Balancer, you must statically map port numbers on a container instance. However, an Application Load Balancer uses dynamic port mapping, so you can run multiple tasks from a single service on the same container instance.

via - https://aws.amazon.com/premiumsupport/knowledge-center/dynamic-port-mapping-ecs

Incorrect options:

Classic Load Balancer + Beanstalk - The Classic Load Balancer doesn't allow you to run multiple copies of a task on the same instance. Instead, with the Classic Load Balancer, you must statically map port numbers on a container instance. So this option is ruled out.

Application Load Balancer + Beanstalk - You can create docker environments that support multiple containers per Amazon EC2 instance with a multi-container Docker platform for Elastic Beanstalk. However, ECS gives you finer control.

Classic Load Balancer + ECS - The Classic Load Balancer doesn't allow you to run multiple copies of a task in the same instance. Instead, with the Classic Load Balancer, you must statically map port numbers on a container instance. So this option is ruled out.

References:

https://aws.amazon.com/premiumsupport/knowledge-center/dynamic-port-mapping-ecs

https://docs.aws.amazon.com/elasticloadbalancing/latest/application/tutorial-target-ecs-containers.html

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

Question 38: **Correct**

Other than the `Resources` section, which of the following sections in a Serverless Application Model (SAM) Template is mandatory?

- ○ **Mappings**
- ○ **Globals**
- ◉ **Transform** **(Correct)**
- ○ **Parameters**

Explanation

Correct option:

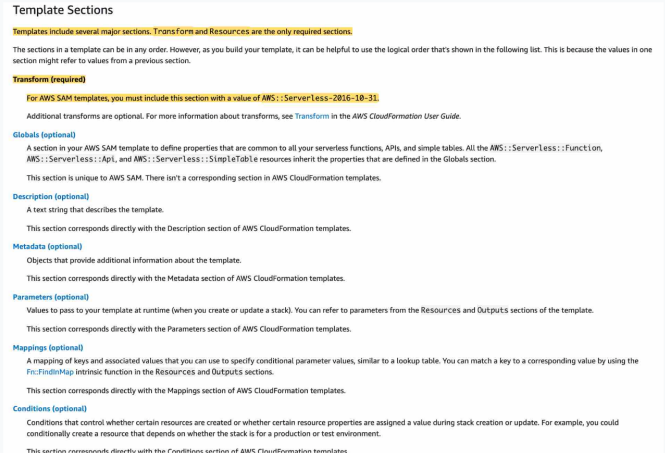
Transform

The AWS Serverless Application Model (AWS SAM) is an open-source framework that you can use to build serverless applications on AWS.

A serverless application is a combination of Lambda functions, event sources, and other resources that work together to perform tasks. Note that a serverless application is more than just a Lambda function—it can include additional resources such as APIs, databases, and event source mappings.

Serverless Application Model (SAM) Templates include several major sections. Transform and Resources are the only required sections.

Please review this note for more details:



Template Sections

Templates include several major sections. `Transform` and `Resources` are the only required sections.

The sections in a template can be in any order. However, as you build your template, it can be helpful to use the logical order that's shown in the following list. This is because the values in one section might refer to values from a previous section.

Transform (required)

For AWS SAM templates, you must include this section with a value of `AWS::Serverless-2016-10-31`.

Additional transforms are optional. For more information about transforms, see Transform in the *AWS CloudFormation User Guide*.

Globals (optional)

A section in your AWS SAM template to define properties that are common to all your serverless functions, APIs, and simple tables. All the `AWS::Serverless::Function`, `AWS::Serverless::Api`, and `AWS::Serverless::SimpleTable` resources inherit the properties that are defined in the Globals section.

This section is unique to AWS SAM. There isn't a corresponding section in AWS CloudFormation templates.

Description (optional)

A text string that describes the template.

This section corresponds directly with the Description section of AWS CloudFormation templates.

Metadata (optional)

Objects that provide additional information about the template.

This section corresponds directly with the Metadata section of AWS CloudFormation templates.

Parameters (optional)

Values to pass 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.

This section corresponds directly with the Parameters section of AWS CloudFormation templates.

Mappings (optional)

A mapping of keys and associated values that you can use to specify conditional parameter values, similar to a lookup table. You can match a key to a corresponding value by using the Fn::FindInMap intrinsic function in the `Resources` and `Outputs` sections.

This section corresponds directly with the Mappings section of AWS CloudFormation templates.

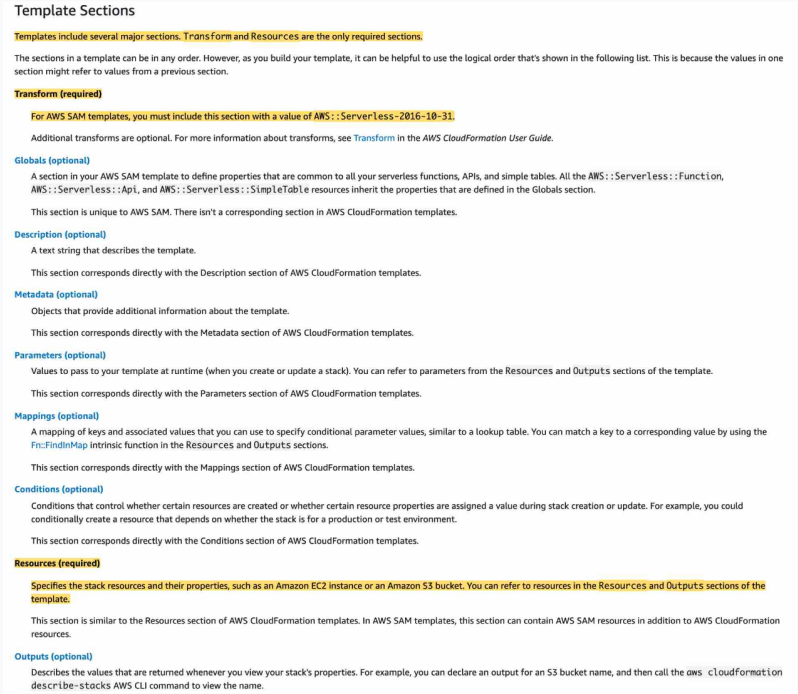
Conditions (optional)

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.

This section corresponds directly with the Conditions section of AWS CloudFormation templates.

Serverless Application Model (SAM) Templates include several major sections. Transform and Resources are the only required sections.

Please review this note for more details:



Template Sections

Templates include several major sections. `Transform` and `Resources` are the only required sections.

The sections in a template can be in any order. However, as you build your template, it can be helpful to use the logical order that's shown in the following list. This is because the values in one section might refer to values from a previous section.

Transform (required)

For AWS SAM templates, you must include this section with a value of `AWS::Serverless-2016-10-31`.

Additional transforms are optional. For more information about transforms, see Transform in the *AWS CloudFormation User Guide*.

Globals (optional)

A section in your AWS SAM template to define properties that are common to all your serverless functions, APIs, and simple tables. All the `AWS::Serverless::Function`, `AWS::Serverless::Api`, and `AWS::Serverless::SimpleTable` resources inherit the properties that are defined in the Globals section.

This section is unique to AWS SAM. There isn't a corresponding section in AWS CloudFormation templates.

Description (optional)

A text string that describes the template.

This section corresponds directly with the Description section of AWS CloudFormation templates.

Metadata (optional)

Objects that provide additional information about the template.

This section corresponds directly with the Metadata section of AWS CloudFormation templates.

Parameters (optional)

Values to pass 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.

This section corresponds directly with the Parameters section of AWS CloudFormation templates.

Mappings (optional)

A mapping of keys and associated values that you can use to specify conditional parameter values, similar to a lookup table. You can match a key to a corresponding value by using the Fn::FindInMap intrinsic function in the `Resources` and `Outputs` sections.

This section corresponds directly with the Mappings section of AWS CloudFormation templates.

Conditions (optional)

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.

This section corresponds directly with the Conditions section of AWS CloudFormation templates.

Resources (required)

Specifies the stack resources and their properties, such as an Amazon EC2 instance or an Amazon S3 bucket. You can refer to resources in the `Resources` and `Outputs` sections of the template.

This section is similar to the Resources section of AWS CloudFormation templates. In AWS SAM templates, this section can contain AWS SAM resources in addition to AWS CloudFormation resources.

Outputs (optional)

Describes the values that are returned whenever you view your stack's properties. For example, you can declare an output for an S3 bucket name, and then call the `aws cloudformation describe-stacks` AWS CLI command to view the name.

via - https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/sam-specification-template-anatomy.html

Incorrect options:

Parameters

Mappings

Globals

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

Reference:

https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/sam-specification-template-anatomy.html

Question 39: **Incorrect**

A developer wants to package the code and dependencies for the application-specific Lambda functions as container images to be hosted on Amazon Elastic Container Registry (ECR).

Which of the following options are correct for the given requirement? (Select two)

- [ ] **Lambda supports both Windows and Linux-based container images**
- [ ] **To deploy a container image to Lambda, the container image must implement the Lambda Runtime API** **(Correct)**
- [x] **You must create the Lambda function from the same account as the container registry in Amazon ECR** **(Correct)**
- [x] **You can test the containers locally using the Lambda Runtime API** **(Incorrect)**
- [ ] **You can deploy Lambda function as a container image, with a maximum size of 15 GB**

Explanation
Correct options:

To deploy a container image to Lambda, the container image must implement the Lambda Runtime API - To deploy a container image to Lambda, the container image must implement the Lambda Runtime API. The AWS open-source runtime interface clients implement the API. You can add a runtime interface client to your preferred base image to make it compatible with Lambda.

You must create the Lambda function from the same account as the container registry in Amazon ECR - You can package your Lambda function code and dependencies as a container image, using tools such as the Docker CLI. You can then upload the image to your container registry hosted on Amazon Elastic Container Registry (Amazon ECR). Note that you must create the Lambda function from the same account as the container registry in Amazon ECR.

Incorrect options:

Lambda supports both Windows and Linux-based container images - Lambda currently supports only Linux-based container images.

You can test the containers locally using the Lambda Runtime API - You can test the containers locally using the Lambda Runtime Interface Emulator.

You can deploy Lambda function as a container image, with a maximum size of 15 GB - You can deploy Lambda function as container image with the maximum size of 10GB.

