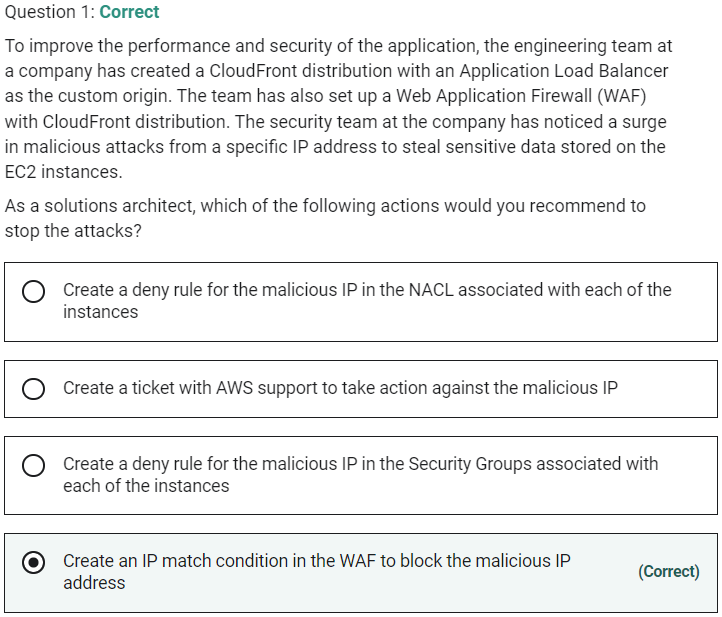
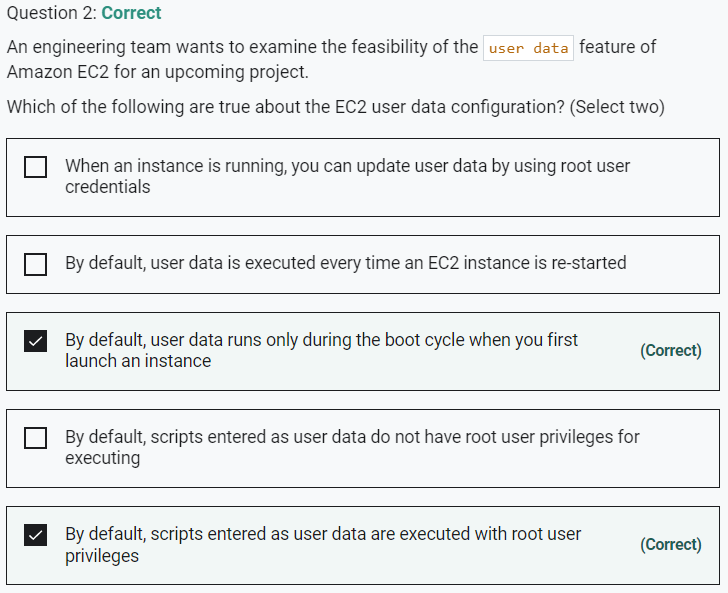
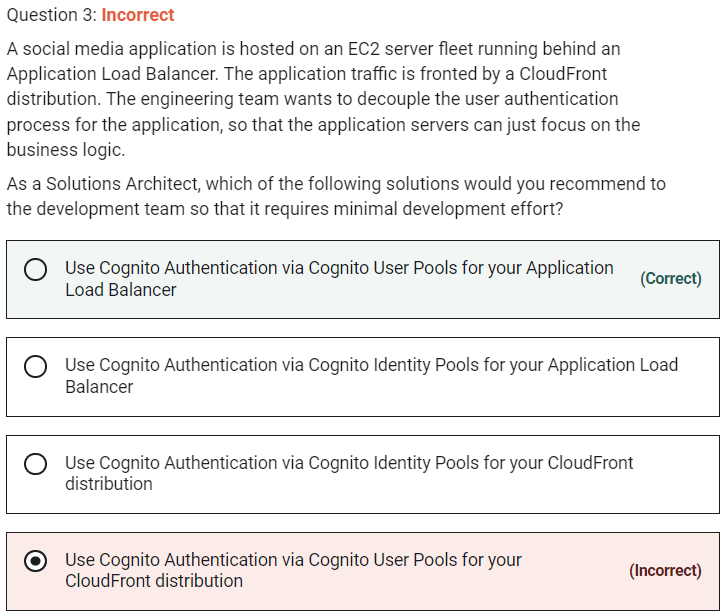
SAA-CO2

Practice Set-2







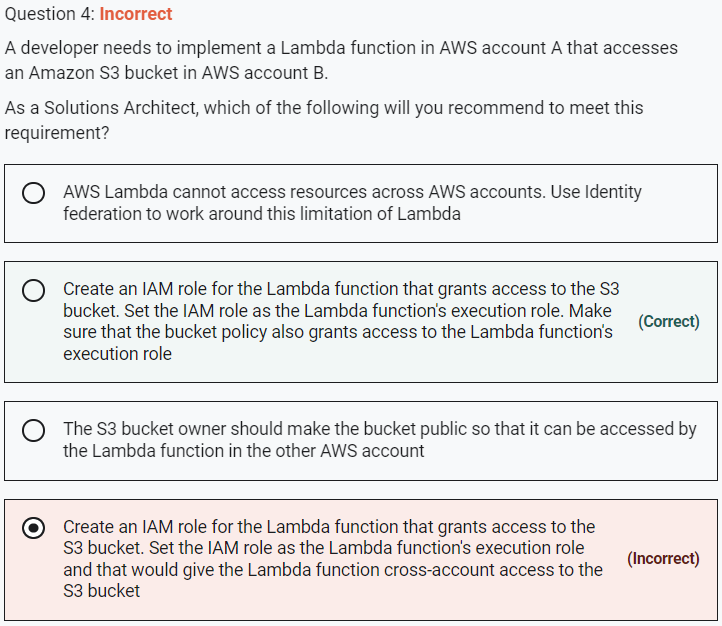
**Use Cognito Authentication via Cognito User Pools for your Application Load Balancer**

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

**Use Cognito Authentication via Cognito Identity Pools for your Application Load Balancer** - There is no such thing as using Cognito Authentication via Cognito Identity Pools for managing user authentication for the application. Application-specific user authentication can be provided via Cognito User Pools. Amazon Cognito identity pools provide temporary AWS credentials for users who are guests (unauthenticated) and for users who have been authenticated and received a token.

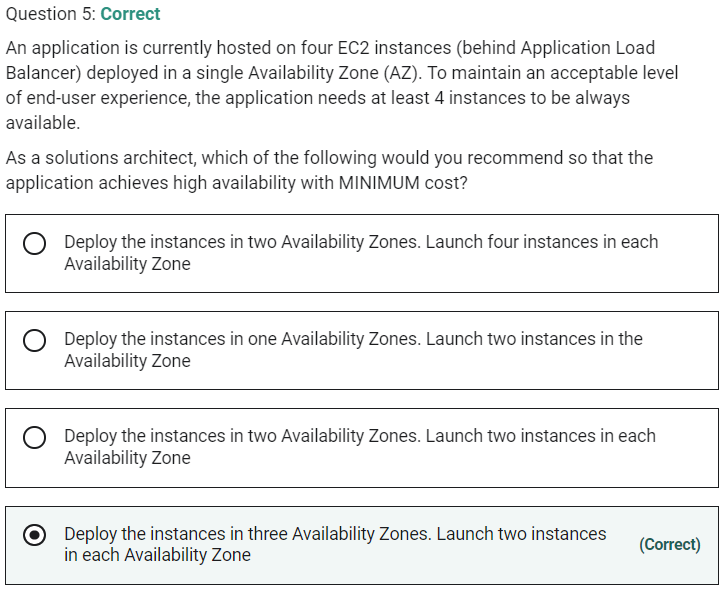
**Use Cognito Authentication via Cognito User Pools for your CloudFront distribution** - You cannot directly integrate Cognito User Pools with CloudFront distribution as you have to create a separate Lambda@Edge function to accomplish the authentication via Cognito User Pools. This involves additional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

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



If the IAM role that you create for the Lambda function is in the same AWS account as the bucket, then you don't need to grant Amazon S3 permissions on both the IAM role and the bucket policy. Instead, you can grant the permissions on the IAM role and then verify that the bucket policy doesn't explicitly deny access to the Lambda function role.

If the IAM role and the bucket are in different accounts, then you need to grant Amazon S3 permissions on both the IAM role and the bucket policy. Therefore, this is the right way of giving access to AWS Lambda for the given use-case.



**Deploy the instances in three Availability Zones. Launch two instances in each Availability Zone**

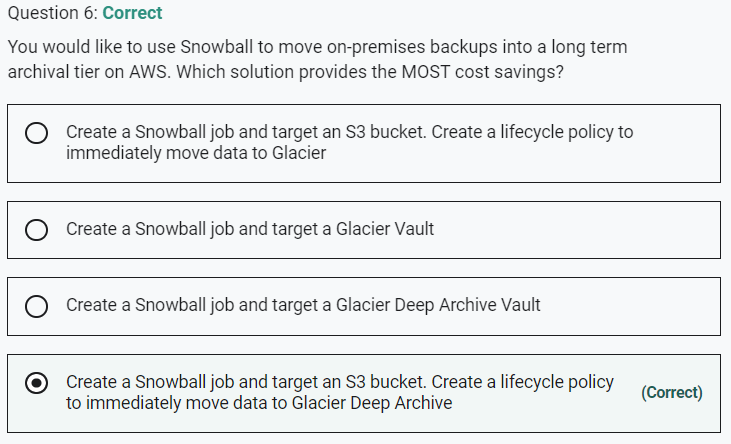
The correct option is to deploy the instances in three Availability Zones and launch two instances in each Availability Zone. Even if one of the AZs goes out of service, still we shall have 4 instances available and the application can maintain an acceptable level of end-user experience. Therefore, we can achieve high availability with just 6 instances in this case.

Incorrect options:

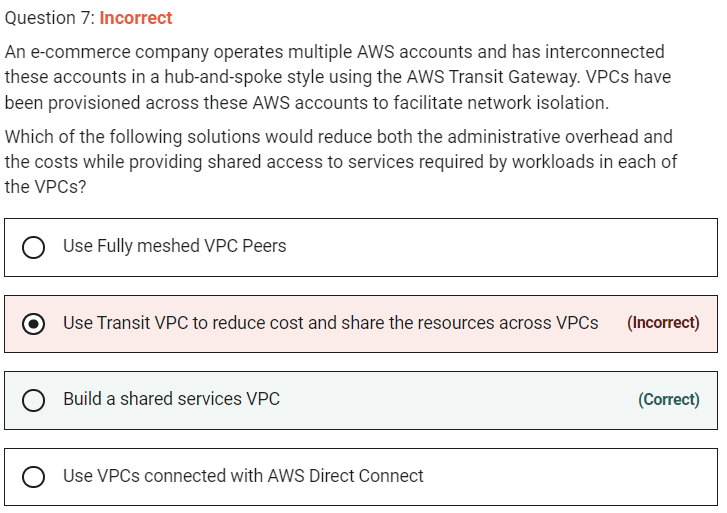
**Deploy the instances in two Availability Zones. Launch two instances in each Availability Zone** - When we launch two instances in two AZs, we run the risk of falling below the minimum acceptable threshold of 4 instances if one of the AZs fails. So this option is ruled out.

