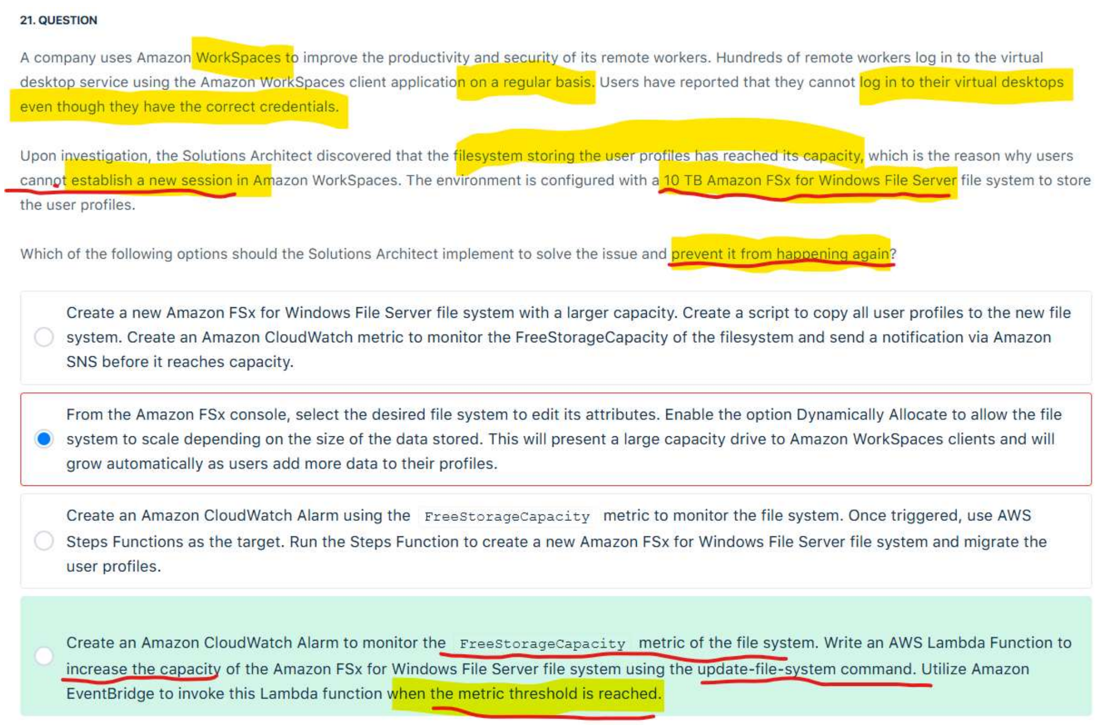
3-simulacoes profissionais

Notebook:	solutions profissional_ef8e1c0f-5d3c-42ab-ab04-9217e53e12cd		
Created:	13/10/2025 01:17	**Updated:**	15/10/2025 01:24
Author:	erikachen19@gmail.com		



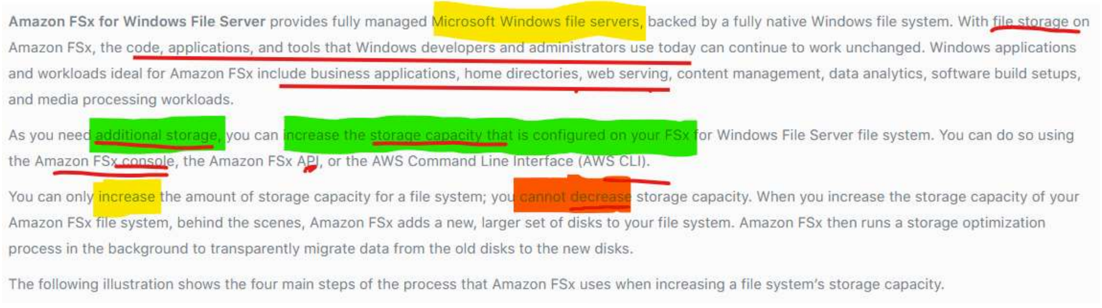
21. QUESTION

A company uses Amazon WorkSpaces to improve the productivity and security of its remote workers. Hundreds of remote workers log in to the virtual desktop service using the Amazon WorkSpaces client application on a regular basis. Users have reported that they cannot log in to their virtual desktops even though they have the correct credentials.