Question 40: **Correct**

A developer in your company was just promoted to Team Lead and will be in charge of code deployment on EC2 instances via AWS CodeCommit and AWS CodeDeploy. Per the new requirements, the deployment process should be able to change permissions for deployed files as well as verify the deployment success.

Which of the following actions should the new Developer take?

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

Explanation

Correct option:

Define an `appspec.yml` file in the root directory: An AppSpec file must be a YAML-formatted file named appspec.yml and it must be placed in the root of the directory structure of an application's source code.

The AppSpec file is used to:

Map the source files in your application revision to their destinations on the instance.

Specify custom permissions for deployed files.

Specify scripts to be run on each instance at various stages of the deployment process.

During deployment, the CodeDeploy agent looks up the name of the current event in the hooks section of the AppSpec file. If the event is not found, the CodeDeploy agent moves on to the next step. If the event is found, the CodeDeploy agent retrieves the list of scripts to execute. The scripts are run sequentially, in the order in which they appear in the file. The status of each script is logged in the CodeDeploy agent log file on the instance.

If a script runs successfully, it returns an exit code of 0 (zero). If the CodeDeploy agent installed on the operating system doesn't match what's listed in the AppSpec file, the deployment fails.

Incorrect options:

Define a `buildspec.yml` file in the root directory - This is a file used by AWS CodeBuild to run a build. This is not relevant to the given use case.

Define a `buildspec.yml` file in the codebuild/ directory - This is a file used by AWS CodeBuild to run a build. This is not relevant to the given use case.

Define an `appspec.yml` file in the codebuild/ directory - This file is for AWS CodeDeploy and must be placed in the root of the directory structure of an application's source code

Question 41: **Correct**

You have launched several AWS Lambda functions written in Java. A new requirement was given that over 1MB of data should be passed to the functions and should be encrypted and decrypted at runtime.

Which of the following methods is suitable to address the given use-case?

- ◯ **Use KMS direct encryption and store as file**
- ◯ **Use KMS Encryption and store as environment variable**
- ◉ **Use Envelope Encryption and reference the data as file within the code** **(Correct)**
- ◯ **Use Envelope Encryption and store as environment variable**

Explanation

Correct option:

Use Envelope Encryption and reference the data as file within the code

While AWS KMS does support sending data up to 4 KB to be encrypted directly, envelope encryption can offer significant performance benefits. When you encrypt data directly with AWS KMS it must be transferred over the network. Envelope encryption reduces the network load since only the request and delivery of the much smaller data key go over the network. The data key is used locally in your application or encrypting AWS service, avoiding the need to send the entire block of data to AWS KMS and suffer network latency.

AWS Lambda environment variables can have a maximum size of 4 KB. Additionally, the direct 'Encrypt' API of KMS also has an upper limit of 4 KB for the data payload. To encrypt 1 MB, you need to use the Encryption SDK and pack the encrypted file with the lambda function.

Incorrect options:

Use KMS direct encryption and store as file - You can only encrypt up to 4 kilobytes (4096 bytes) of arbitrary data such as an RSA key, a database password, or other sensitive information, so this option is not correct for the given use-case.

Use Envelope Encryption and store as an environment variable - Environment variables must not exceed 4 KB, so this option is not correct for the given use-case.

Use KMS Encryption and store as an environment variable - You can encrypt up to 4 kilobytes (4096 bytes) of arbitrary data such as an RSA key, a database password, or other sensitive information. Lambda Environment variables must not exceed 4 KB. So this option is not correct for the given use-case.

References:

https://docs.aws.amazon.com/lambda/latest/dg/gettingstarted-limits.html

Question 42: **Correct**

You have created a continuous delivery service model with automated steps using AWS CodePipeline. Your pipeline uses your code, maintained in a CodeCommit repository, AWS CodeBuild, and AWS Elastic Beanstalk to automatically deploy your code every time there is a code change. However, the deployment to Elastic Beanstalk is taking a very long time due to resolving dependencies on all of your 100 target EC2 instances.

Which of the following actions should you take to improve performance with limited code changes?

- (●) **Bundle the dependencies in the source code during the build stage of CodeBuild** **(Correct)**
- ( ) **Create a custom platform for Elastic Beanstalk**
- ( ) **Bundle the dependencies in the source code in CodeCommit**
- ( ) **Store the dependencies in S3, to be used while deploying to Beanstalk**

Explanation

Correct option:

Bundle the dependencies in the source code during the build stage of CodeBuild

AWS CodeBuild is a fully managed build service. There are no servers to provision and scale, or software to install, configure, and operate.

A typical application build process includes phases like preparing the environment, updating the configuration, downloading dependencies, running unit tests, and finally, packaging the built artifact.

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

This will allow the code bundle to be deployed to Elastic Beanstalk to have both the dependencies and the code, hence speeding up the deployment time to Elastic Beanstalk

Incorrect options:

Bundle the dependencies in the source code in CodeCommit - This is not the best practice and could make the CodeCommit repository huge.

Store the dependencies in S3, to be used while deploying to Beanstalk - This option acts as a distractor. S3 can be used as a storage location for your source code, logs, and other artifacts that are created when you use Elastic Beanstalk. Dependencies are used during the process of building code, not while deploying to Beanstalk.

Create a custom platform for Elastic Beanstalk - This is a more advanced feature that requires code changes, so does not fit the use-case.

Question 43: **Correct**

A diagnostic lab stores its data on DynamoDB. The lab wants to backup a particular DynamoDB table data on Amazon S3, so it can download the S3 backup locally for some operational use.

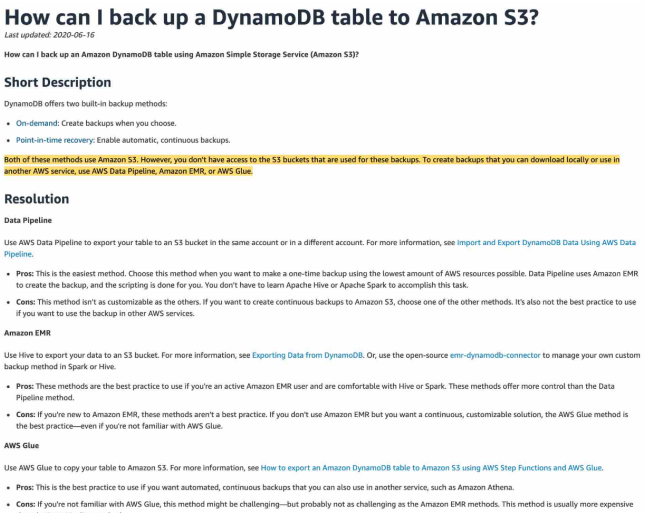
Which of the following options is NOT feasible?

- ◯ **Use Hive with Amazon EMR to export your data to an S3 bucket and download locally**
- ◯ **Use AWS Glue to copy your table to Amazon S3 and download locally**
- ◉ **Use the DynamoDB on-demand backup capability to write to Amazon S3 and download locally** **(Correct)**
- ◯ **Use AWS Data Pipeline to export your table to an S3 bucket in the account of your choice and download locally**

Explanation

Correct option:

Use the DynamoDB on-demand backup capability to write to Amazon S3 and download locally - This option is not feasible for the given use-case. DynamoDB has two built-in backup methods (On-demand, Point-in-time recovery) that write to Amazon S3, but you will not have access to the S3 buckets that are used for these backups.



How can I back up a DynamoDB table to Amazon S3?

Last updated: 2020-06-16

How can I back up an Amazon DynamoDB table using Amazon Simple Storage Service (Amazon S3)?

Short Description

DynamoDB offers two built-in backup methods:

- On-demand: Create backups when you choose.
- Point-in-time recovery: Enable automatic, continuous backups.

Both of these methods use Amazon S3. However, you don't have access to the S3 buckets that are used for these backups. To create backups that you can download locally or use in another AWS service, use AWS Data Pipeline, Amazon EMR, or AWS Glue.

Resolution

Data Pipeline

Use AWS Data Pipeline to export your table to an S3 bucket in the same account or in a different account. For more information, see Import and Export DynamoDB Data Using AWS Data Pipeline.

- **Pros:** This is the easiest method. Choose this method when you want to make a one-time backup using the lowest amount of AWS resources possible. Data Pipeline uses Amazon EMR to create the backup, and the scripting is done for you. You don't have to learn Apache Hive or Apache Spark to accomplish this task.
- **Cons:** This method isn't as customizable as the others. If you want to create continuous backups to Amazon S3, choose one of the other methods. It's also not the best practice to use if you want to use the backup in other AWS services.

Amazon EMR

Use Hive to export your data to an S3 bucket. For more information, see Exporting Data from DynamoDB. Or, use the open-source emr-dynamodb-connector to manage your own custom backup method in Spark or Hive.

- **Pros:** These methods are the best practice to use if you're an active Amazon EMR user and are comfortable with Hive or Spark. These methods offer more control than the Data Pipeline method.
- **Cons:** If you're new to Amazon EMR, these methods aren't a best practice. If you don't use Amazon EMR but you want a continuous, customizable solution, the AWS Glue method is the best practice—even if you're not familiar with AWS Glue.

AWS Glue

Use AWS Glue to copy your table to Amazon S3. For more information, see How to export an Amazon DynamoDB table to Amazon S3 using AWS Step Functions and AWS Glue.

- **Pros:** This is the best practice to use if you want automated, continuous backups that you can also use in another service, such as Amazon Athena.
- **Cons:** If you're not familiar with AWS Glue, this method might be challenging—but probably not as challenging as the Amazon EMR methods. This method is usually more expensive

via - https://aws.amazon.com/premiumsupport/knowledge-center/back-up-dynamodb-s3/

Incorrect options:

Use AWS Data Pipeline to export your table to an S3 bucket in the account of your choice and download locally - This is the easiest method. This method is used when you want to make a one-time backup using the lowest amount of AWS resources possible. Data Pipeline uses Amazon EMR to create the backup, and the scripting is done for you. You don't have to learn Apache Hive or Apache Spark to accomplish this task.

Use Hive with Amazon EMR to export your data to an S3 bucket and download locally - Use Hive to export data to an S3 bucket. Or, use the open-source emr-dynamodb-connector to manage your own custom backup method in Spark or Hive. These methods are the best practice to use if you're an active Amazon EMR user and are comfortable with Hive or Spark. These methods offer more control than the Data Pipeline method.

Use AWS Glue to copy your table to Amazon S3 and download locally - Use AWS Glue to copy your table to Amazon S3. This is the best practice to use if you want automated, continuous backups that you can also use in another service, such as Amazon Athena.