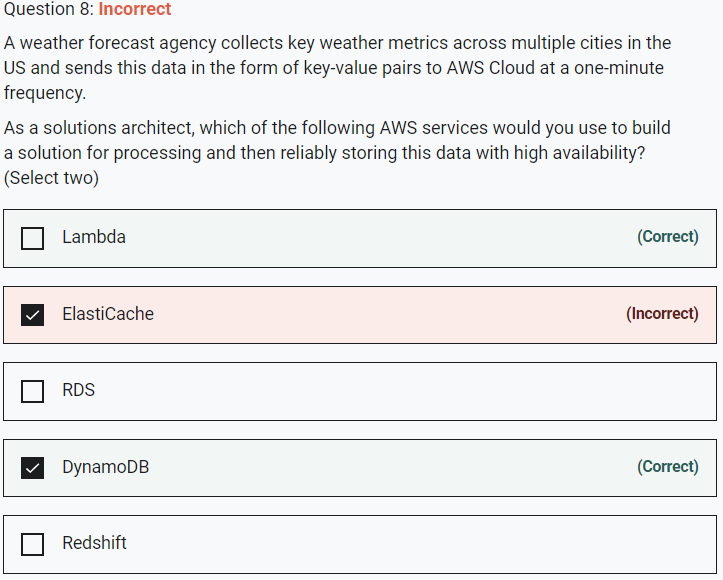
**Deploy the instances in two Availability Zones. Launch four instances in each Availability Zone** - When we launch four instances in two AZs, we have to bear costs for 8 instances which is NOT cost-optimal. So this option is ruled out.

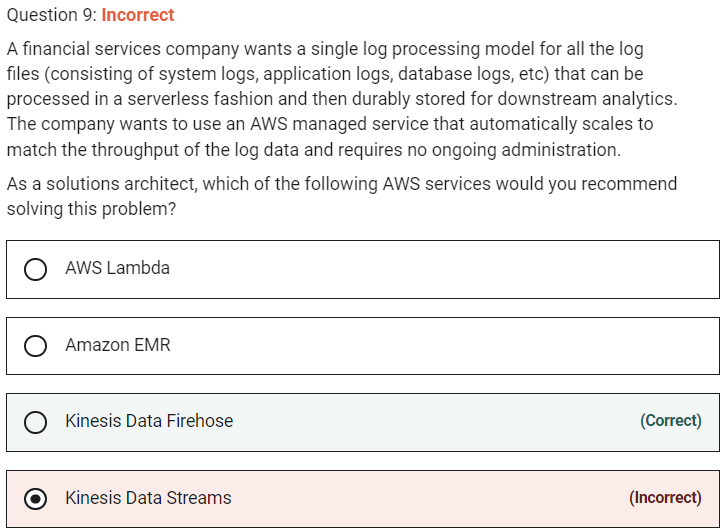
**Deploy the instances in one Availability Zones. Launch two instances in the Availability Zone** - We can't have just two instances in a single AZ as that is below the minimum acceptable threshold of 4 instances.



You can't move data directly from Snowball into Glacier, you need to go through S3 first, and then use a lifecycle policy. So this option is correct.



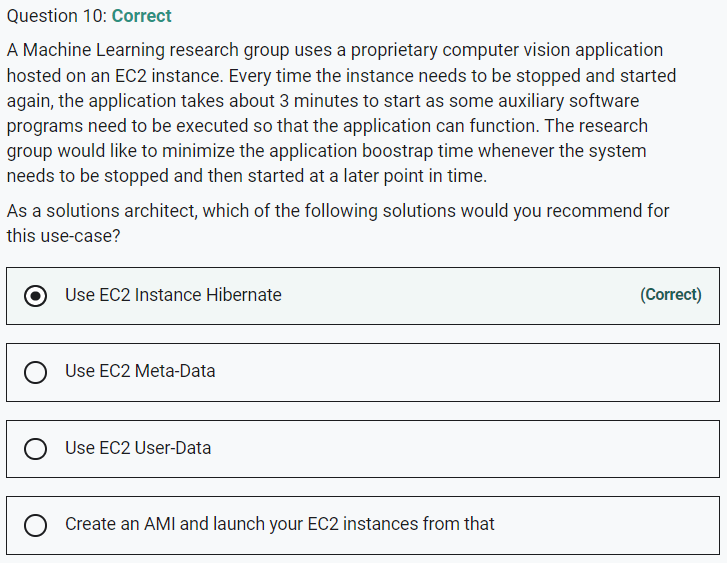




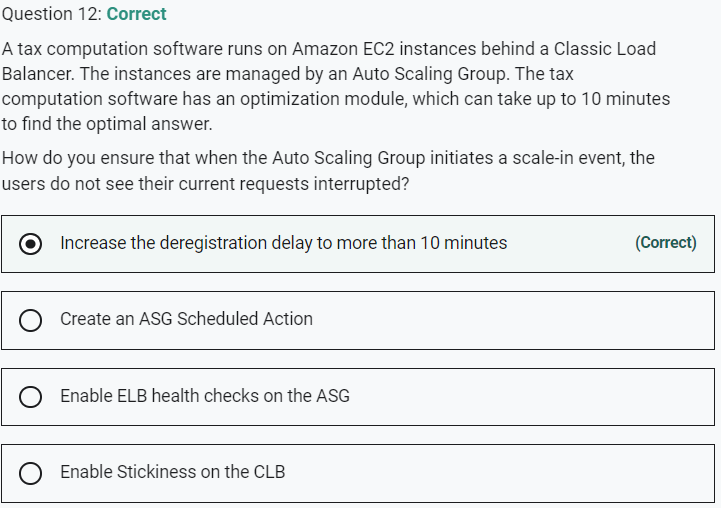
**Kinesis Data Firehose**

Amazon Kinesis Data Firehose is the easiest way to reliably load streaming data into data lakes, data stores, and analytics tools. It can capture, transform, and load streaming data into Amazon S3, Amazon Redshift, Amazon Elasticsearch Service, and Splunk, enabling near real-time analytics with existing business intelligence tools and dashboards you’re already using today. It is a fully managed service that automatically scales to match the throughput of your data and requires no ongoing administration. Therefore, this is the correct option.

**Kinesis Data Streams** - Amazon Kinesis Data Streams (KDS) is a massively scalable and durable real-time data streaming service. The throughput of an Amazon Kinesis data stream is designed to scale without limits via increasing the number of shards within a data stream. With Amazon Kinesis Data Streams, you can scale up to a sufficient number of shards (note, however, that you'll need to provision enough shards ahead of time). As it requires manual administration of shards, it's not the correct choice for the given use-case.

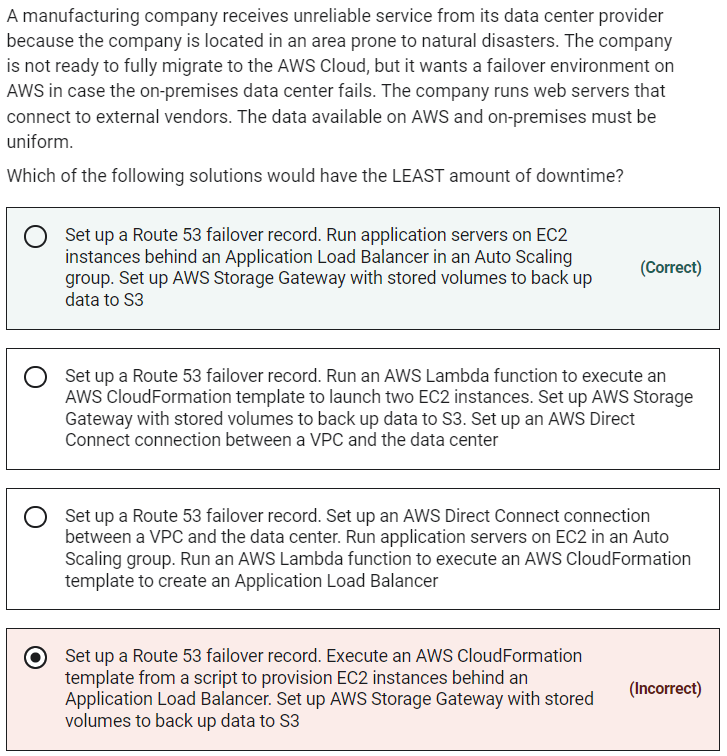






**Increase the deregistration delay to more than 10 minutes**

Elastic Load Balancing stops sending requests to targets that are deregistering. By default, Elastic Load Balancing waits 300 seconds before completing the deregistration process, which can help in-flight requests to the target to complete. We need to update this value to more than 10 minutes to allow our tax software to complete in-flight requests. Therefore this is the correct option.

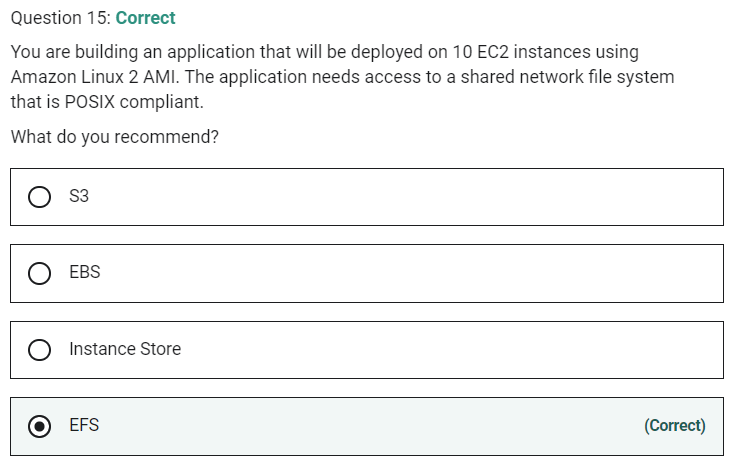


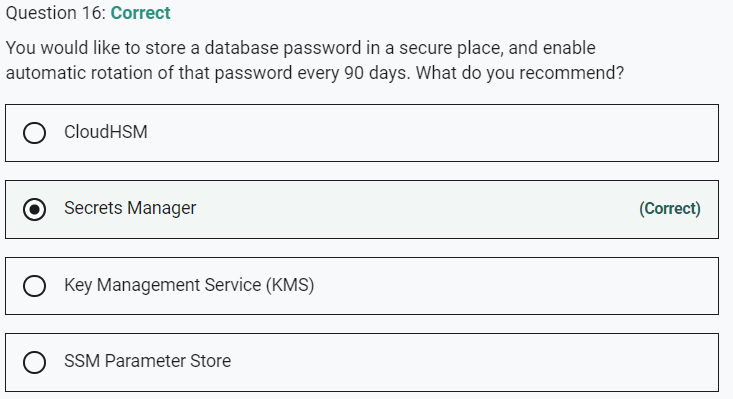
If you have multiple resources that perform the same function, you can configure DNS failover so that Route 53 will route your traffic from an unhealthy resource to a healthy resource.

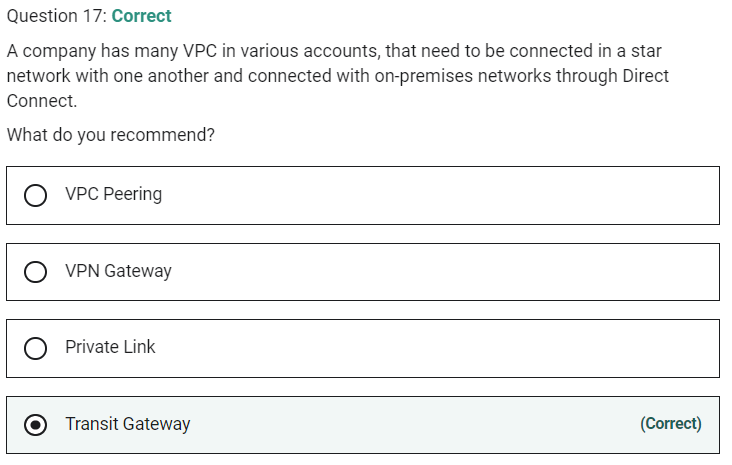
Elastic Load Balancing is used to automatically distribute your incoming application traffic across all the EC2 instances that you are running. You can use Elastic Load Balancing to manage incoming requests by optimally routing traffic so that no one instance is overwhelmed. Your load balancer acts as a single point of contact for all incoming web traffic to your Auto Scaling group.