Upon investigation, the Solutions Architect discovered that the filesystem storing the user profiles has reached its capacity, which is the reason why users cannot establish a new session in Amazon WorkSpaces. The environment is configured with a 10 TB Amazon FSx for Windows File Server file system to store the user profiles.

Which of the following options should the Solutions Architect implement to solve the issue and prevent it from happening again?

- Create a new Amazon FSx for Windows File Server file system with a larger capacity. Create a script to copy all user profiles to the new file system. Create an Amazon CloudWatch metric to monitor the FreeStorageCapacity of the filesystem and send a notification via Amazon SNS before it reaches capacity.
- [✓ From the Amazon FSx console, select the desired file system to edit its attributes. Enable the option Dynamically Allocate to allow the file system to scale depending on the size of the data stored. This will present a large capacity drive to Amazon WorkSpaces clients and will grow automatically as users add more data to their profiles.]
- Create an Amazon CloudWatch Alarm using the `FreeStorageCapacity` metric to monitor the file system. Once triggered, use AWS Steps Functions as the target. Run the Steps Function to create a new Amazon FSx for Windows File Server file system and migrate the user profiles.
- Create an Amazon CloudWatch Alarm to monitor the `FreeStorageCapacity` metric of the file system. Write an AWS Lambda Function to increase the capacity of the Amazon FSx for Windows File Server file system using the update-file-system command. Utilize Amazon EventBridge to invoke this Lambda function when the metric threshold is reached.

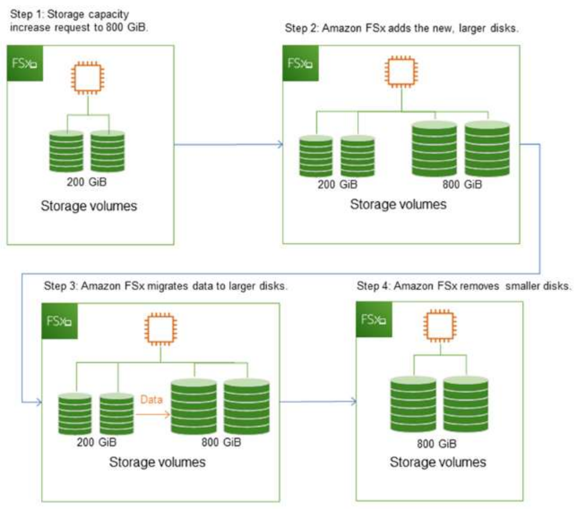


Amazon FSx for Windows File Server provides fully managed Microsoft Windows file servers, backed by a fully native Windows file system. With file storage on Amazon FSx, the code, applications, and tools that Windows developers and administrators use today can continue to work unchanged. Windows applications and workloads ideal for Amazon FSx include business applications, home directories, web serving, content management, data analytics, software build setups, and media processing workloads.

As you need additional storage, you can increase the storage capacity that is configured on your FSx for Windows File Server file system. You can do so using the Amazon FSx console, the Amazon FSx API, or the AWS Command Line Interface (AWS CLI).

You can only increase the amount of storage capacity for a file system; you cannot decrease storage capacity. When you increase the storage capacity of your Amazon FSx file system, behind the scenes, Amazon FSx adds a new, larger set of disks to your file system. Amazon FSx then runs a storage optimization process in the background to transparently migrate data from the old disks to the new disks.

The following illustration shows the four main steps of the process that Amazon FSx uses when increasing a file system's storage capacity.


Step 1: Storage capacity increase request to 800 GiB.
200 GiB
Storage volumes
Step 2: Amazon FSx adds the new, larger disks.
200 GiB
800 GiB
Storage volumes
Step 3: Amazon FSx migrates data to larger disks.
Data
200 GiB
800 GiB
Storage volumes
Step 4: Amazon FSx removes smaller disks.
800 GiB
Storage volumes

EventBridge (CloudWatch Events) helps you to respond to state changes in your AWS resources. With EventBridge (CloudWatch Events), you can create rules that match selected events in the stream and route them to your AWS Lambda function to take action.