References:

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

https://docs.aws.amazon.com/datapipeline/latest/DeveloperGuide/dp-importexport-ddb-part1.html

https://docs.aws.amazon.com/emr/latest/ReleaseGuide/EMR_Hive_Commands.html#EMR_Hive_Commands_exporting

https://aws.amazon.com/blogs/big-data/how-to-export-an-amazon-dynamodb-table-to-amazon-s3-using-aws-step-functions-and-aws-glue/

Question 44: **Correct**

An e-commerce company manages a microservices application that receives orders from various partners through a customized API for each partner exposed via Amazon API Gateway. The orders are processed by a shared Lambda function.

How can the company notify each partner regarding the status of their respective orders in the most efficient manner, without affecting other partners' orders? Also, the solution should be scalable to accommodate new partners with minimal code changes required.

- ○ **Set up a separate SNS topic for each partner. Modify the Lambda function to publish messages for each partner to the partner's SNS topic**
- ○ **Set up a separate Lambda function for each partner. Set up an SNS topic and subscribe each partner to the SNS topic. Modify each partner's Lambda function to publish messages with specific attributes to the SNS topic and apply the appropriate filter policy to the topic subscriptions**
- ○ **Set up a separate SNS topic for each partner and subscribe each partner to the respective SNS topic. Modify the Lambda function to publish messages with specific attributes to the partner's SNS topic and apply the appropriate filter policy to the topic subscriptions**
- ◉ **Set up an SNS topic and subscribe each partner to the SNS topic. Modify the Lambda function to publish messages with specific attributes to the SNS topic and apply the appropriate filter policy to the topic subscriptions** (Correct)

Explanation
Correct option:

Set up an SNS topic and subscribe each partner to the SNS topic. Modify the Lambda function to publish messages with specific attributes to the SNS topic and apply the appropriate filter policy to the topic subscriptions

An Amazon SNS topic is a logical access point that acts as a communication channel. A topic lets you group multiple endpoints (such as AWS Lambda, Amazon SQS, HTTP/S, or an email address). For example, to broadcast the messages of a message-producer system (such as, an e-commerce website) working with multiple other services that require its messages (for example, checkout and fulfillment systems), you can create a topic for your producer system.

By default, an Amazon SNS topic subscriber receives every message that's published to the topic. To receive only a subset of the messages, a subscriber must assign a filter policy to the topic subscription. A filter policy is a JSON object containing properties that define which messages the subscriber receives. Amazon SNS supports policies that act on the message attributes or the message body, according to the filter policy scope that you set for the subscription. Filter policies for the message body assume that the message payload is a well-formed JSON object.

For the given use case, you can change the Lambda function to publish messages with specific attributes to the single SNS topic and apply the appropriate filter policy to the topic subscriptions for each of the partners. This is also easily scalable for new partners since only the filter policy needs to be set up for the new partner.

Incorrect options:

Set up a separate SNS topic for each partner. Modify the Lambda function to publish messages for each partner to the partner's SNS topic

Set up a separate SNS topic for each partner and subscribe each partner to the respective SNS topic. Modify the Lambda function to publish messages with specific attributes to the partner's SNS topic and apply the appropriate filter policy to the topic subscriptions

Both of these options represent an inefficient solution as there is no need to segregate each partner's updates into a separate SNS topic. A single SNS topic with distinct filter policies is sufficient.

Set up a separate Lambda function for each partner. Set up an SNS topic and subscribe each partner to the SNS topic. Modify each partner's Lambda function to publish messages with specific attributes to the SNS topic and apply the appropriate filter policy to the topic subscriptions - This is again an inefficient solution as there is no need to create a separate Lambda function for each partner as just a shared Lambda function is sufficient to process the orders and send an update to the single SNS topic with distinct filter policies.

References:

https://docs.aws.amazon.com/sns/latest/dg/sns-create-topic.html

https://docs.aws.amazon.com/sns/latest/dg/sns-message-filtering.html

Question 45: **Correct**

As a Senior Developer, you are tasked with creating several API Gateway powered APIs along with your team of developers. The developers are working on the API in the development environment, but they find the changes made to the APIs are not reflected when the API is called.

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

- ○ **Use Stage Variables for development state of API**
- ○ **Developers need IAM permissions on API execution component of API Gateway**
- ◉ **Redeploy the API to an existing stage or to a new stage** **(Correct)**
- ○ **Enable Lambda authorizer to access API**

Explanation
Correct option:

Redeploy the API to an existing stage or to a new stage

After creating your API, you must deploy it to make it callable by your users. To deploy an API, you create an API deployment and associate it with a stage. A stage is a logical reference to a lifecycle state of your API (for example, dev, prod, beta, v2). API stages are identified by the API ID and stage name. Every time you update an API, you must redeploy the API to an existing stage or to a new stage. Updating an API includes modifying routes, methods, integrations, authorizers, and anything else other than stage settings.

Incorrect options:

Developers need IAM permissions on API execution component of API Gateway - Access control access to Amazon API Gateway APIs is done with IAM permissions. To call a deployed API or to refresh the API caching, you must grant the API caller permissions to perform required IAM actions supported by the API execution component of API Gateway. In the current scenario, developers do not need permissions on "execution components" but on "management components" of API Gateway that help them to create, deploy, and manage an API. Hence, this statement is an incorrect option.

Enable Lambda authorizer to access API - 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. So, this feature too helps in access control, but in the current scenario its the developers and not the users who are facing the issue. So, this statement is an incorrect option.

Use Stage Variables for development state of API - 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

Question 46: **Incorrect**

A developer is defining the signers that can create signed URLs for their Amazon CloudFront distributions.

Which of the following statements should the developer consider while defining the signers? (Select two)

- [ ] **When you use the root user to manage CloudFront key pairs, you can only have up to two active CloudFront key pairs per AWS account** (Correct)
- [x] **You can also use AWS Identity and Access Management (IAM) permissions policies to restrict what the root user can do with CloudFront key pairs** (Incorrect)
- [x] **CloudFront key pairs can be created with any account that has administrative permissions and full access to CloudFront resources** (Incorrect)
- [ ] **Both the signers (trusted key groups and CloudFront key pairs) can be managed using the CloudFront APIs**
- [ ] **When you create a signer, the public key is with CloudFront and private key is used to sign a portion of URL** (Correct)

Explanation

Correct options:

When you create a signer, the public key is with CloudFront and private key is used to sign a portion of URL - Each signer that you use to create CloudFront signed URLs or signed cookies must have a public–private key pair. The signer uses its private key to sign the URL or cookies, and CloudFront uses the public key to verify the signature.

When you create signed URLs or signed cookies, you use the private key from the signer's key pair to sign a portion of the URL or the cookie. When someone requests a restricted file, CloudFront compares the signature in the URL or cookie with the unsigned URL or cookie, to verify that it hasn't been tampered with. CloudFront also verifies that the URL or cookie is valid, meaning, for example, that the expiration date and time haven't passed.

When you use the root user to manage CloudFront key pairs, you can only have up to two active CloudFront key pairs per AWS account - When you use the root user to manage CloudFront key pairs, you can only have up to two active CloudFront key pairs per AWS account.

Whereas, with CloudFront key groups, you can associate a higher number of public keys with your CloudFront distribution, giving you more flexibility in how you use and manage the public keys. By default, you can associate up to four key groups with a single distribution, and you can have up to five public keys in a key group.

Incorrect options:

You can also use AWS Identity and Access Management (IAM) permissions policies to restrict what the root user can do with CloudFront key pairs - When you use the AWS account root user to manage CloudFront key pairs, you can't restrict what the root user can do or the conditions in which it can do them. You can't apply IAM permissions policies to the root user, which is one reason why AWS best practices recommend against using the root user.

CloudFront key pairs can be created with any account that has administrative permissions and full access to CloudFront resources - CloudFront key pairs can only be created using the root user account and hence is not a best practice to create CloudFront key pairs as signers.

Both the signers (trusted key groups and CloudFront key pairs) can be managed using the CloudFront APIs - With CloudFront key groups, you can manage public keys, key groups, and trusted signers using the CloudFront API. You can use the API to automate key creation and key rotation. When you use the AWS root user, you have to use the AWS Management Console to manage CloudFront key pairs, so you can't automate the process.

Reference:

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

Question 47: **Incorrect**

A developer needs to automate software package deployment to both Amazon EC2 instances and virtual servers running on-premises, as part of continuous integration and delivery that the business has adopted.

Which AWS service should he use to accomplish this task?

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

Explanation

Correct option:

Continuous integration is a DevOps software development practice where developers regularly merge their code changes into a central repository, after which automated builds and tests are run.

Continuous delivery is a software development practice where code changes are automatically prepared for a release to production. A pillar of modern application development, continuous delivery expands upon continuous integration by deploying all code changes to a testing environment and/or a production environment after the build stage.

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. This is the right choice for the current use case.

Incorrect options:

AWS CodePipeline - AWS CodePipeline is a fully managed "continuous delivery" service that helps you automate your release pipelines for fast and reliable application and infrastructure updates. CodePipeline automates the build, test, and deploy phases of your release process every time there is a code change, based on the release model you define. This enables you to rapidly and reliably deliver features and updates. Whereas CodeDeploy is a deployment service, CodePipeline is a continuous delivery service. For our current scenario, CodeDeploy is the correct choice.

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

your builds are not left waiting in a queue.

AWS Elastic Beanstalk - AWS Elastic Beanstalk is an easy-to-use service for deploying and scaling web applications and services developed with Java, .NET, PHP, Node.js, Python, Ruby, Go, and Docker on familiar servers such as Apache, Nginx, Passenger, and IIS. You can simply upload your code and Elastic Beanstalk automatically handles the deployment, from capacity provisioning, load balancing, auto-scaling to application health monitoring. At the same time, you retain full control over the AWS resources powering your application and can access the underlying resources at any time.

References:

https://aws.amazon.com/codedeploy/

https://aws.amazon.com/codepipeline/

https://aws.amazon.com/codebuild/

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

https://aws.amazon.com/devops/continuous-delivery/

https://aws.amazon.com/devops/continuous-integration/

Question 48: **Correct**

A pharmaceutical company runs their database workloads on Provisioned IOPS SSD (io1) volumes.

As a Developer Associate, which of the following options would you identify as an **INVALID** configuration for io1 EBS volume types?