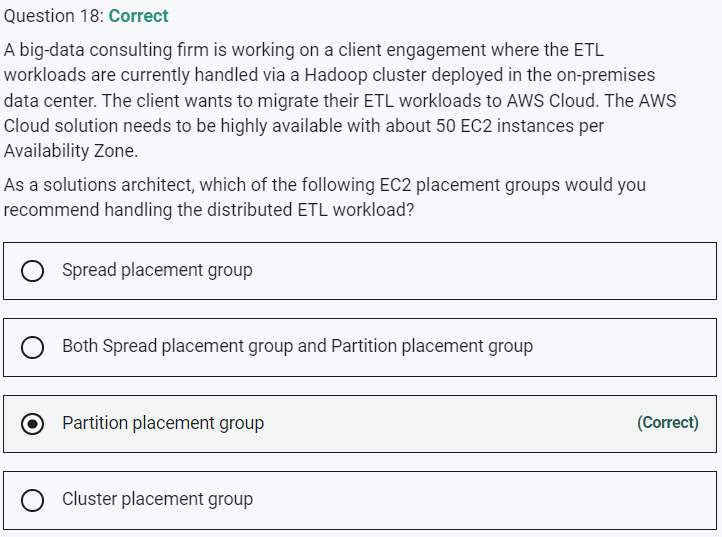
AWS Storage Gateway is a hybrid cloud storage service that gives you on-premises access to virtually unlimited cloud storage. Storage Gateway optimizes data transfer to AWS by sending only changed data and compressing data. Storage Gateway also integrates natively with Amazon S3 cloud storage which makes your data available for in-cloud processing.

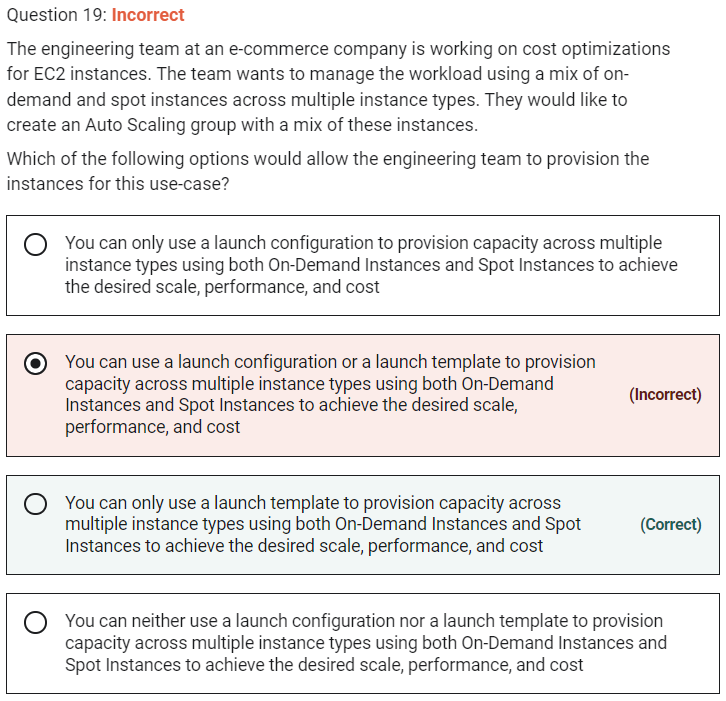












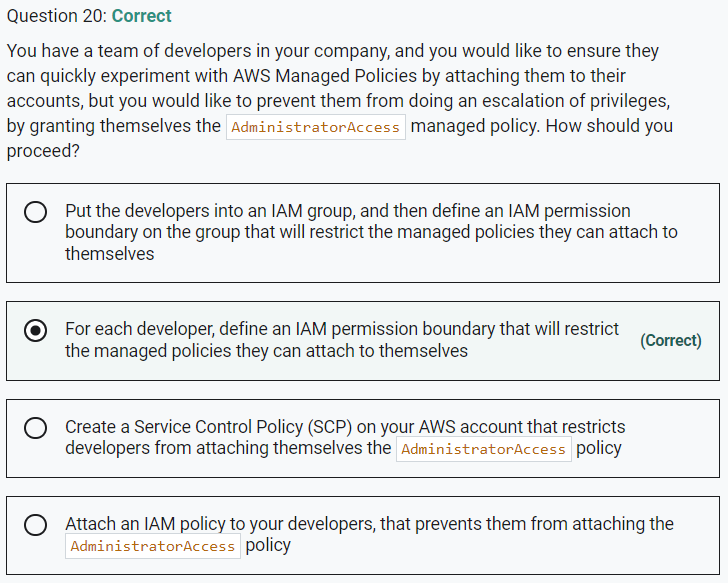
**You can only use a launch template to provision capacity across multiple instance types using both On-Demand Instances and Spot Instances to achieve the desired scale, performance, and cost**

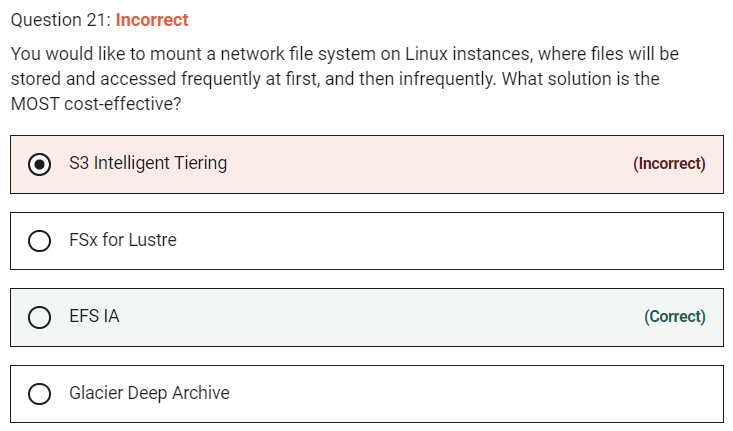
A launch template is similar to a launch configuration, in that it specifies instance configuration information such as the ID of the Amazon Machine Image (AMI), the instance type, a key pair, security groups, and the other parameters that you use to launch EC2 instances. Also, defining a launch template instead of a launch configuration allows you to have multiple versions of a template.

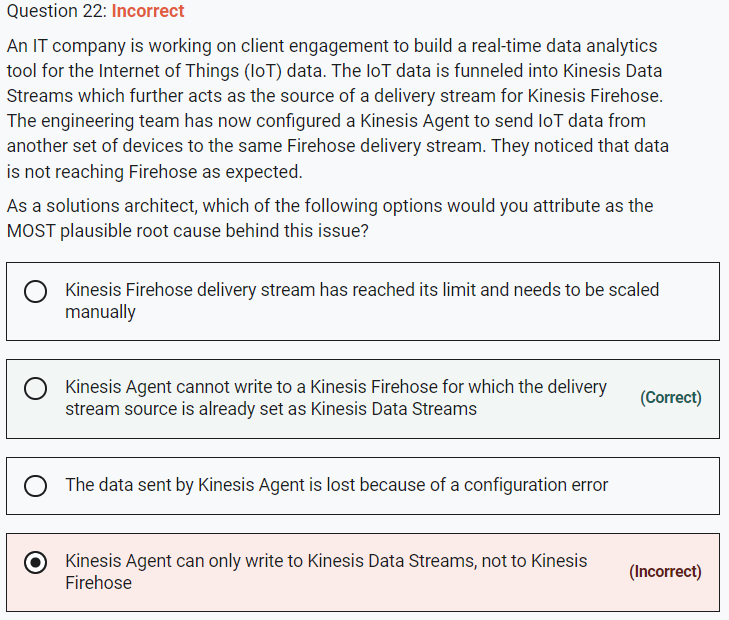
With launch templates, you can provision capacity across multiple instance types using both On-Demand Instances and Spot Instances to achieve the desired scale, performance, and cost. Hence this is the correct option.

A launch configuration is an instance configuration template that an Auto Scaling group uses to launch EC2 instances. When you create a launch configuration, you specify information for the instances such as the ID of the Amazon Machine Image (AMI), the instance type, a key pair, one or more security groups, and a block device mapping.

You cannot use a launch configuration to provision capacity across multiple instance types using both On-Demand Instances and Spot Instances. Therefore both these options are incorrect.







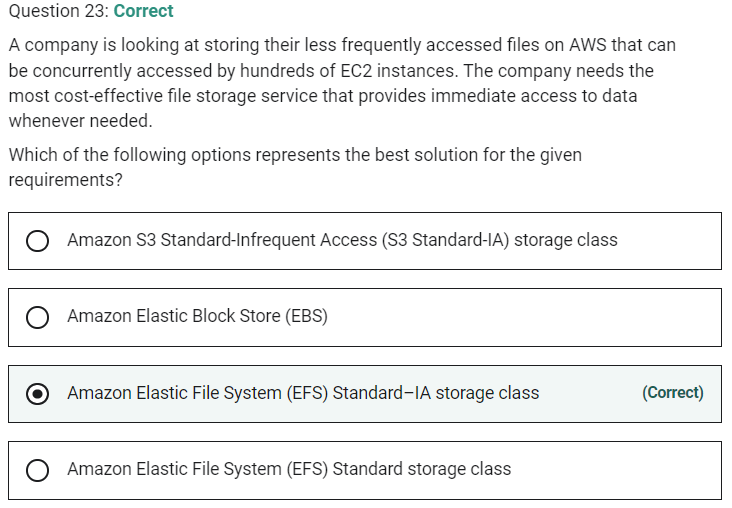
**Kinesis Agent cannot write to a Kinesis Firehose for which the delivery stream source is already set as Kinesis Data Streams**

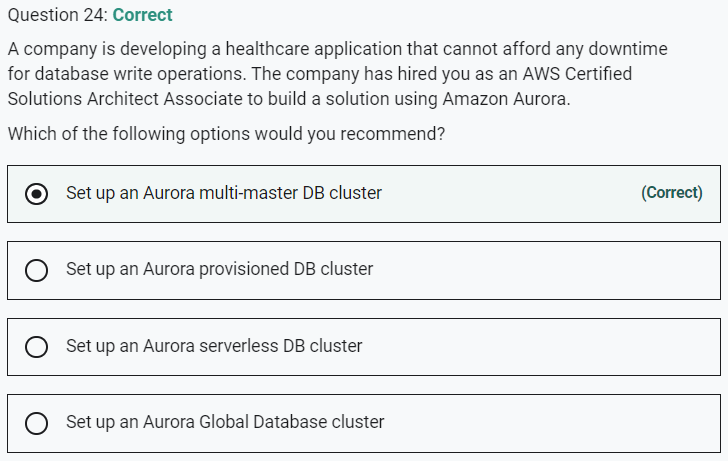
When a Kinesis data stream is configured as the source of a Firehose delivery stream, Firehose’s PutRecord and PutRecordBatch operations are disabled and Kinesis Agent cannot write to Firehose delivery stream directly. Data needs to be added to the Kinesis data stream through the Kinesis Data Streams PutRecord and PutRecords operations instead. Therefore, this option is correct.

Incorrect options:

**Kinesis Agent can only write to Kinesis Data Streams, not to Kinesis Firehose** - Kinesis Agent is a stand-alone Java software application that offers an easy way to collect and send data to Kinesis Data Streams or Kinesis Firehose. So this option is incorrect.

**Kinesis Firehose delivery stream has reached its limit and needs to be scaled manually** - Kinesis Firehose is a fully managed service that automatically scales to match the throughput of your data and requires no ongoing administration. Therefore this option is not correct.