Using the update-file-system command, you can use AWS SDK or CLI to programmatically increase the size of the FSx file system. You can use Amazon CloudWatch to monitor the metrics of the file system and trigger the Lambda function to perform an action.

The option that says: **Create a new Amazon FSx for Windows File Server file system with a larger capacity. Create a script to copy all user profiles to the new file system. Create an Amazon CloudWatch metric to monitor the FreeStorageCapacity of the filesystem and send a notification via Amazon SNS before it reaches capacity** is incorrect. You don't have to manually copy all user data to a new volume. You can increase the file system and Amazon FSx automatically migrates the data to a larger volume in the background.

The option that says: **Create an Amazon CloudWatch Alarm using the FreeStorageCapacity metric to monitor the file system. Once triggered, use AWS Steps Functions as the target. Run the Steps Function to create a new Amazon FSx for Windows File Server file system and migrate the user profiles** is incorrect. You don't need to create the Step Function to migrate the user profiles. Amazon FSx automatically migrates the data in the background when you increase the file system size.

The option that says: **From the Amazon FSx console, select the desired file system to edit its attributes. Enable the option Dynamically Allocate to allow the file system to scale depending on the size of the data stored. This will present a large capacity drive to Amazon WorkSpaces clients and will grow automatically as users add more data to their profiles** is incorrect. There is no option to Dynamically Allocate the file system size. You can manually adjust the file system size using the Amazon FSx console, the Amazon FSx API, or the AWS CLI.

22. QUESTION

A company has a hybrid set up for its mobile application. The on-premises data center hosts a 3TB MySQL database server that handles the write-intensive requests from the application. The on-premises network is connected to the AWS VPC with a VPN. On AWS, the serverless application runs on AWS Lambda and API Gateway with an Amazon DynamoDB table used for saving user preferences. The application scales well as more users are using the mobile app. The user traffic is unpredictable but there is an average increase of about 20% each month. A few months into operation, the company noticed the exponential increase of costs for AWS Lambda. The Solutions Architect noticed that the Lambda execution time averages 4.5 minutes and most of that is wait time due to latency when calling the on-premises data MySQL server.

Which of the following solutions should the Solutions Architect implement to reduce the overall cost?

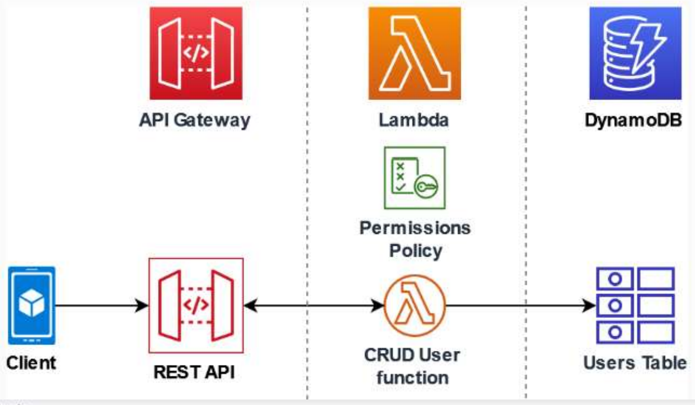
1.Provision an AWS Direct Connect connection from the on-premises data center to Amazon VPC instead of a VPN to significantly reduce the network latency to the MySQL server.
2. Configure caching on the mobile application to reduce the overall AWS Lambda function calls.
3. Gradually lower the timeout and memory properties of the Lambda functions without increasing the execution time.
4. Add an Amazon Elasticache cluster in front of DynamoDB to cache the frequently accessed records.

1. Provision an AWS Direct Connect connection from the on-premises data center to Amazon VPC instead of a VPN to significantly reduce the network latency to the MySQL server.
2. Create a CloudFront distribution with the API Gateway as the origin to cache the API responses and reduce the Lambda invocations.
3. Convert the Lambda functions to run them on Amazon EC2 Reserved Instances. Use Auto Scaling on peak time with a combination of Spot instances to further reduce costs.
4. Configure Auto Scaling on Amazon DynamoDB to automatically adjust the capacity with user traffic.