- ○ **200 GiB size volume with 10000 IOPS**
- ○ **200 GiB size volume with 2000 IOPS**
- ○ **200 GiB size volume with 5000 IOPS**
- ◉ **200 GiB size volume with 15000 IOPS** **(Correct)**

Explanation

Correct option:

200 GiB size volume with 15000 IOPS - This is an invalid configuration. The maximum ratio of provisioned IOPS to requested volume size (in GiB) is 50:1. So, for a 200 GiB volume size, max IOPS possible is 200*50 = 10000 IOPS.

Overview of Provisioned IOPS SSD (io1) volumes:

Overview of Provisioned IOPS SSD (io1) volumes:

Provisioned IOPS SSD (io1) volumes

Provisioned IOPS SSD (io1) volumes are designed to meet the needs of I/O-intensive workloads, particularly database workloads, that are sensitive to storage performance and consistency. Unlike gp2, which uses a bucket and credit model to calculate performance, an io1 volume allows you to specify a consistent IOPS rate when you create the volume, and Amazon EBS delivers the provisioned performance 99.9 percent of the time.

An io1 volume can range in size from 4 GiB to 16 TiB. You can provision from 100 IOPS up to 64,000 IOPS per volume on Instances built on the Nitro System and up to 32,000 on other instances. The maximum ratio of provisioned IOPS to requested volume size (in GiB) is 50:1. For example, a 100 GiB volume can be provisioned with up to 5,000 IOPS. On a supported instance type, any volume 1,280 GiB in size or greater allows provisioning up to the 64,000 IOPS maximum (50 × 1,280 GiB = 64,000).

An io1 volume provisioned with up to 32,000 IOPS supports a maximum I/O size of 256 KiB and yields as much as 500 MiB/s of throughput. With the I/O size at the maximum, peak throughput is reached at 2,000 IOPS. A volume provisioned with more than 32,000 IOPS (up to the cap of 64,000 IOPS) supports a maximum I/O size of 16 KiB and yields as much as 1,000 MiB/s of throughput. The following graph illustrates these performance characteristics:

via - https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html

Incorrect options:

Provisioned IOPS SSD (io1) volumes allow you to specify a consistent IOPS rate when you create the volume, and Amazon EBS delivers the provisioned performance 99.9 percent of the time. An io1 volume can range in size from 4 GiB to 16 TiB. The maximum ratio of provisioned IOPS to the requested volume size (in GiB) is 50:1. For example, a 100 GiB volume can be provisioned with up to 5,000 IOPS.

200 GiB size volume with 2000 IOPS - As explained above, up to 10000 IOPS is a valid configuration for the given use-case.

200 GiB size volume with 10000 IOPS - As explained above, up to 10000 IOPS is a valid configuration for the given use-case.

200 GiB size volume with 5000 IOPS - As explained above, up to 10000 IOPS is a valid configuration for the given use-case.

Reference:

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-volume-types.html

Question 49: **Correct**

A development team is working on an AWS Lambda function that accesses DynamoDB. The Lambda function must do an upsert, that is, it must retrieve an item and update some of its attributes or create the item if it does not exist.

Which of the following represents the solution with MINIMUM IAM permissions that can be used for the Lambda function to achieve this functionality?

- ○ **dynamodb:GetRecords, dynamodb:PutItem, dynamodb:UpdateTable**
- ○ **dynamodb:AddItem, dynamodb:GetItem**
- ◉ **dynamodb:UpdateItem, dynamodb:GetItem** **(Correct)**
- ○ **dynamodb:UpdateItem, dynamodb:GetItem, dynamodb:PutItem**

Explanation
Correct option:

dynamodb:UpdateItem, dynamodb:GetItem - With Amazon DynamoDB transactions, you can group multiple actions together and submit them as a single all-or-nothing TransactWriteItems or TransactGetItems operation.

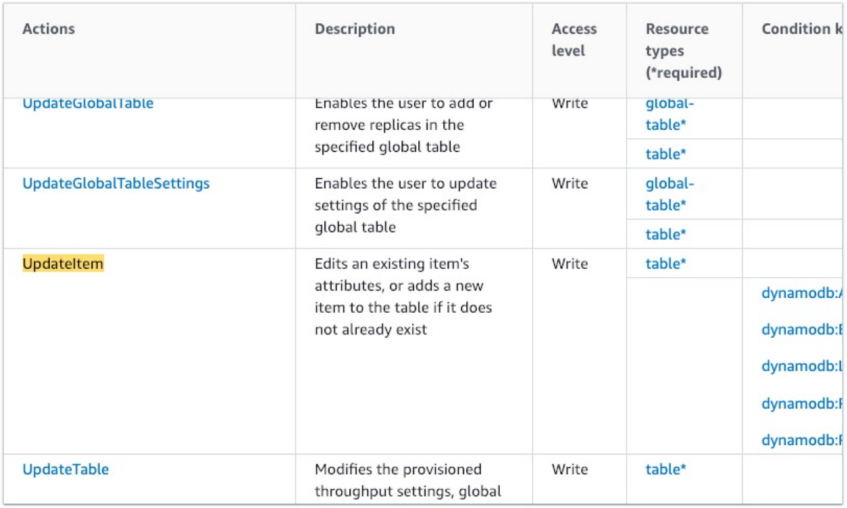
You can use AWS Identity and Access Management (IAM) to restrict the actions that transactional operations can perform in Amazon DynamoDB. Permissions for Put, Update, Delete, and Get actions are governed by the permissions used for the underlying PutItem, UpdateItem, DeleteItem, and GetItem operations. For the ConditionCheck action, you can use the `dynamodb:ConditionCheck` permission in IAM policies.

`UpdateItem` action of DynamoDB APIs, edits an existing item's attributes or adds a new item to the table if it does not already exist. You can put, delete, or add attribute values. You can also perform a conditional update on an existing item (insert a new attribute name-value pair if it doesn't exist, or replace an existing name-value pair if it has certain expected attribute values).

There is no need to inlcude the `dynamodb:PutItem` action for the given use-case.

So, the IAM policy must include permissions to get and update the item in the DynamoDB table.

Actions defined by DynamoDB:



Actions	Description	Access level	Resource types (*required)	Condition k
UpdateGlobalTable	Enables the user to add or remove replicas in the specified global table	Write	global-table*	
			table*	
UpdateGlobalTableSettings	Enables the user to update settings of the specified global table	Write	global-table*	
			table*	
UpdateItem	Edits an existing item's attributes, or adds a new item to the table if it does not already exist	Write	table*	
				dynamodb:A
				dynamodb:B
				dynamodb:D
				dynamodb:F
				dynamodb:F
UpdateTable	Modifies the provisioned throughput settings, global	Write	table*	

via - https://docs.aws.amazon.com/service-authorization/latest/reference/list_amazondynamodb.html

Incorrect options:

dynamodb:AddItem, dynamodb:GetItem

dynamodb:GetRecords, dynamodb:PutItem, dynamodb:UpdateTable

dynamodb:UpdateItem, dynamodb:GetItem, dynamodb:PutItem

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

References:

https://docs.aws.amazon.com/service-authorization/latest/reference/list_amazondynamodb.html

https://docs.amazonaws.cn/en_us/amazondynamodb/latest/developerguide/transaction-apis-iam.html

https://docs.aws.amazon.com/service-authorization/latest/reference/list_amazondynamodb.html

Question 50: **Correct**

An application running on EC2 instances processes messages from an SQS queue. However, sometimes the messages are not processed and they end up in errors. These messages need to be isolated for further processing and troubleshooting.

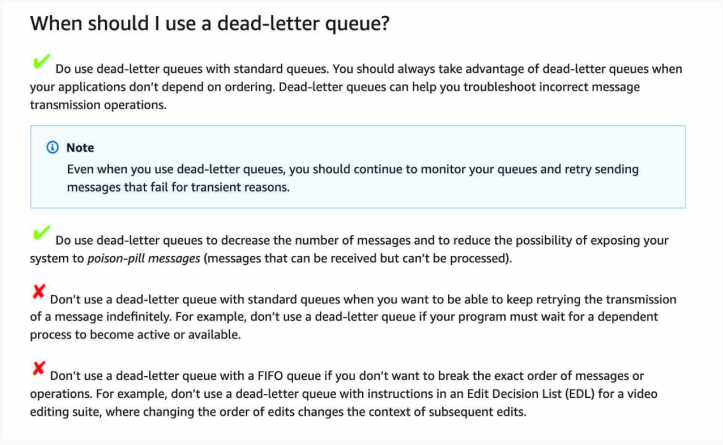
Which of the following options will help achieve this?

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

Explanation

Correct option:

Implement a Dead-Letter Queue - Amazon SQS supports dead-letter queues, which other queues (source queues) can target for messages that can't be processed (consumed) successfully. Dead-letter queues are useful for debugging your application or messaging system because they let you isolate problematic messages to determine why their processing doesn't succeed. Amazon SQS does not create the dead-letter queue automatically. You must first create the queue before using it as a dead-letter queue.



When should I use a dead-letter queue?

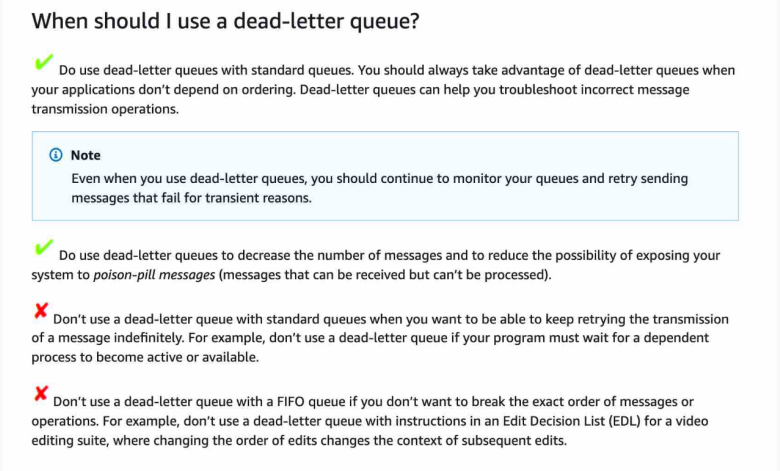
✔ Do use dead-letter queues with standard queues. You should always take advantage of dead-letter queues when your applications don't depend on ordering. Dead-letter queues can help you troubleshoot incorrect message transmission operations.