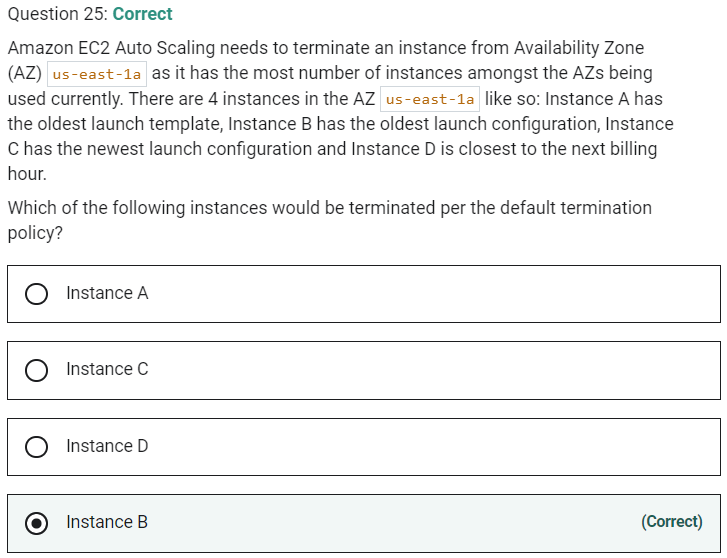


**Set up an Aurora serverless DB cluster**

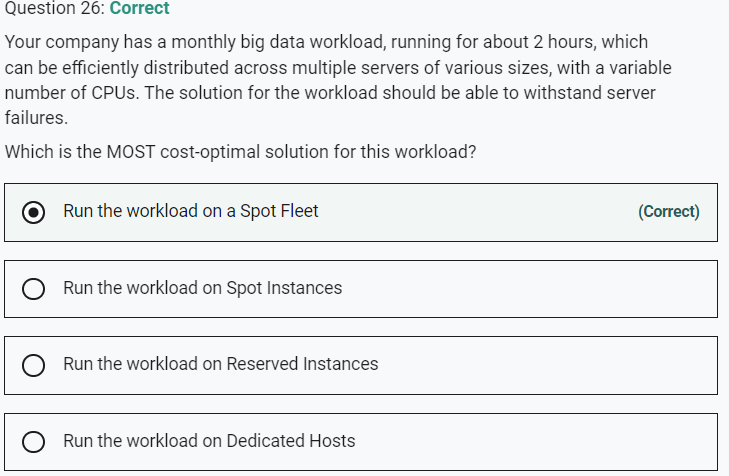
**Set up an Aurora provisioned DB cluster**

**Set up an Aurora Global Database cluster**

These three options represent Aurora single-master clusters. In a single-master cluster, a single DB instance performs all write operations and any other DB instances are read-only. If the writer DB instance becomes unavailable, a failover mechanism promotes one of the read-only instances to be the new writer. As there is a brief downtime during this failover, so these three options are incorrect for the given use case.

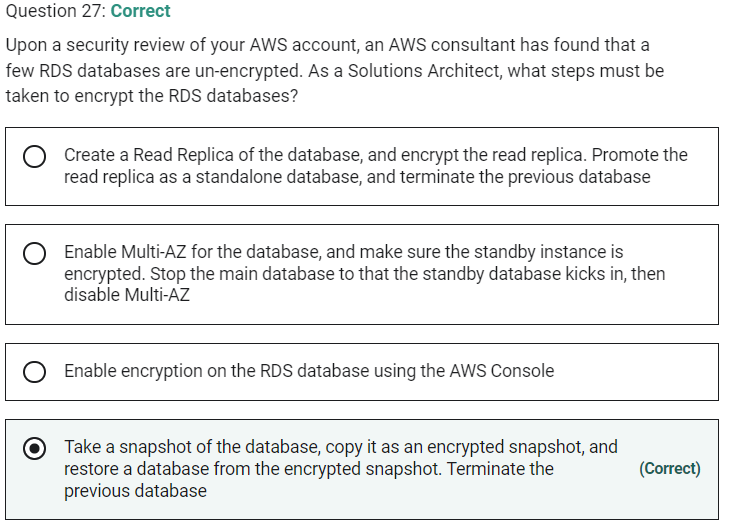


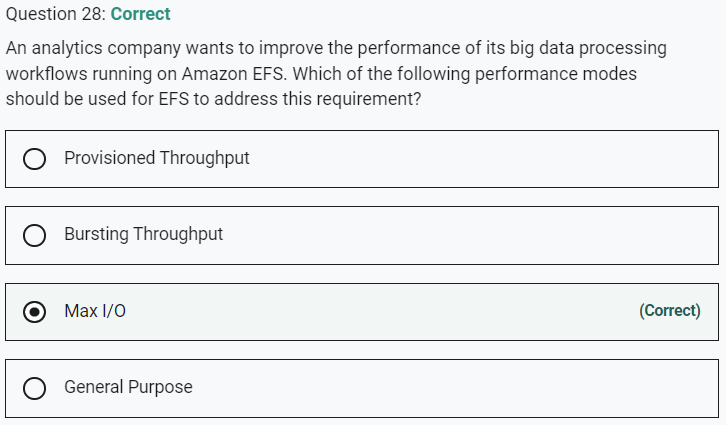
Per the default termination policy, the first priority is given to any allocation strategy for On-Demand vs Spot instances. As no such information has been provided for the given use-case, so this criterion can be ignored. The next priority is to consider any instance with the oldest launch template unless there is an instance that uses a launch configuration. So this rules out Instance A. Next, you need to consider any instance which has the oldest launch configuration. This implies Instance B will be selected for termination and Instance C will also be ruled out as it has the newest launch configuration. Instance D, which is closest to the next billing hour, is not selected as this criterion is last in the order of priority.



The Spot Fleet selects the Spot Instance pools that meet your needs and launches Spot Instances to meet the target capacity for the fleet. By default, Spot Fleets are set to maintain target capacity by launching replacement instances after Spot Instances in the fleet are terminated.

A Spot Instance is an unused EC2 instance that is available for less than the On-Demand price. Spot Instances provide great cost efficiency, but we need to select an instance type in advance. In this case, we want to use the most cost-optimal option and leave the selection of the cheapest spot instance to a Spot Fleet request, which can be optimized with the lowestPrice strategy. So this is the correct option