[✓
1. Migrate the on-premises MySQL database server to Amazon RDS for MySQL. Enable Multi-AZ to ensure high availability.
2. Configure API caching on Amazon API Gateway to reduce the overall number of invocations to the Lambda functions.
3. Gradually lower the timeout and memory properties of the Lambda functions without increasing the execution time.
4. Configure Auto Scaling on Amazon DynamoDB to automatically adjust the capacity based on user traffic.
]

1. Migrate the on-premises MySQL database server to Amazon RDS for MySQL. Enable Multi-AZ to ensure high availability.
2. Create a CloudFront distribution with the API Gateway as the origin to cache the API responses and reduce the Lambda invocations.
3. Gradually lower the timeout and memory properties of the Lambda functions without increasing the execution time.
4. Configure Auto Scaling on Amazon DynamoDB to automatically adjust the capacity with user traffic and enable DynamoDB Accelerator to cache frequently accessed records.

Migrating the on-premises MySQL server to Amazon RDS provides the best latency for the Lambda functions which will significantly reduce the cost for execution time. API Gateway can cache the API request to reduce the Lambda invocation which can reduce the cost further. Auto Scaling for DynamoDB also reduces the cost by provisioning capacity depending on the current user traffic.



The following option is incorrect:

– **Provision an AWS Direct Connect connection from the on-premises data center to Amazon VPC instead of a VPN to significantly reduce the network latency to the MySQL server.**

– **Configure caching on the mobile application to reduce the overall AWS Lambda function calls.**

– **Gradually lower the timeout and memory properties of the Lambda functions without increasing the execution time.**

– **Add an Amazon Elasticache cluster in front of DynamoDB to cache the frequently accessed records.**

Although the Direct Connect connection can reduce the network latency compared to a VPN connection, provisioning a Direct Connection just for a single application is not economical. Having the MySQL server hosted in AWS offers even far better network latency. Provisioning an Elasticache cluster also increases the cost. Caching the API requests should be done on the API Gateway, not on the mobile app itself.

The following option is incorrect:

– **Provision an AWS Direct Connect connection from the on-premises data center to Amazon VPC instead of a VPN to significantly reduce the network latency to the MySQL server.**

– **Create a CloudFront distribution with the API Gateway as the origin to cache the API responses and reduce the Lambda invocations.**

– **Convert the Lambda functions to run them on Amazon EC2 Reserved Instances. Use Auto Scaling on peak time with a combination of Spot instances to further reduce costs.**

– **Configure Auto Scaling on Amazon DynamoDB to automatically adjust the capacity with user traffic.**

Provisioning a Direct Connection just for the application is not economical even if it offers better latency than a VPN connection. Caching the API requests should be done on the API Gateway, and not on CloudFront. EC2 Reserve instances could be more expensive than Lambda functions when application traffic is low.

The following option is incorrect:

– **Migrate the on-premises MySQL database server to Amazon RDS for MySQL. Enable Multi-AZ to ensure high availability.**

– **Create a CloudFront distribution with the API Gateway as the origin to cache the API responses and reduce the Lambda invocations.**

– **Gradually lower the timeout and memory properties of the Lambda functions without increasing the execution time.**

– **Configure Auto Scaling on Amazon DynamoDB to automatically adjust the capacity with user traffic and enable DynamoDB Accelerator to cache frequently accessed records.**

Caching the API requests should be done on the API Gateway, and not on CloudFront. DynamoDB Accelerator is used for caching requests if you need response times in microseconds. This is very expensive.

23. QUESTION

A company is implementing cloud best practices for its infrastructure. The Solutions Architect is using AWS CloudFormation templates for infrastructure-as-code of its two-tier web application. The application frontend is hosted on an Auto Scaling group of Amazon EC2 instances while the database is an Amazon RDS for MySQL instance. For security purposes, the database password must be rotated every 60 days.

Which of the following solutions is the MOST secure way to store and retrieve the database password for the web application?