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

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

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

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



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

Incorrect options:

Increase the VisibilityTimeout - When a consumer receives and processes a message from a queue, the message remains in the queue. Amazon SQS doesn't automatically delete the message. Immediately after a message is received, it remains in the queue. To prevent other consumers from processing the message again, Amazon SQS sets a visibility timeout, a period of time during which Amazon SQS prevents other consumers from receiving and processing the message. Increasing visibility timeout will not help in troubleshooting the messages running into error or isolating them from the rest. Hence this is an incorrect option for the current use case.

Use DeleteMessage - Deletes the specified message from the specified queue. This will not help understand the reason for error or isolate messages ending with the error.

Reduce the VisibilityTimeout - As explained above, VisibilityTimeout makes sure that the message is not read by any other consumer while it is being processed by one consumer. By reducing the VisibilityTimeout, more consumers will receive the same failed message. Hence, this is an incorrect option for this use case.

References:

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

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

Question 51: **Correct**

The technology team at an investment bank uses DynamoDB to facilitate high-frequency trading where multiple trades can try and update an item at the same time.

Which of the following actions would make sure that only the last updated value of any item is used in the application?

- ○ **Use ConsistentRead = false while doing PutItem operation for any item**
- ○ **Use ConsistentRead = true while doing PutItem operation for any item**
- ◉ **Use ConsistentRead = true while doing GetItem operation for any item** **(Correct)**
- ○ **Use ConsistentRead = true while doing UpdateItem operation for any item**

Explanation
Correct option:

Use ConsistentRead = true while doing GetItem operation for any item

DynamoDB supports eventually consistent and strongly consistent reads.

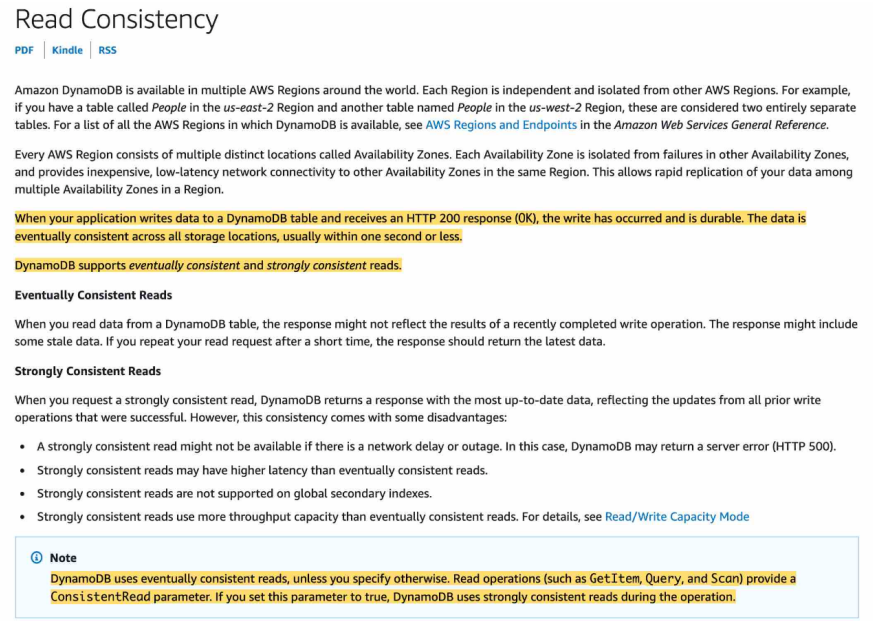
Eventually Consistent Reads

When you read data from a DynamoDB table, the response might not reflect the results of a recently completed write operation. The response might include some stale data. If you repeat your read request after a short time, the response should return the latest data.

Strongly Consistent Reads

When you request a strongly consistent read, DynamoDB returns a response with the most up-to-date data, reflecting the updates from all prior write operations that were successful.

DynamoDB uses eventually consistent reads by default. Read operations (such as GetItem, Query, and Scan) provide a ConsistentRead parameter. If you set this parameter to true, DynamoDB uses strongly consistent reads during the operation. As per the given use-case, to make sure that only the last updated value of any item is used in the application, you should use strongly consistent reads by setting ConsistentRead = true for GetItem operation.



Read Consistency

PDF | Kindle | RSS

Amazon DynamoDB is available in multiple AWS Regions around the world. Each Region is independent and isolated from other AWS Regions. For example, if you have a table called *People* in the *us-east-2* Region and another table named *People* in the *us-west-2* Region, these are considered two entirely separate tables. For a list of all the AWS Regions in which DynamoDB is available, see AWS Regions and Endpoints in the *Amazon Web Services General Reference*.

Every AWS Region consists of multiple distinct locations called Availability Zones. Each Availability Zone is isolated from failures in other Availability Zones, and provides inexpensive, low-latency network connectivity to other Availability Zones in the same Region. This allows rapid replication of your data among multiple Availability Zones in a Region.

When your application writes data to a DynamoDB table and receives an HTTP 200 response (OK), the write has occurred and is durable. The data is eventually consistent across all storage locations, usually within one second or less.

DynamoDB supports *eventually consistent* and *strongly consistent* reads.

Eventually Consistent Reads

When you read data from a DynamoDB table, the response might not reflect the results of a recently completed write operation. The response might include some stale data. If you repeat your read request after a short time, the response should return the latest data.

Strongly Consistent Reads

When you request a strongly consistent read, DynamoDB returns a response with the most up-to-date data, reflecting the updates from all prior write operations that were successful. However, this consistency comes with some disadvantages:

- A strongly consistent read might not be available if there is a network delay or outage. In this case, DynamoDB may return a server error (HTTP 500).
- Strongly consistent reads may have higher latency than eventually consistent reads.
- Strongly consistent reads are not supported on global secondary indexes.
- Strongly consistent reads use more throughput capacity than eventually consistent reads. For details, see Read/Write Capacity Mode

Note

DynamoDB uses eventually consistent reads, unless you specify otherwise. Read operations (such as `GetItem`, `Query`, and `Scan`) provide a `ConsistentRead` parameter. If you set this parameter to true, DynamoDB uses strongly consistent reads during the operation.

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

Incorrect options:

Use ConsistentRead = true while doing UpdateItem operation for any item

Use ConsistentRead = true while doing PutItem operation for any item

Use ConsistentRead = false while doing PutItem operation for any item

As mentioned in the explanation above, strongly consistent reads apply only while using the read operations (such as GetItem, Query, and Scan). So these three options are incorrect.

Reference:

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

Question 52: **Correct**

A Developer is configuring Amazon EC2 Auto Scaling group to scale dynamically.

Which metric below is NOT part of Target Tracking Scaling Policy?

- (●) **ApproximateNumberOfMessagesVisible** **(Correct)**
- ( ) **ASGAverageNetworkOut**
- ( ) **ASGAverageCPUUtilization**
- ( ) **ALBRequestCountPerTarget**

Explanation

Correct option:

ApproximateNumberOfMessagesVisible - This is a CloudWatch Amazon SQS queue metric. The number of messages in a queue might not change proportionally to the size of the Auto Scaling group that processes messages from the queue. Hence, this metric does not work for target tracking.

Incorrect options:

With target tracking scaling policies, you select a scaling metric and set a target value. Amazon EC2 Auto Scaling creates and manages the CloudWatch alarms that trigger the scaling policy and calculates the scaling adjustment based on the metric and the target value.

It is important to note that a target tracking scaling policy assumes that it should scale out your Auto Scaling group when the specified metric is above the target value. You cannot use a target tracking scaling policy to scale out your Auto Scaling group when the specified metric is below the target value.

ASGAverageCPUUtilization - This is a predefined metric for target tracking scaling policy. This represents the Average CPU utilization of the Auto Scaling group.

ASGAverageNetworkOut - This is a predefined metric for target tracking scaling policy. This represents the Average number of bytes sent out on all network interfaces by the Auto Scaling group.

ALBRequestCountPerTarget - This is a predefined metric for target tracking scaling policy. This represents the Number of requests completed per target in an Application Load Balancer target group.

Reference:

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

Question 53: **Correct**

As a Team Lead, you are expected to generate a report of the code builds for every week to report internally and to the client. This report consists of the number of code builds performed for a week, the percentage success and failure, and overall time spent on these builds by the team members. You also need to retrieve the CodeBuild logs for failed builds and analyze them in Athena.

Which of the following options will help achieve this?

- ○ **Use AWS CloudTrail and deliver logs to S3**
- ○ **Use AWS Lambda integration**
- ◉ **Enable S3 and CloudWatch Logs integration** **(Correct)**
- ○ **Use CloudWatch Events**

Explanation

Correct option:

Enable S3 and CloudWatch Logs integration - AWS CodeBuild monitors functions on your behalf and reports metrics through Amazon CloudWatch. These metrics include the number of total builds, failed builds, successful builds, and the duration of builds. You can monitor your builds at two levels: Project level, AWS account level. You can export log data from your log groups to an Amazon S3 bucket and use this data in custom processing and analysis, or to load onto other systems.

Incorrect options:

Use CloudWatch Events - You can integrate CloudWatch Events with CodeBuild. However, we are looking at storing and running queries on logs, so Cloudwatch logs with S3 integration makes sense for this context.o

Use AWS Lambda integration - Lambda is a good choice to use boto3 library to read logs programmatically. But, CloudWatch and S3 integration is already built-in and is an optimized way of managing the given use-case.

Use AWS CloudTrail and deliver logs to S3 - AWS CodeBuild is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in CodeBuild. CloudTrail captures all API calls for CodeBuild as events, including calls from the CodeBuild console and from code calls to the CodeBuild APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an S3 bucket, including events for CodeBuild. This is an important feature for monitoring a service but isn't a good fit for the current scenario.

References:

https://docs.aws.amazon.com/codebuild/latest/userguide/monitoring-metrics.html

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

https://docs.aws.amazon.com/codebuild/latest/userguide/getting-started-input-

Question 54: **Correct**

An e-commerce company uses AWS CloudFormation to implement Infrastructure as Code for the entire organization. Maintaining resources as stacks with CloudFormation has greatly reduced the management effort needed to manage and maintain the resources. However, a few teams have been complaining of failing stack updates owing to out-of-band fixes running on the stack resources.

Which of the following is the best solution that can help in keeping the CloudFormation stack and its resources in sync with each other?