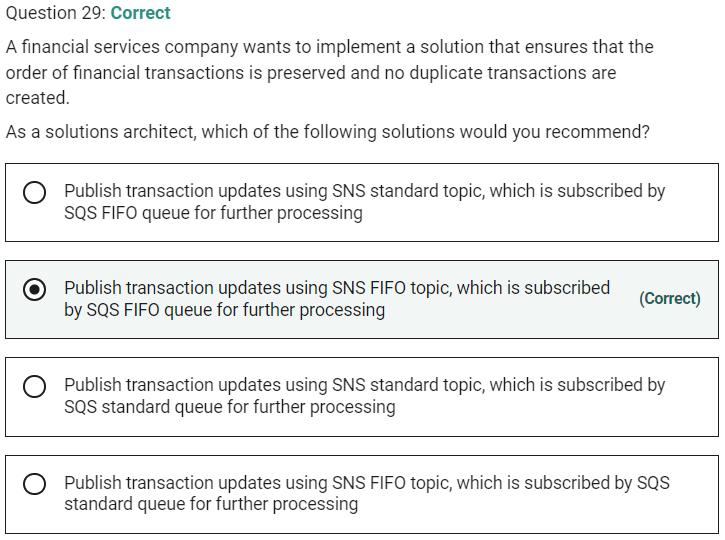


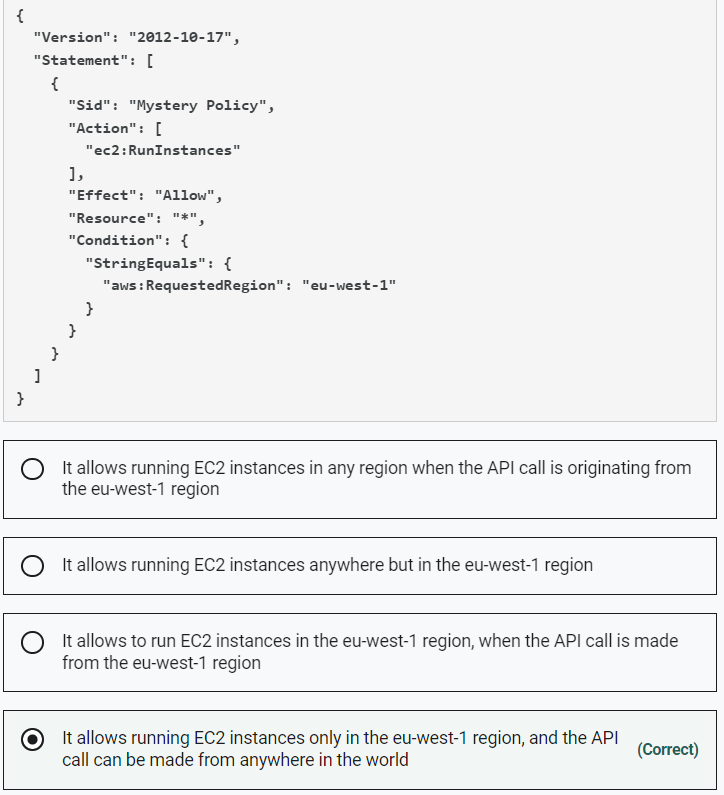


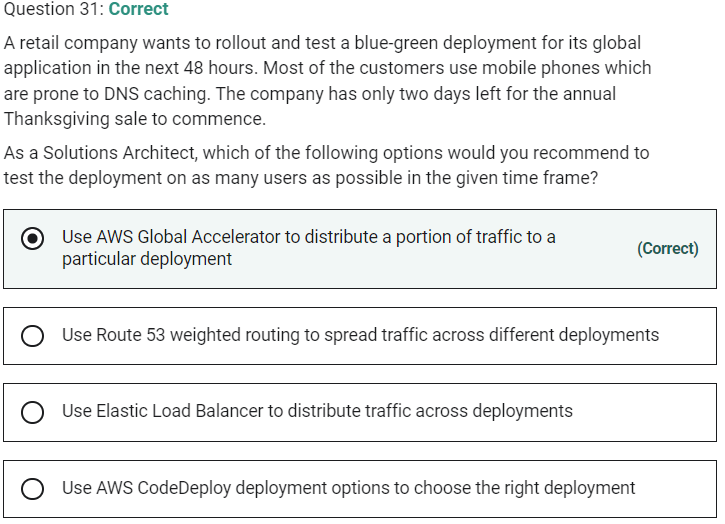
**Provisioned Throughput**

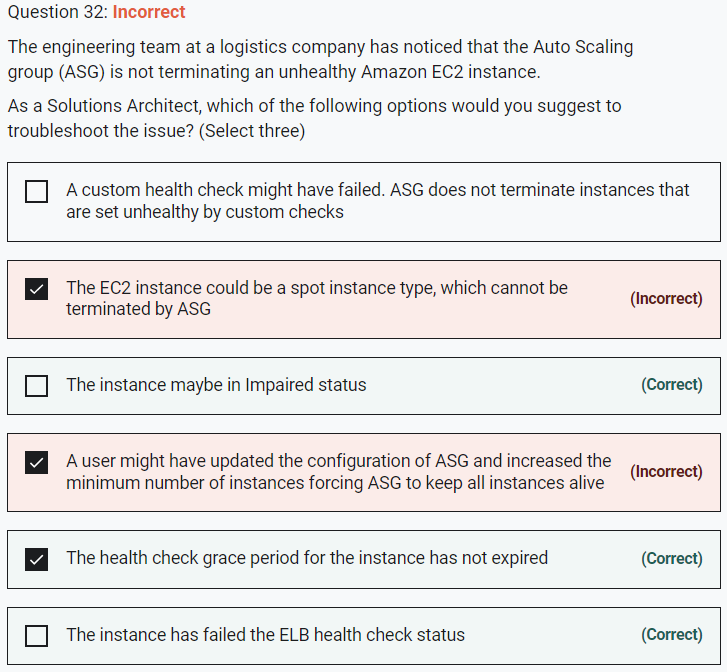
**Bursting Throughput**

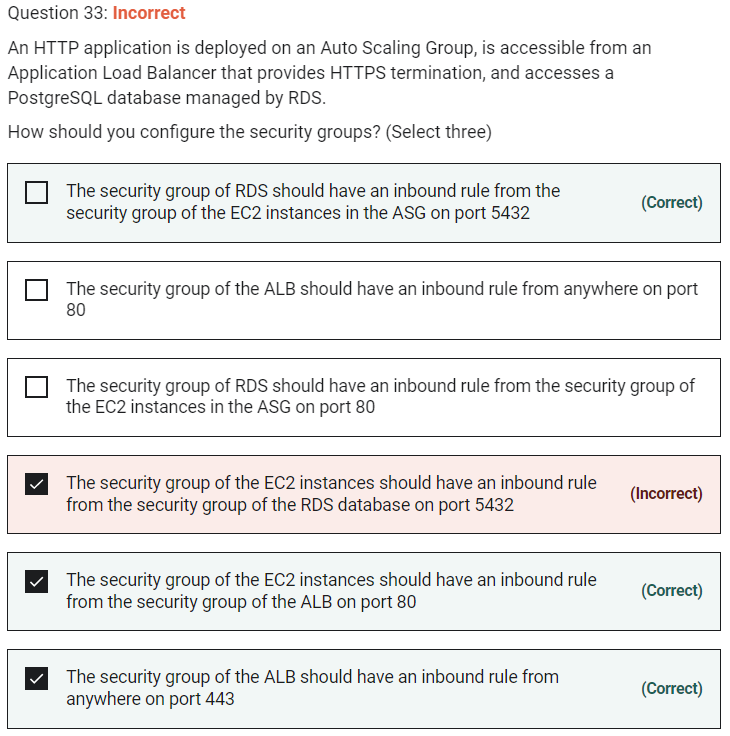
These two options have been added as distractors as these refer to the throughput mode of EFS and not the performance mode





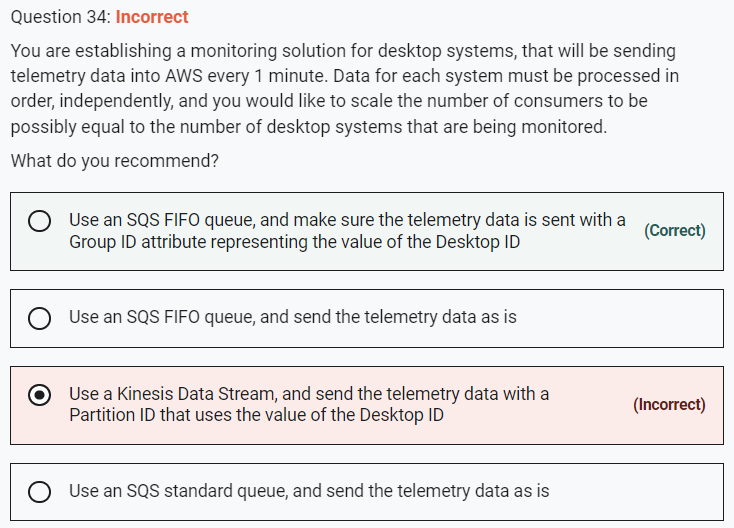


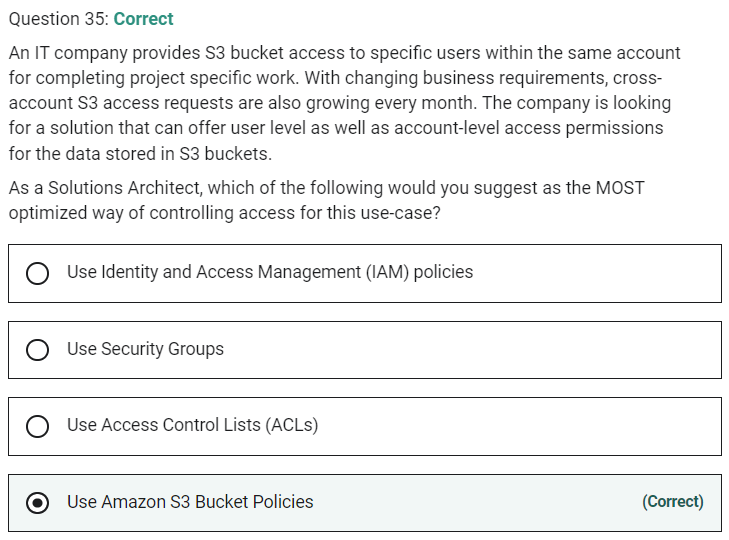




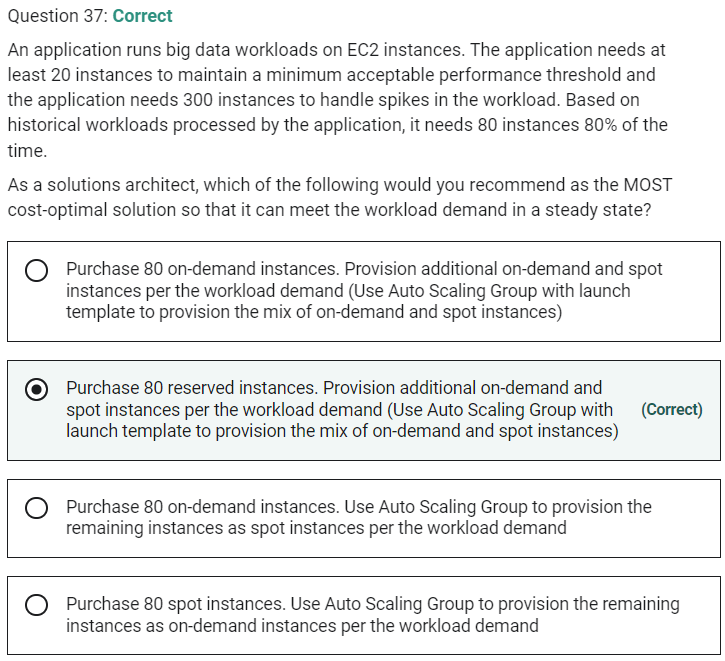
PostgreSQL port = 5432 HTTP port = 80 HTTPS port = 443

The traffic goes like this : The client sends an HTTPS request to ALB on port 443. This is handled by the rule - The security group of the ALB should have an inbound rule from anywhere on port 443. The ALB then forwards the request to one of the EC2 instances. This is handled by the rule - The security group of the EC2 instances should have an inbound rule from the security group of the ALB on port 80. The EC2 instance further accesses the PostgreSQL database managed by RDS on port 5432. This is handled by the rule - The security group of RDS should have an inbound rule from the security group of the EC2 instances in the ASG on port 5432.

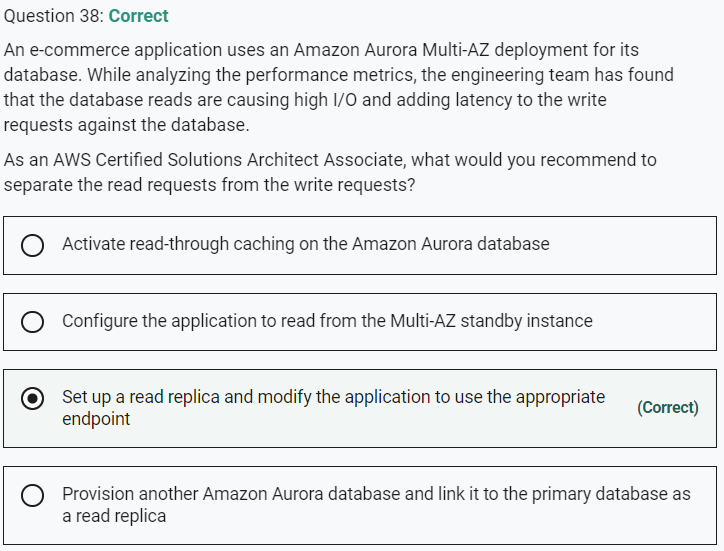


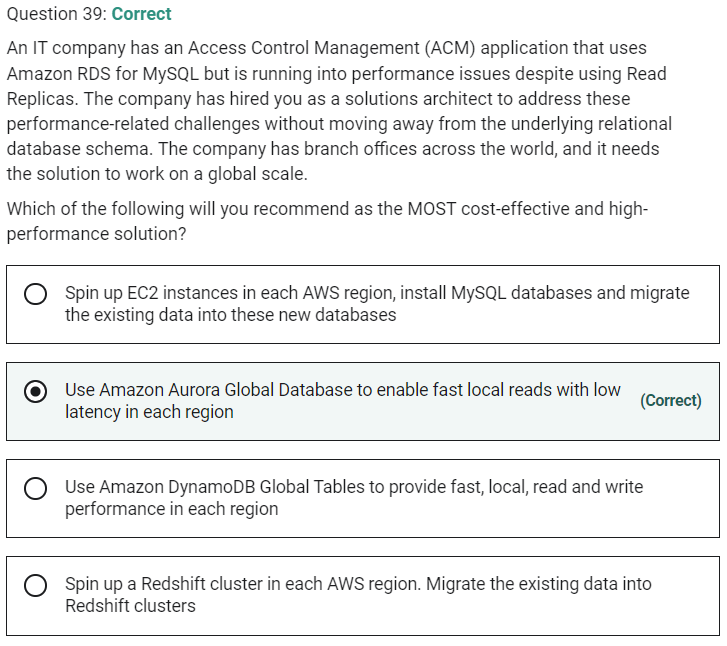


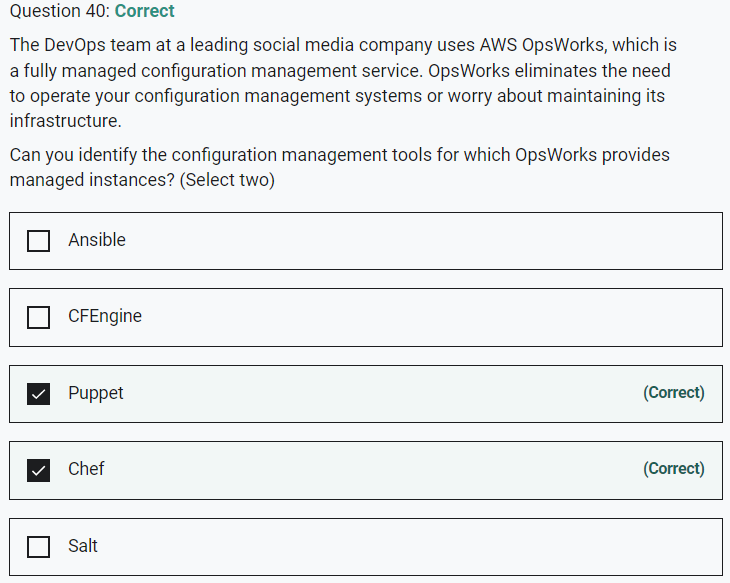


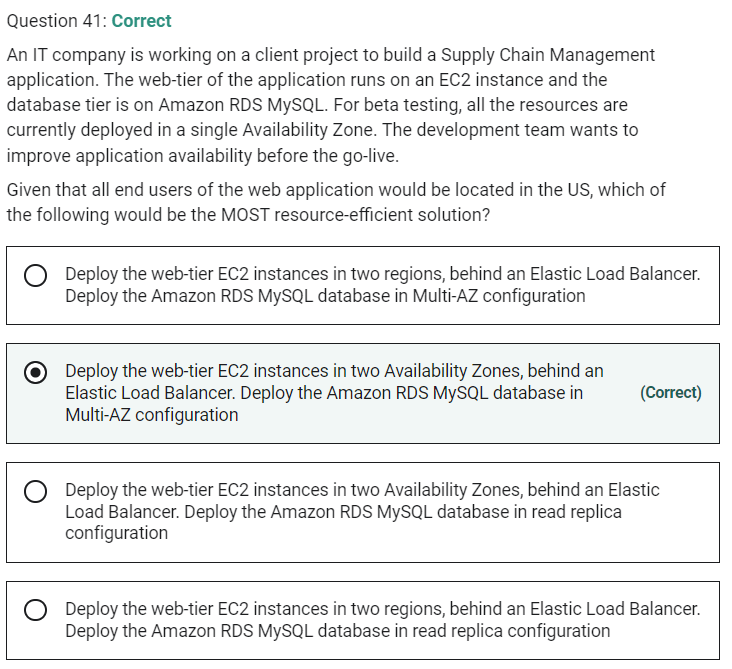


As the steady-state workload demand is 80 instances, we can save on costs by purchasing 80 reserved instances. Based on additional workload demand, we can specify a mix of on-demand and spot instances using Application Load Balancer with a launch template to provision the mix of on-demand and spot instances.