- On the CloudFormation template, create an encrypted parameter using AWS Systems Manager Parameter Store for the database password. Add a `UserData` property to reference the encrypted parameter in the initialization script of the Auto Scaling group's launch template. Use the `Fn::GetAtt` intrinsic function to reference the encrypted parameter as the value of the `MasterUserPassword` property in the `AWS::RDS::DBInstance` resource.
- [✓ On the CloudFormation template, create an AWS Secrets Manager secret resource for the database password. Modify the application to retrieve the database password from Secrets Manager when it launches. Use a dynamic reference for the secret resource to be placed as the value of the `MasterUserPassword` property of the `AWS::RDS::DBInstance` resource.]
- On the CloudFormation template, create a database password parameter. Add a `UserData` property to reference the password parameter in the initialization script of the Auto Scaling group's launch template using the `Ref` intrinsic function. Save the password inside the EC2 instance upon its launch. Use the `Ref` intrinsic function to reference the parameter as the value of the `MasterUserPassword` property in the `AWS::RDS::DBInstance` resource.
- On the CloudFormation template, create an AWS Secrets Manager secret resource for the database password. Add a `UserData` property to reference the secret resource in the initialization script of the Auto Scaling group's launch template using the `Ref` intrinsic function. Use the `Ref` intrinsic function to reference the secret resource as the value of the `MasterUserPassword` property in the `AWS::RDS::DBInstance` resource.

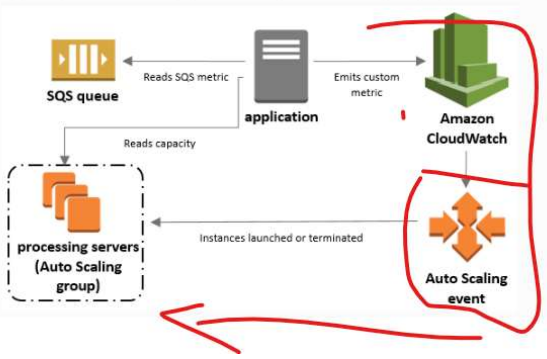
24. QUESTION

A media company in South Korea offers high-quality wildlife photos to its clients. Its photographers upload a large number of photographs to the company's Amazon S3 bucket. Currently, the company is using a dedicated group of on-premises servers to process the photos and uses an open-source messaging system to deliver job information to the servers. After processing, the data would go to a tape library and be stored for long-term archival. The company decided to shift everything to AWS Cloud, and the solutions architect was tasked to implement the same existing infrastructure design and leverage AWS tools such as storage and messaging services to minimize cost.

Which of the following options is the recommended solution that will meet the requirement?

- Initially change the storage class of the S3 objects to S3 IA-Standard. Then create an Auto-scaling group of spot instance workers that scale according to the queue depth in SQS to process job messages. After the data has been processed, transfer your S3 objects to Amazon S3-IA.
- SQS will handle the job messages, while CloudWatch alarms will terminate any idle EC2 worker instances. After the data has been processed, change the storage class of your S3 objects to S3 IA-Standard.
- [✓ Create an Auto-scaling group of spot instance workers that scale according to the queue depth in SQS to process job messages. After the data has been processed, transfer your S3 objects to Amazon Glacier.]
- SNS will handle the passing of job messages, while CloudWatch alarms will terminate any idle spot worker instances. After the data has been processed, transfer your S3 objects to Amazon Glacier.

There are some scenarios where you might think about scaling in response to activity in an **Amazon SQS queue**. For example, suppose that you have a web app that lets users upload images and use them online. In this scenario, each image requires resizing and encoding before it can be published. The app runs on EC2 instances in an Auto Scaling group, and it's configured to handle your typical upload rates. Unhealthy instances are terminated and replaced to maintain current instance levels at all times. The app places the raw bitmap data of the images in an SQS queue for processing. It processes the images and then publishes the processed images where they can be viewed by users. The architecture for this scenario works well if the number of image uploads doesn't vary over time. But if the number of uploads changes over time, you might consider using dynamic scaling to scale the capacity of your Auto Scaling group.



25. QUESTION

A company has deployed a multi-tier web application on AWS that uses Compute Optimized Instances for server-side processing and Storage Optimized EC2 Instances to store various media files. To ensure data durability, there is a scheduled job that replicates the files to each EC2 instance. The current architecture worked for a few months but it started to fail as the number of files grew, which is why the management decided to redesign the system.