- ○ **Use Tag feature of CloudFormation to monitor the changes happening on specific resources**
- ○ **Use CloudFormation in Elastic Beanstalk environment to reduce direct changes to CloudFormation resources**
- ○ **Use Change Sets feature of CloudFormation**
- ◉ **Use Drift Detection feature of CloudFormation** **(Correct)**

Explanation
Correct option:

Use Drift Detection feature of CloudFormation

Drift detection enables you to detect whether a stack's actual configuration differs, or has drifted, from its expected configuration. Use CloudFormation to detect drift on an entire stack, or individual resources within the stack. A resource is considered to have drifted if any of its actual property values differ from the expected property values. This includes if the property or resource has been deleted. A stack is considered to have drifted if one or more of its resources have drifted.

To determine whether a resource has drifted, CloudFormation determines the expected resource property values, as defined in the stack template and any values specified as template parameters. CloudFormation then compares those expected values with the actual values of those resource properties as they currently exist in the stack. A resource is considered to have drifted if one or more of its properties have been deleted, or had their value changed.

You can then take corrective action so that your stack resources are again in sync with their definitions in the stack template, such as updating the drifted resources directly so that they agree with their template definition. Resolving drift helps to ensure configuration consistency and successful stack operations.

Incorrect options:

Use CloudFormation in Elastic Beanstalk environment to reduce direct changes to CloudFormation resources - Elastic Beanstalk environment provides full access to the resources created. So, it is possible to edit the resources and hence does not solve the issue mentioned for the given use case.

Use Tag feature of CloudFormation to monitor the changes happening on specific resources - Tags help you identify and categorize the resources created as part of CloudFormation template. This feature is not helpful for the given use case.

Use Change Sets feature of CloudFormation - When you need to update a stack, understanding how your changes will affect running resources before you implement them can help you update stacks with confidence. Change sets allow you to preview how proposed changes to a stack might impact your running resources, for example, whether your changes will delete or replace any critical resources, AWS CloudFormation makes the changes to your stack only when you decide to execute the change set, allowing you to decide whether to proceed with your proposed changes or explore other changes by creating another change set. Change sets are not useful for the given use-case.

References:

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/detect-drift-stack.html

https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-cfn-updating-stacks-changesets.html

Question 55: **Correct**

An Auto Scaling group has a maximum capacity of 3, a current capacity of 2, and a scaling policy that adds 3 instances.

When executing this scaling policy, what is the expected outcome?

- (●) **Amazon EC2 Auto Scaling adds only 1 instance to the group** **(Correct)**
- ( ) **Amazon EC2 Auto Scaling adds 3 instances to the group**
- ( ) **Amazon EC2 Auto Scaling does not add any instances to the group, but suggests changing the scaling policy to add one instance**
- ( ) **Amazon EC2 Auto Scaling adds 3 instances to the group and scales down 2 of those instances eventually**

Explanation

Correct option:

A scaling policy instructs Amazon EC2 Auto Scaling to track a specific CloudWatch metric, and it defines what action to take when the associated CloudWatch alarm is in ALARM.

When a scaling policy is executed, if the capacity calculation produces a number outside of the minimum and maximum size range of the group, Amazon EC2 Auto Scaling ensures that the new capacity never goes outside of the minimum and maximum size limits.

How Scaling Policies Work

A scaling policy instructs Amazon EC2 Auto Scaling to track a specific CloudWatch metric, and it defines what action to take when the associated CloudWatch alarm is in ALARM. The metrics that are used to trigger an alarm are an aggregation of metrics coming from all of the instances in the Auto Scaling group. (For example, let's say you have an Auto Scaling group with two instances where one instance is at 60 percent CPU and the other is at 40 percent CPU. On average, they are at 50 percent CPU.) When the policy is in effect, Amazon EC2 Auto Scaling adjusts the group's desired capacity up or down when the alarm is triggered.

When a scaling policy is executed, if the capacity calculation produces a number outside of the minimum and maximum size range of the group, Amazon EC2 Auto Scaling ensures that the new capacity never goes outside of the minimum and maximum size limits. Capacity is measured in one of two ways: using the same units that you chose when you set the desired capacity in terms of instances, or using capacity units (if instance weighting is applied).

- Example 1: An Auto Scaling group has a maximum capacity of 3, a current capacity of 2, and a scaling policy that adds 3 instances. When executing this scaling policy, Amazon EC2 Auto Scaling adds only 1 instance to the group to prevent the group from exceeding its maximum size.
- Example 2: An Auto Scaling group has a minimum capacity of 2, a current capacity of 3, and a scaling policy that removes 2 instances. When executing this scaling policy, Amazon EC2 Auto Scaling removes only 1 instance from the group to prevent the group from becoming less than its minimum size.

to prevent the group from exceeding its maximum size.

- Example 2: An Auto Scaling group has a minimum capacity of 2, a current capacity of 3, and a scaling policy that removes 2 instances. When executing this scaling policy, Amazon EC2 Auto Scaling removes only 1 instance from the group to prevent the group from becoming less than its minimum size.

When the desired capacity reaches the maximum size limit, scaling out stops. If demand drops and capacity decreases, Amazon EC2 Auto Scaling can scale out again.

The exception is when you use instance weighting. In this case, Amazon EC2 Auto Scaling can scale out above the maximum size limit, but only by up to your maximum instance weight. Its intention is to get as close to the new desired capacity as possible but still adhere to the allocation strategies that are specified for the group. The allocation strategies determine which instance types to launch. The weights determine how many capacity units each instance contributes to the desired capacity of the group based on its instance type.

- Example 3: An Auto Scaling group has a maximum capacity of 12, a current capacity of 10, and a scaling policy that adds 5 capacity units. Instance types have one of three weights assigned: 1, 4, or 6. When executing the scaling policy, Amazon EC2 Auto Scaling chooses to launch an instance type with a weight of 6 based on the allocation strategy. The result of this scale-out event is a group with a desired capacity of 12 and a current capacity of 16.

via - https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scale-based-on-demand.html

Amazon EC2 Auto Scaling adds only 1 instance to the group

For the given use-case, Amazon EC2 Auto Scaling adds only 1 instance to the group to prevent the group from exceeding its maximum size.

Incorrect options:

Amazon EC2 Auto Scaling adds 3 instances to the group - This is an incorrect statement. Auto Scaling ensures that the new capacity never goes outside of the minimum and maximum size limits.

Amazon EC2 Auto Scaling adds 3 instances to the group and scales down 2 of those instances eventually - This is an incorrect statement. Adding the instances initially and immediately downsizing them is impractical.

Amazon EC2 Auto Scaling does not add any instances to the group, but suggests changing the scaling policy to add one instance - This option has been added as a distractor.

Reference:

https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scale-based-on-demand.html

Question 56: **Correct**

A company has a cloud system in AWS with components that send and receive messages using SQS queues. While reviewing the system you see that it processes a lot of information and would like to be aware of any limits of the system.

Which of the following represents the maximum number of messages that can be stored in an SQS queue?

- ○ **10000**
- ○ **100000**
- ○ **10000000**
- ◉ **no limit** **(Correct)**

Explanation
Correct option:

"no limit": There are no message limits for storing in SQS, but 'in-flight messages' do have limits. Make sure to delete messages after you have processed them. There can be a maximum of approximately 120,000 inflight messages (received from a queue by a consumer, but not yet deleted from the queue).

Incorrect options:

"10000"

"100000"

"10000000"

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

Reference:

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

Question 57: **Incorrect**

A developer with access to the AWS Management Console terminated an instance in the us-east-1a availability zone. The attached EBS volume remained and is now available for attachment to other instances. Your colleague launches a new Linux EC2 instance in the us-east-1e availability zone and is attempting to attach the EBS volume. Your colleague informs you that it is not possible and need your help.

Which of the following explanations would you provide to them?

- (●) **EBS volumes are region locked** **(Incorrect)**
- ( ) **The required IAM permissions are missing**
- ( ) **The EBS volume is encrypted**
- ( ) **EBS volumes are AZ locked** **(Correct)**

Explanation
Correct option:

EBS volumes are AZ locked

An Amazon EBS volume is a durable, block-level storage device that you can attach to your instances. After you attach a volume to an instance, you can use it as you would use a physical hard drive. EBS volumes are flexible. For current-generation volumes attached to current-generation instance types, you can dynamically increase size, modify the provisioned IOPS capacity, and change volume type on live production volumes.

When you create an EBS volume, it is automatically replicated within its Availability Zone to prevent data loss due to the failure of any single hardware component. You can attach an EBS volume to an EC2 instance in the same Availability Zone.

Incorrect options:

EBS volumes are region locked - It's confined to an Availability Zone and not by region.

The required IAM permissions are missing - This is a possibility as well but if permissions are not an issue then you are still confined to an availability zone.

The EBS volume is encrypted - This doesn't affect the ability to attach an EBS volume.

Reference:

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

Question 58: **Correct**

A university has created a student portal that is accessible through a smartphone app and web application. The smartphone app is available in both Android and IOS and the web application works on most major browsers. Students will be able to do group study online and create forum questions. All changes made via smartphone devices should be available even when offline and should synchronize with other devices.

Which of the following AWS services will meet these requirements?

- (●) **Cognito Sync** (Correct)
- ( ) **BeanStalk**
- ( ) **Cognito User Pools**
- ( ) **Cognito Identity Pools**

Explanation
Correct option:

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. The client libraries cache data locally so your app can read and write data regardless of device connectivity status. When the device is online, you can synchronize data, and if you set up push sync, notify other devices immediately that an update is available.

Incorrect options:

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 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.

Exam Alert:

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

Features of Amazon Cognito

User pools

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

User pools provide:

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

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

Identity pools

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

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

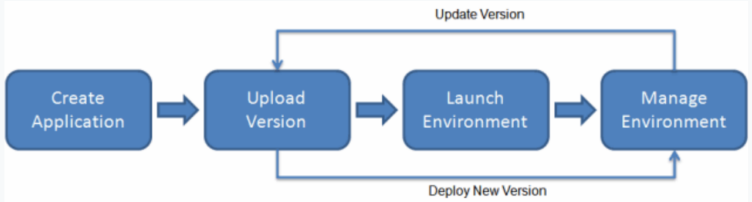
To save user profile information, your identity pool needs to be integrated with a user pool.

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

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

Beanstalk - 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

Reference:

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

Question 59: **Correct**

A pharmaceutical company uses Amazon EC2 instances for application hosting and Amazon CloudFront for content delivery. A new research paper with critical findings has to be shared with a research team that is spread across the world.