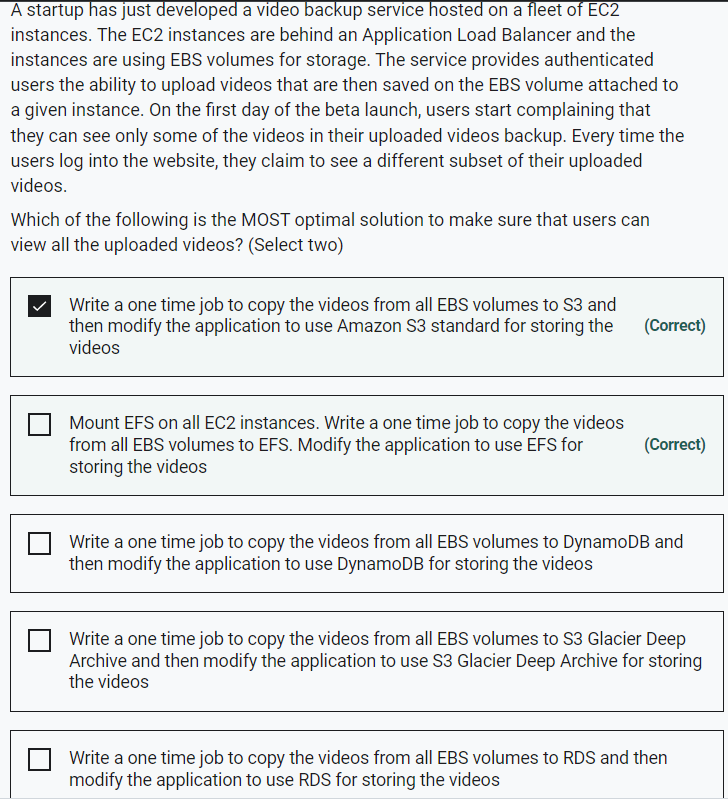








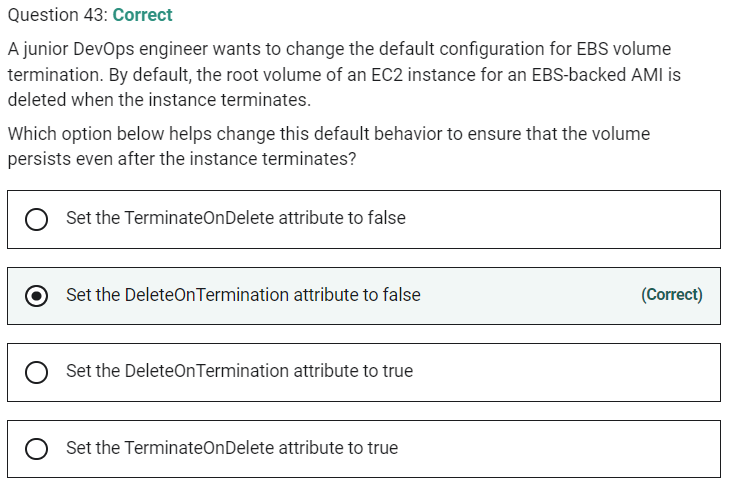
**Deploy the web-tier EC2 instances in two regions, behind an Elastic Load Balancer. Deploy the Amazon RDS MySQL database in Multi-AZ configuration** - As Elastic Load Balancing does not work across regions, so this option is incorrect.

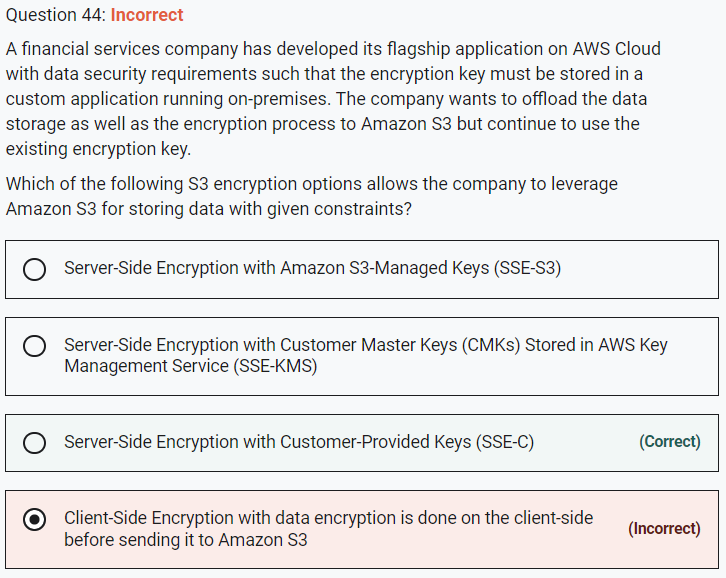


**Write a one time job to copy the videos from all EBS volumes to S3 Glacier Deep Archive and then modify the application to use S3 Glacier Deep Archive for storing the videos** - Glacier Deep Archive is meant to be used for long term data archival. It cannot be used to serve static content such as videos or images via a web application. So this option is incorrect.

**Write a one time job to copy the videos from all EBS volumes to RDS and then modify the application to use RDS for storing the videos** - RDS is a relational database and not the right candidate for storing videos.

**Write a one time job to copy the videos from all EBS volumes to DynamoDB and then modify the application to use DynamoDB for storing the videos** - DynamoDB is a NoSQL database and not the right candidate for storing videos.

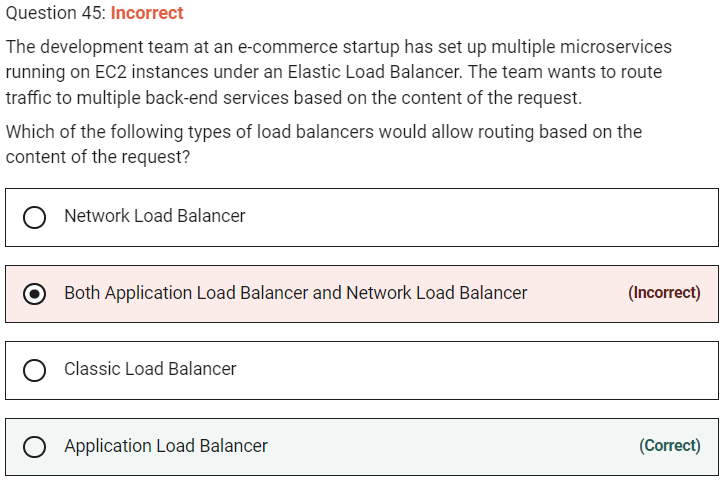


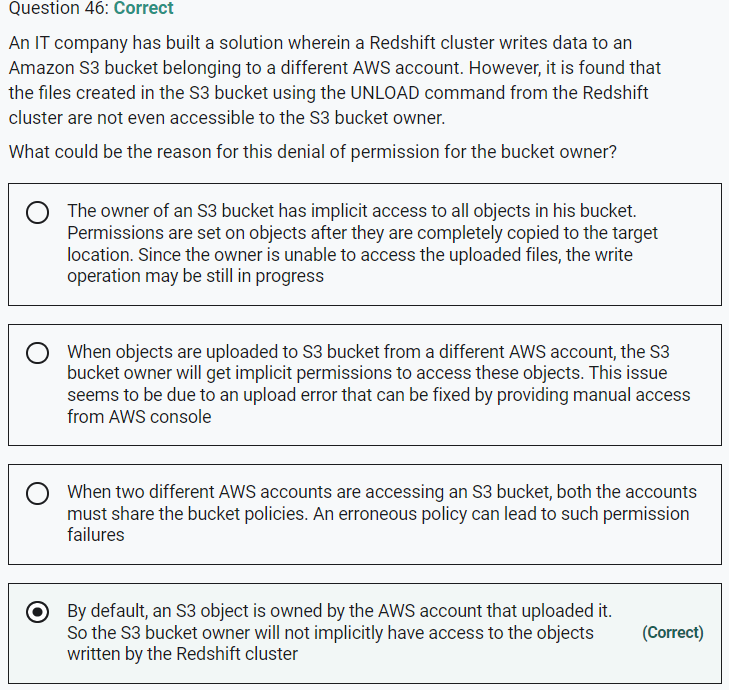


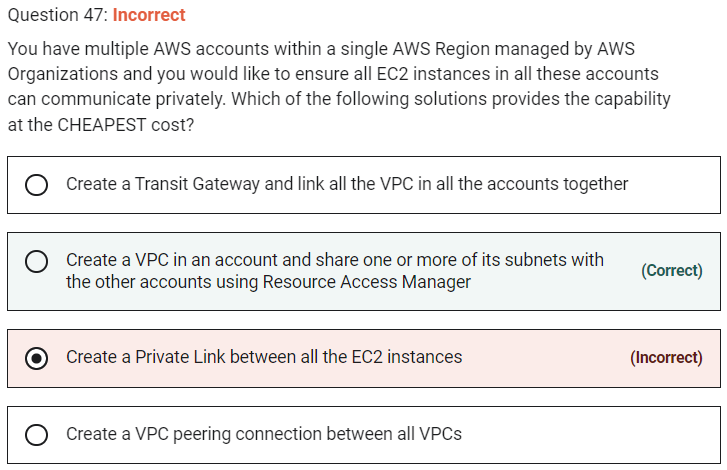
**Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3)** - When you use Server-Side Encryption with Amazon S3-Managed Keys (SSE-S3), each object is encrypted with a unique key. As an additional safeguard, it encrypts the key itself with a master key that it regularly rotates. So this option is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

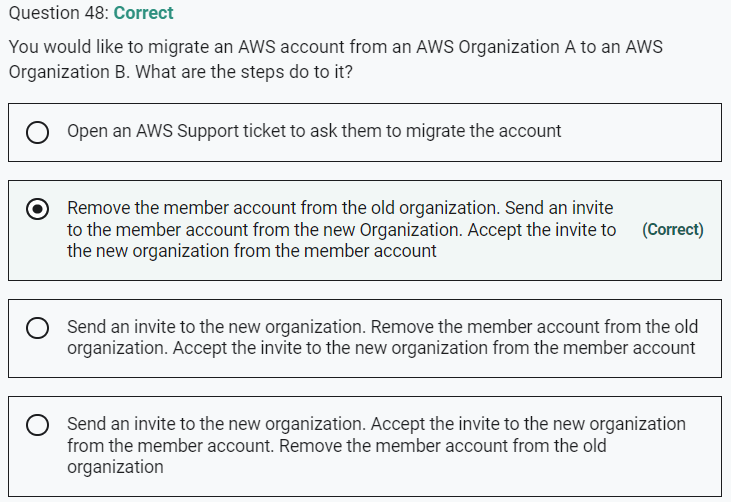
**Server-Side Encryption with Customer Master Keys (CMKs) Stored in AWS Key Management Service (SSE-KMS)** - Server-Side Encryption with Customer Master Keys (CMKs) stored in AWS Key Management Service (SSE-KMS) is similar to SSE-S3. SSE-KMS provides you with an audit trail that shows when your CMK was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

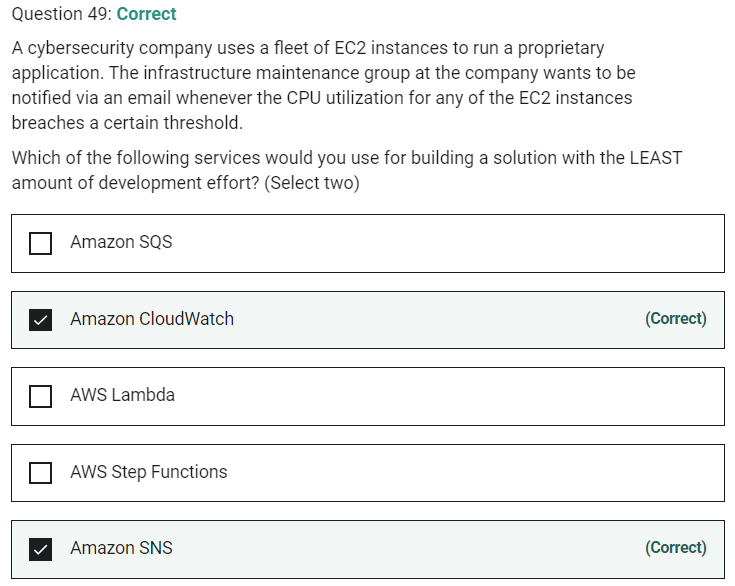
**Client-Side Encryption with data encryption is done on the client-side before sending it to Amazon S3** - You can encrypt the data client-side and upload the encrypted data to Amazon S3. In this case, you manage the encryption process, the encryption keys, and related tools.