Which of the following represents the most optimal solution to address this requirement without compromising the security of the content?

- ○ **Using CloudFront's Field-Level Encryption to help protect sensitive data**
- ◉ **Use CloudFront signed URL feature to control access to the file** **(Correct)**
- ○ **Configure AWS Web Application Firewall (WAF) to monitor and control the HTTP and HTTPS requests that are forwarded to CloudFront**
- ○ **Use CloudFront signed cookies feature to control access to the file**

Explanation

Correct option:

Use CloudFront signed URL feature to control access to the file

A signed URL includes additional information, for example, expiration date and time, that gives you more control over access to your content.

Here's an overview of how you configure CloudFront for signed URLs and how CloudFront responds when a user uses a signed URL to request a file:

1. In your CloudFront distribution, specify one or more trusted key groups, which contain the public keys that CloudFront can use to verify the URL signature. You use the corresponding private keys to sign the URLs.
2. Develop your application to determine whether a user should have access to your content and to create signed URLs for the files or parts of your application that you want to restrict access to.
3. A user requests a file for which you want to require signed URLs. Your application verifies that the user is entitled to access the file: they've signed in, they've paid for access to the content, or they've met some other requirement for access.
4. Your application creates and returns a signed URL to the user. The signed URL allows the user to download or stream the content.

This step is automatic; the user usually doesn't have to do anything additional to access the content. For example, if a user is accessing your content in a web browser, your application returns the signed URL to the browser. The browser immediately uses the signed URL to access the file in the CloudFront edge cache without any intervention from the user.

1. CloudFront uses the public key to validate the signature and confirm that the URL hasn't been tampered with. If the signature is invalid, the request is rejected. If the request meets the requirements in the policy statement, CloudFront does the standard operations: determines whether the file is already in the edge cache, forwards the request to the origin if necessary, and returns the file to the user.

Incorrect options:

Use CloudFront signed cookies feature to control access to the file - CloudFront signed cookies allow you to control who can access your content when you don't want to change your current URLs or when you want to provide access to multiple restricted files, for example, all of the files in the subscribers' area of a website. Our requirement has only one file that needs to be shared and hence signed URL is the optimal solution.

Signed URLs take precedence over signed cookies. If you use both signed URLs and signed cookies to control access to the same files and a viewer uses a signed URL to request a file, CloudFront determines whether to return the file to the viewer based only on the signed URL.

Configure AWS Web Application Firewall (WAF) to monitor and control the HTTP and HTTPS requests that are forwarded to CloudFront - AWS WAF is a web application firewall that lets you monitor the HTTP and HTTPS requests that are forwarded to CloudFront, and lets you control access to your content. Based on conditions that you specify, such as the values of query strings or the IP addresses that requests originate from, CloudFront responds to requests either with the requested content or with an HTTP status code 403 (Forbidden). A firewall is optimal for broader use cases than restricted access to a single file.

Using CloudFront's Field-Level Encryption to help protect sensitive data - CloudFront's field-level encryption further encrypts sensitive data in an HTTPS form using field-specific encryption keys (which you supply) before a POST request is forwarded to your origin. This ensures that sensitive data can only be decrypted and viewed by certain components or services in your application stack. This feature is not useful for the given use case.

References:

https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/private-content-signed-urls.html

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

https://aws.amazon.com/about-aws/whats-new/2017/12/introducing-field-level-encryption-on-amazon-cloudfront/

Question 60: **Correct**

A company is using a Border Gateway Protocol (BGP) based AWS VPN connection to connect from its on-premises data center to Amazon EC2 instances in the company's account. The development team can access an EC2 instance in subnet A but is unable to access an EC2 instance in subnet B in the same VPC.

Which logs can be used to verify whether the traffic is reaching subnet B?

- (●) **VPC Flow Logs** **(Correct)**
- ( ) **Subnet logs**
- ( ) **BGP logs**
- ( ) **VPN logs**

Explanation
Correct option:

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 can be published to Amazon CloudWatch Logs or Amazon S3. After you've created a flow log, you can retrieve and view its data in the chosen destination.

You can create a flow log for a VPC, a subnet, or a network interface. If you create a flow log for a subnet or VPC, each network interface in that subnet or VPC is monitored.

Flow log data for a monitored network interface is recorded as flow log records, which are log events consisting of fields that describe the traffic flow.

To create a flow log, you specify:

1. The resource for which to create the flow log
2. The type of traffic to capture (accepted traffic, rejected traffic, or all traffic)
3. The destinations to which you want to publish the flow log data

Incorrect options:

VPN logs

Subnet logs

BGP logs

These three options are incorrect and have been added as distractors.

Reference:

https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html

Question 61: **Correct**

A company wants to automate its order fulfillment and inventory tracking workflow. Starting from order creation to updating inventory to shipment, the entire process has to be tracked, managed and updated automatically.

Which of the following would you recommend as the most optimal solution for this requirement?

- ○ **Use Amazon SNS to develop event-driven applications that can share information**
- ○ **Configure Amazon EventBridge to track the flow of work from order management to inventory tracking systems**
- ◉ **Use AWS Step Functions to coordinate and manage the components of order management and inventory tracking workflow** **(Correct)**
- ○ **Use Amazon Simple Queue Service (Amazon SQS) queue to pass information from order management to inventory tracking workflow**

Explanation

Correct option:

Use AWS Step Functions to coordinate and manage the components of order management and inventory tracking workflow

AWS Step Functions is a serverless function orchestrator that makes it easy to sequence AWS Lambda functions and multiple AWS services into business-critical applications. Through its visual interface, you can create and run a series of checkpointed and event-driven workflows that maintain the application state. The output of one step acts as an input to the next. Each step in your application executes in order, as defined by your business logic.

AWS Step Functions enables you to implement a business process as a series of steps that make up a workflow. The individual steps in the workflow can invoke a Lambda function or a container that has some business logic, update a database such as DynamoDB or publish a message to a queue once that step or the entire workflow completes execution.

Benefits of Step Functions:

Build and update apps quickly: AWS Step Functions lets you build visual workflows that enable the fast translation of business requirements into technical requirements. You can build applications in a matter of minutes, and when needs change, you can swap or reorganize components without customizing any code.

Improve resiliency: AWS Step Functions manages state, checkpoints and restarts for you to make sure that your application executes in order and as expected. Built-in try/catch, retry and rollback capabilities deal with errors and exceptions automatically.

Write less code: AWS Step Functions manages the logic of your application for you and implements basic primitives such as branching, parallel execution, and timeouts. This removes extra code that may be repeated in your microservices and functions.

How Step Functions work:

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

Incorrect options:

Use Amazon Simple Queue Service (Amazon SQS) queue to pass information from order management to inventory tracking workflow - You should consider AWS Step Functions when you need to coordinate service components in the development of highly scalable and auditable applications. You should consider using Amazon Simple Queue Service (Amazon SQS), when you need a reliable, highly scalable, hosted queue for sending, storing, and receiving messages between services. Step Functions keeps track of all tasks and events in an application. Amazon SQS requires you to implement your own application-level tracking, especially if your application uses multiple queues.

Configure Amazon EventBridge to track the flow of work from order management to inventory tracking systems - Both Amazon EventBridge and Amazon SNS can be used to develop event-driven applications, and your choice will depend on your specific needs. Amazon EventBridge is recommended when you want to build an application that reacts to events from SaaS applications and/or AWS services. Amazon EventBridge is the only event-based service that integrates directly with third-party SaaS partners.

Use Amazon SNS to develop event-driven applications that can share information - Amazon SNS is recommended when you want to build an application that reacts to high throughput or low latency messages published by other applications or microservices (as Amazon SNS provides nearly unlimited throughput), or for applications that need very high fan-out (thousands or millions of endpoints).

References:

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

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

Question 62: **Correct**

A junior developer has been asked to configure access to an Amazon EC2 instance hosting a web application. The developer has configured a new security group to permit incoming HTTP traffic from 0.0.0.0/0 and retained any default outbound rules. A custom Network Access Control List (NACL) connected with the instance's subnet is configured to permit incoming HTTP traffic from 0.0.0.0/0 and retained any default outbound rules.

Which of the following solutions would you suggest if the EC2 instance needs to accept and respond to requests from the internet?

- ○ **The configuration is complete on the EC2 instance for accepting and responding to requests**
- ○ **An outbound rule on the security group has to be configured, to allow the response to be sent to the client on the HTTP port**
- ◉ **An outbound rule must be added to the Network ACL (NACL) to allow the response to be sent to the client on the ephemeral port range** **(Correct)**
- ○ **Outbound rules need to be configured both on the security group and on the NACL for sending responses to the Internet Gateway**

Explanation
Correct option:

An outbound rule must be added to the Network ACL (NACL) to allow the response to be sent to the client on the ephemeral port range

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. By default, each custom Network ACL denies all inbound and outbound traffic until you add rules.

To enable the connection to a service running on an instance, the associated network ACL must allow both: 1. Inbound traffic on the port that the service is listening on 2. Outbound traffic to 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 becomes the destination port for return traffic from the service. Outbound traffic to the ephemeral port must be allowed in the network ACL.

Incorrect options:

The configuration is complete on the EC2 instance for accepting and responding to

Explanation
Correct option:

An outbound rule must be added to the Network ACL (NACL) to allow the response to be sent to the client on the ephemeral port range

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. By default, each custom Network ACL denies all inbound and outbound traffic until you add rules.

To enable the connection to a service running on an instance, the associated network ACL must allow both: 1. Inbound traffic on the port that the service is listening on 2. Outbound traffic to 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 becomes the destination port for return traffic from the service. Outbound traffic to the ephemeral port must be allowed in the network ACL.

Incorrect options:

The configuration is complete on the EC2 instance for accepting and responding to requests - As explained above, this is an incorrect statement.

An outbound rule on the security group has to be configured, to allow the response to be sent to the client on the HTTP port - Security groups are stateful. Therefore you don't need a rule that allows responses to inbound traffic.

*Outbound rules need to be configured both on the security group and on the NACL for sending responses to the Internet Gateway** - Security Groups are stateful. Hence, return traffic is automatically allowed, so there is no need to configure an outbound rule on the security group.