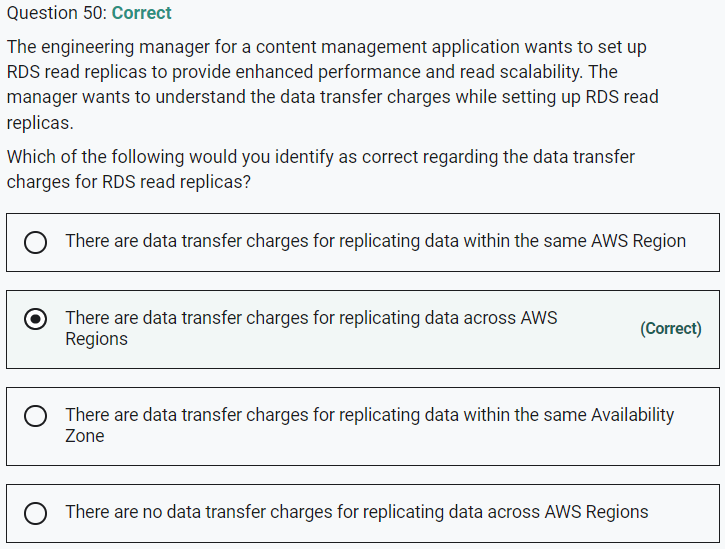


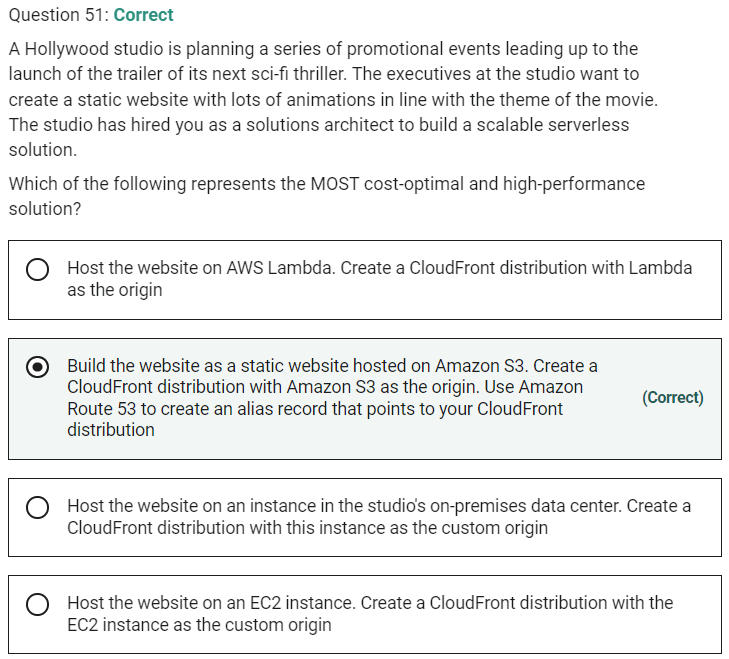


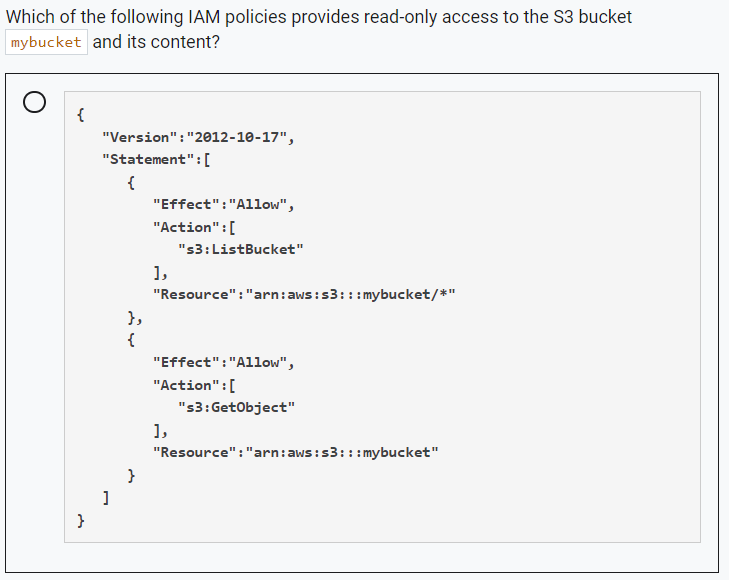






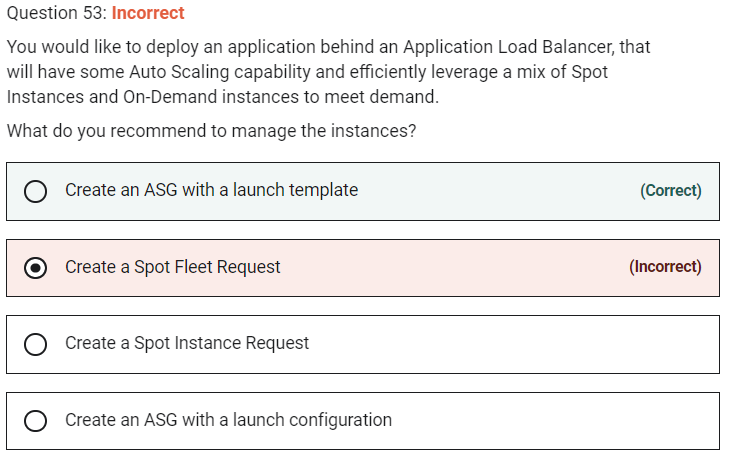




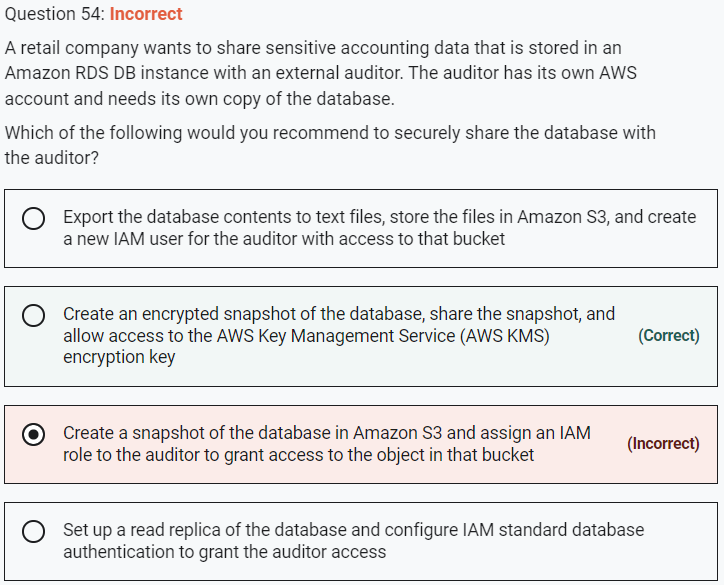








**Create a Spot Fleet Request** - Spot Fleet requests will help launch a mix of On-Demand and Spot, but won't have the auto-scaling capability we need. So this option is incorrect.



**Create an encrypted snapshot of the database, share the snapshot, and allow access to the AWS Key Management Service (AWS KMS) encryption key**

You can share the AWS Key Management Service (AWS KMS) customer master key (CMK) that was used to encrypt the snapshot with any accounts that you want to be able to access the snapshot. You can share AWS KMS CMKs with another AWS account by adding the other account to the AWS KMS key policy.

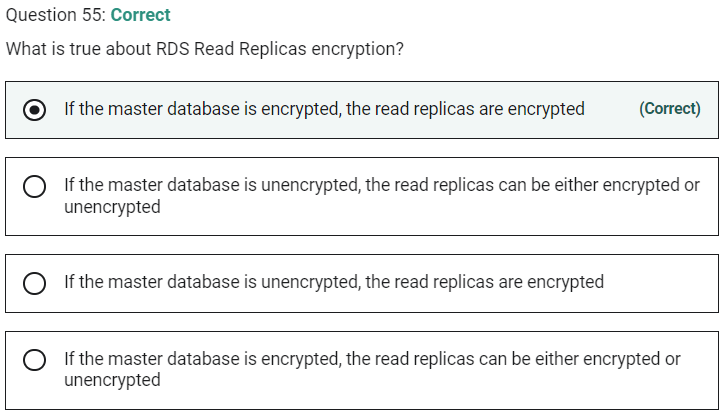
Making an encrypted snapshot of the database will give the auditor a copy of the database, as required for the given use case.

Incorrect options:

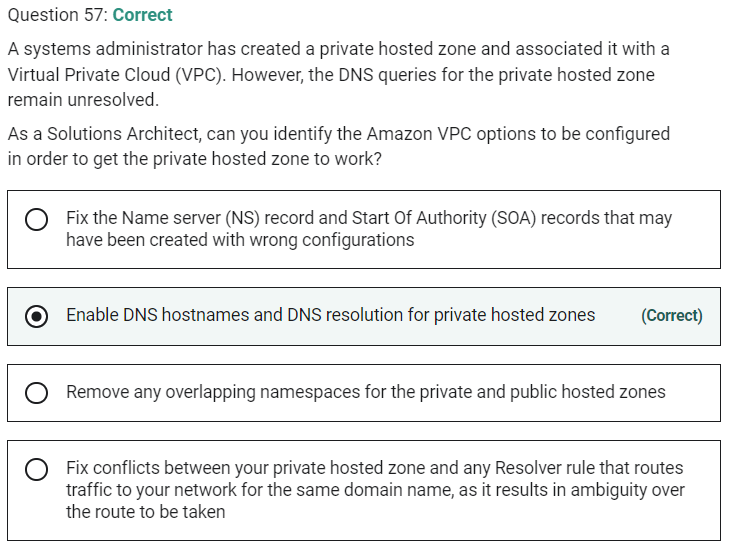
**Create a snapshot of the database in Amazon S3 and assign an IAM role to the auditor to grant access to the object in that bucket** - RDS stores the DB snapshots in the Amazon S3 bucket belonging to the same AWS region where the RDS instance is located. RDS stores these on your behalf and you do not have direct access to these snapshots in S3, so it's not possible to grant access to the snapshot objects in S3.