References:

https://aws.amazon.com/premiumsupport/knowledge-center/resolve-connection-sg-acl-inbound/

https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html#nacl-ephemeral-ports

Question 63: **Correct**

The app development team at a social gaming mobile app wants to simplify the user sign up process for the app. The team is looking for a fully managed scalable solution for user management in anticipation of the rapid growth that the app foresees.

As a Developer Associate, which of the following solutions would you suggest so that it requires the LEAST amount of development effort?

- ○ **Create a custom solution using Lambda and DynamoDB to facilitate sign up and user management for the mobile app**
- ◉ **Use Cognito User pools to facilitate sign up and user management for the mobile app** **(Correct)**
- ○ **Use Cognito Identity pools to facilitate sign up and user management for the mobile app**
- ○ **Create a custom solution using EC2 and DynamoDB to facilitate sign up and user management for the mobile app**

Explanation

Correct option:

Use Cognito User pools to facilitate sign up and user management for the mobile app

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

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.

Cognito is fully managed by AWS and works out of the box so it meets the requirements for the given use-case.

What Is Amazon Cognito?

PDF | Kindle

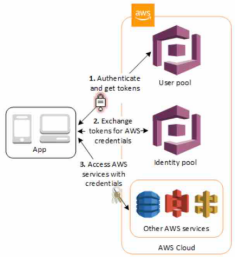
Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. Your users can sign in directly with a user name and password, or through a third party such as Facebook, Amazon, Google or Apple.

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

An Amazon Cognito user pool and identity pool used together

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

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



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

Incorrect options:

Use Cognito Identity pools to facilitate sign up and user management for the mobile app - 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.

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

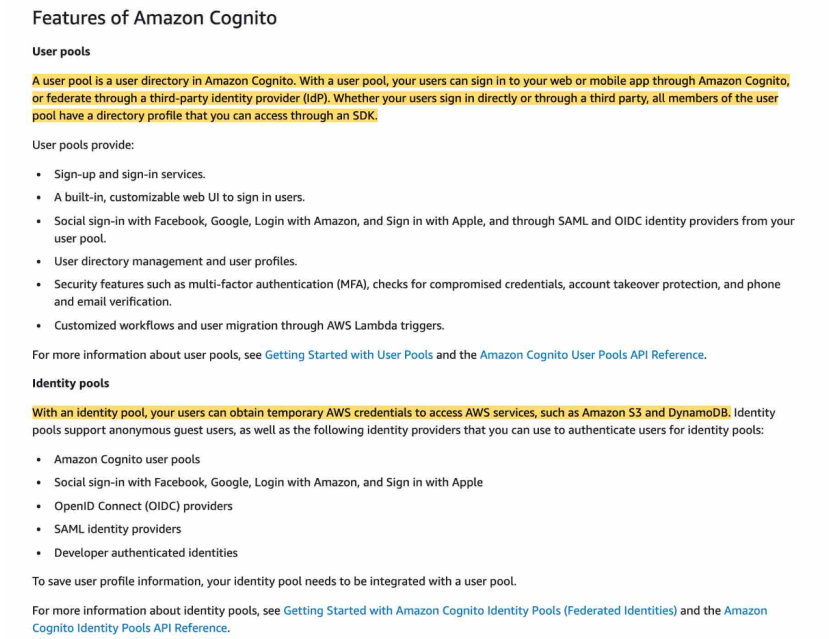
A user pool is a user directory in Amazon Cognito. With a user pool, your users can sign in to your web or mobile app through Amazon Cognito, or federate through a third-party identity provider (IdP). Whether your users sign in directly or through a third party, all members of the user pool have a directory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

Exam Alert:

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



Features of Amazon Cognito

User pools

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

User pools provide:

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

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

Identity pools

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

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

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

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

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

Create a custom solution with EC2 and DynamoDB to facilitate sign up and user management for the mobile app

Create a custom solution with Lambda and DynamoDB to facilitate sign up and user management for the mobile app

As the problem statement mentions that the solution needs to be fully managed and should require the least amount of development effort, so you cannot use EC2 or Lambda functions with DynamoDB to create a custom solution.

Reference:

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

Question 64: **Correct**

A CRM application is hosted on Amazon EC2 instances with the database tier using DynamoDB. The customers have raised privacy and security concerns regarding sending and receiving data across the public internet.

As a developer associate, which of the following would you suggest as an optimal solution for providing communication between EC2 instances and DynamoDB without using the public internet?

- (●) **Configure VPC endpoints for DynamoDB that will provide required internal access without using public internet** **(Correct)**
- ( ) **Create a NAT Gateway to provide the necessary communication channel between EC2 instances and DynamoDB**
- ( ) **Create an Internet Gateway to provide the necessary communication channel between EC2 instances and DynamoDB**
- ( ) **The firm can use a virtual private network (VPN) to route all DynamoDB network traffic through their own corporate network infrastructure**

Explanation

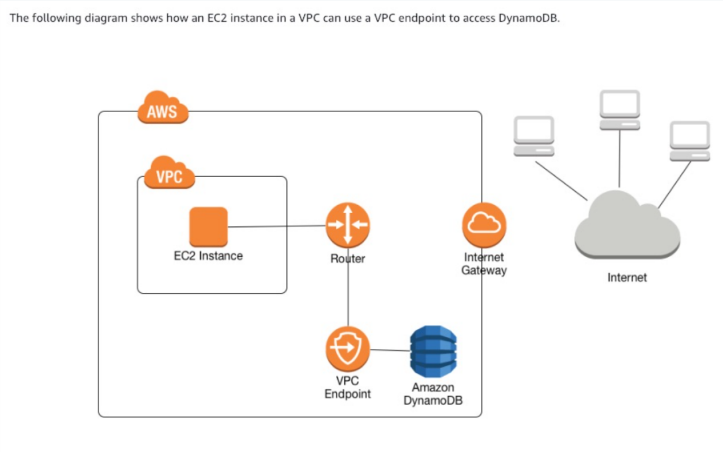
Correct option:

Configure VPC endpoints for DynamoDB that will provide required internal access without using public internet

When you create a VPC endpoint for DynamoDB, any requests to a DynamoDB endpoint within the Region (for example, dynamodb.us-west-2.amazonaws.com) are routed to a private DynamoDB endpoint within the Amazon network. You don't need to modify your applications running on EC2 instances in your VPC. The endpoint name remains the same, but the route to DynamoDB stays entirely within the Amazon network, and does not access the public internet. You use endpoint policies to control access to DynamoDB. Traffic between your VPC and the AWS service does not leave the Amazon network.

Using Amazon VPC Endpoints to Access DynamoDB:

Using Amazon VPC Endpoints to Access DynamoDB:



via - https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/vpc-endpoints-dynamodb.html

Incorrect options:

The firm can use a virtual private network (VPN) to route all DynamoDB network traffic through their own corporate network infrastructure - You can address the requested security concerns by using a virtual private network (VPN) to route all DynamoDB network traffic through your own corporate network infrastructure. However, this approach can introduce bandwidth and availability challenges and hence is not an optimal solution here.

Create a NAT Gateway to provide the necessary communication channel between EC2 instances and DynamoDB - You can use a network address translation (NAT) gateway to enable instances in a private subnet to connect to the internet or other AWS services, but prevent the internet from initiating a connection with those instances. NAT Gateway is not useful here since the instance and DynamoDB are present in AWS network and do not need NAT Gateway for communicating with each other.

Create an Internet Gateway to provide the necessary communication channel between EC2 instances and DynamoDB - An internet gateway is a horizontally scaled, redundant, and highly available VPC component that allows communication between your VPC and the internet. An internet gateway serves two purposes: to provide a target in your VPC route tables for internet-routable traffic, and to perform network address translation (NAT) for instances that have been assigned public IPv4 addresses. Using an Internet Gateway would imply that the EC2 instances are connecting to DynamoDB using the public internet. Therefore, this option is incorrect.

References:

https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/vpc-endpoints-dynamodb.html

Question 65: **Incorrect**

While troubleshooting, a developer realized that the Amazon EC2 instance is unable to connect to the Internet using the Internet Gateway.

Which conditions should be met for Internet connectivity to be established? (Select two)

- [ ] **The route table in the instance's subnet should have a route to an Internet Gateway** (Correct)
- [ ] **The subnet has been configured to be Public and has no access to the internet**
- [x] **The network ACLs associated with the subnet must have rules to allow inbound and outbound traffic** (Correct)
- [ ] **The instance's subnet is associated with multiple route tables with conflicting configurations**
- [x] **The instance's subnet is not associated with any route table** (Incorrect)

Explanation

Correct options:

The network ACLs associated with the subnet must have rules to allow inbound and outbound traffic - The network access control lists (ACLs) that are associated with the subnet must have rules to allow inbound and outbound traffic on port 80 (for HTTP traffic) and port 443 (for HTTPs traffic). This is a necessary condition for Internet Gateway connectivity

The route table in the instance's subnet should have a route to an Internet Gateway - A route table contains a set of rules, called routes, that are used to determine where network traffic from your subnet or gateway is directed. The route table in the instance's subnet should have a route defined to the Internet Gateway.

Incorrect options:

The instance's subnet is not associated with any route table - This is an incorrect statement. A subnet is implicitly associated with the main route table if it is not explicitly associated with a particular route table. So, a subnet is always associated with some route table.

The instance's subnet is associated with multiple route tables with conflicting configurations - This is an incorrect statement. A subnet can only be associated with one route table at a time.

Explanation
Correct options:

The network ACLs associated with the subnet must have rules to allow inbound and outbound traffic - The network access control lists (ACLs) that are associated with the subnet must have rules to allow inbound and outbound traffic on port 80 (for HTTP traffic) and port 443 (for HTTPs traffic). This is a necessary condition for Internet Gateway connectivity

The route table in the instance's subnet should have a route to an Internet Gateway - A route table contains a set of rules, called routes, that are used to determine where network traffic from your subnet or gateway is directed. The route table in the instance's subnet should have a route defined to the Internet Gateway.

Incorrect options:

The instance's subnet is not associated with any route table - This is an incorrect statement. A subnet is implicitly associated with the main route table if it is not explicitly associated with a particular route table. So, a subnet is always associated with some route table.

The instance's subnet is associated with multiple route tables with conflicting configurations - This is an incorrect statement. A subnet can only be associated with one route table at a time.

The subnet has been configured to be Public and has no access to internet - This is an incorrect statement. Public subnets have access to the internet via Internet Gateway.

Reference:

https://docs.aws.amazon.com/vpc/latest/userguide/VPC_Route_Tables.html