**Export the database contents to text files, store the files in Amazon S3, and create a new IAM user for the auditor with access to that bucket** - This solution is feasible though not optimal. It requires a lot of unnecessary work and is difficult to audit when such bulk data is exported into text files.

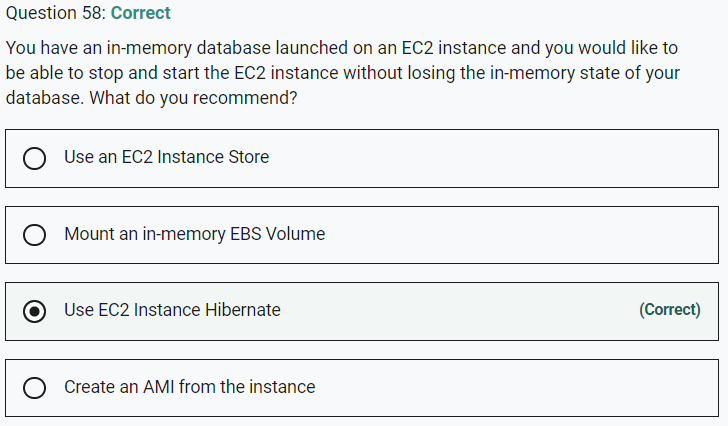
**Set up a read replica of the database and configure IAM standard database authentication to grant the auditor access** - Read Replicas make it easy to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads. Creating Read Replicas for audit purposes is overkill. Also, the question mentions that the auditor needs to have their own copy of the database, which is not possible with replicas.

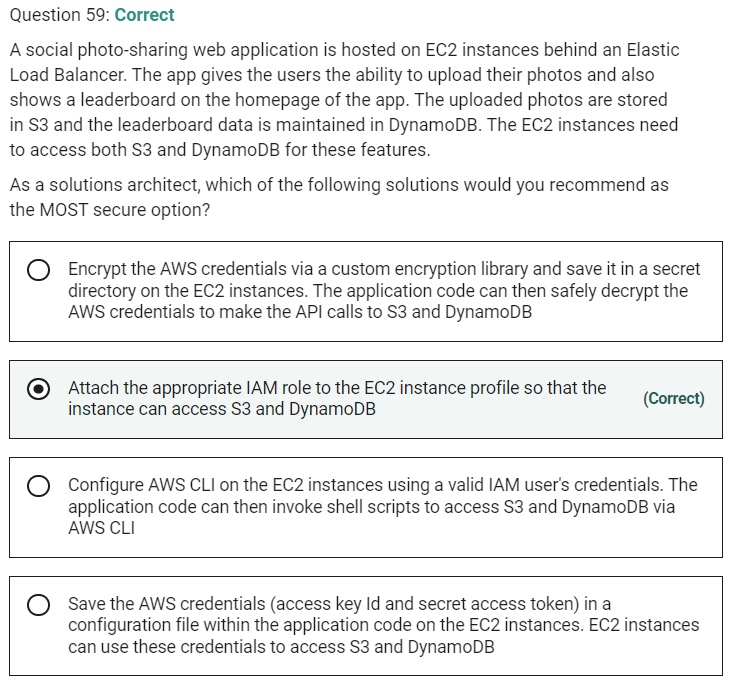


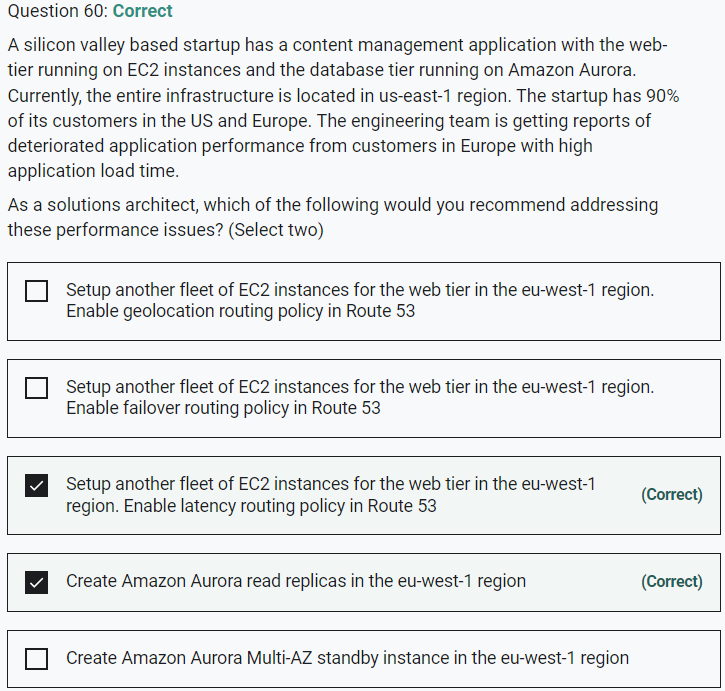


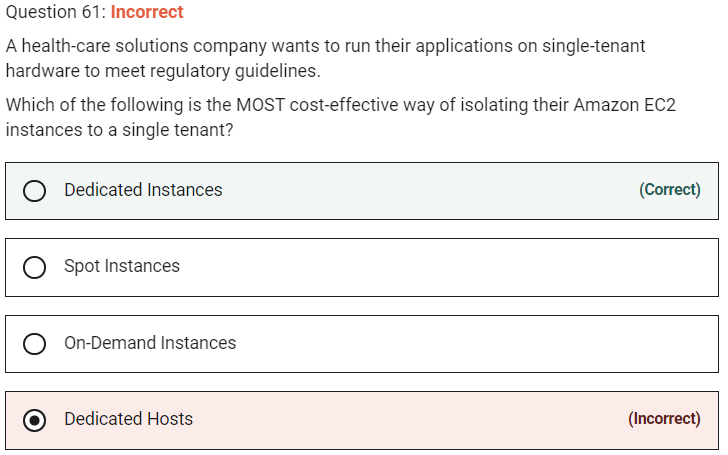


DNS hostnames and DNS resolution are required settings for private hosted zones.

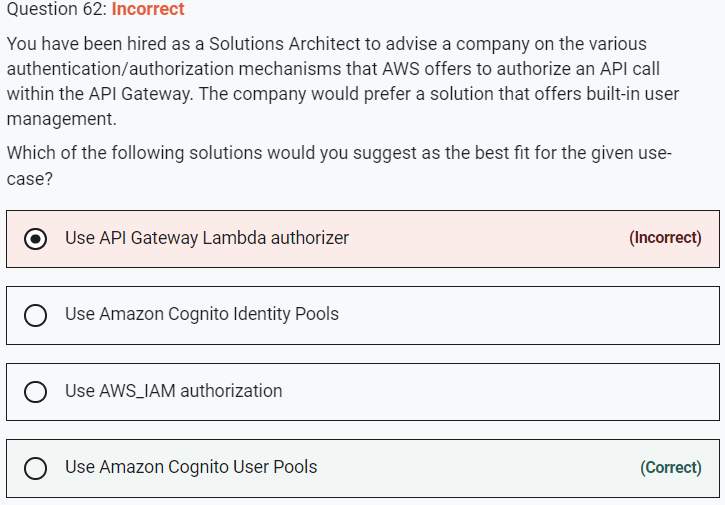








**Dedicated Hosts** - An Amazon EC2 Dedicated Host is a physical server with EC2 instance capacity fully dedicated to your use. Dedicated Hosts allow you to use your existing software licenses on EC2 instances. With a Dedicated Host, you have visibility and control over how instances are placed on the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

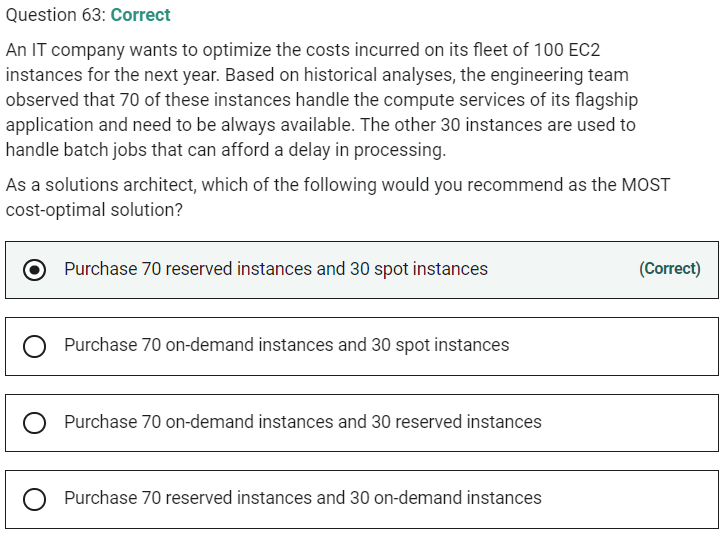


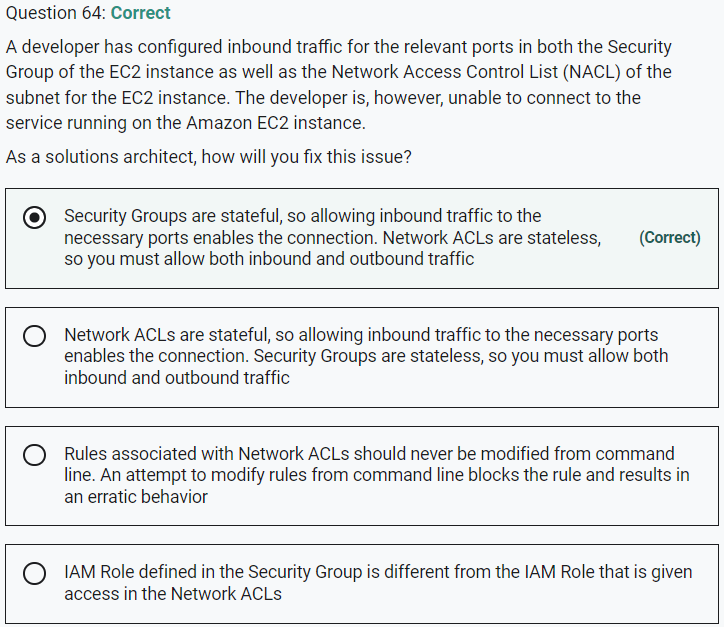
**Use AWS\_IAM authorization**

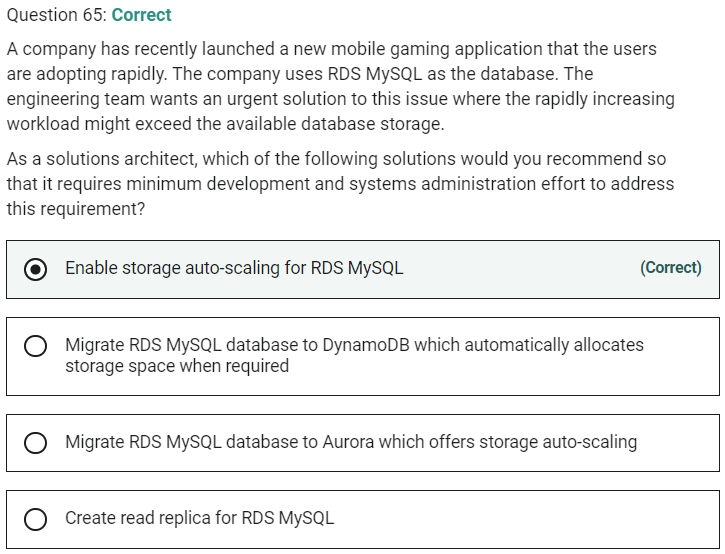
**Use API Gateway Lambda authorizer**

In addition, both these options do not offer built-in user management.

identity pools aren't an authentication mechanism in themselves







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

Free available space is less than 10 percent of the allocated storage.

The low-storage condition lasts at least five minutes.

At least six hours have passed since the last storage modification.