Which of the following options should the solutions architect implement in order to launch a new architecture with improved data durability and cost-efficiency?

- Migrate all media files to Amazon EFS then attach this new drive as a mount point to a new set of Storage Optimized EC2 Instances. For the web servers, set up an Elastic Load Balancer with an Auto Scaling of EC2 instances and use this as the origin for a new Amazon CloudFront web distribution. Use a combination of Cost Explorer and AWS Trusted advisor checks to monitor the operating costs and identify potential savings.
- [✓ Migrate all media files to an Amazon S3 bucket and use this as the origin for the new CloudFront web distribution. Set up an Elastic Load Balancer with an Auto Scaling of EC2 instances to host the web servers. Use a combination of Cost Explorer and AWS Trusted advisor checks to monitor the operating costs and identify potential savings.]
- Migrate the web application to AWS Elastic Beanstalk and move all media files to Amazon EFS for a durable and scalable storage. Set up an Amazon CloudFront distribution with EFS as the origin. Use a combination of Consolidated Billing and AWS Trusted advisor checks to monitor the operating costs and identify potential savings.
- Migrate and host the entire web application to Amazon S3 for a more cost-effective web hosting. Enable cross-region replication to improve data durability. Use a combination of Consolidated Billing and AWS Trusted advisor checks to monitor the operating costs and identify potential savings.

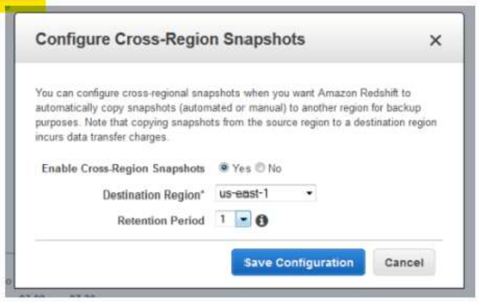
26. QUESTION

A major telecommunications company is planning to set up a disaster recovery solution for its Amazon Redshift cluster which is being used by its online data analytics application. Database encryption is enabled on their clusters using AWS KMS and it is required that the recovery site should be at least 500 miles from their primary cloud location.

Which of the following is the most suitable solution to meet these requirements and to make its architecture highly available?

- Develop a scheduled job using AWS Lambda which will regularly take a snapshot of the Redshift cluster and copy it to another region.
- In your Redshift cluster, enable the cross-region snapshot copy feature to copy snapshots to another region.
- [✓ Set up a `snapshot copy grant` for a master key in the destination region and enable cross-region snapshots in your Redshift cluster to copy snapshots of the cluster to another region.]
- Create a new AWS CloudFormation stack that will deploy the cluster in another region and will regularly back up the data to an S3 bucket, configured with cross-region replication. In case of an outage in the primary region, just use the snapshot from the S3 bucket and then start the cluster.

Snapshots are point-in-time backups of a cluster. There are two types of snapshots: automated and manual. Amazon Redshift stores these snapshots internally in Amazon S3 by using an encrypted Secure Sockets Layer (SSL) connection. Amazon Redshift automatically takes incremental snapshots that track changes to the cluster since the previous automated snapshot.



Configure Cross-Region Snapshots ×

You can configure cross-regional snapshots when you want Amazon Redshift to automatically copy snapshots (automated or manual) to another region for backup purposes. Note that copying snapshots from the source region to a destination region incurs data transfer charges.

Enable Cross-Region Snapshots: [✓ Yes] No
Destination Region*: us-east-1
Retention Period: 1

Automated snapshots retain all of the data required to restore a cluster from a snapshot. You can take a manual snapshot any time. When you restore from a snapshot, Amazon Redshift creates a new cluster and makes the new cluster available before all of the data is loaded, so you can begin querying the new cluster immediately. The cluster streams data on demand from the snapshot in response to active queries, then loads the remaining data in the background.

When you launch an **Amazon Redshift** cluster, you can choose to encrypt it with a master key from the AWS Key Management Service (AWS KMS). AWS KMS keys are specific to a region. If you want to enable cross-region snapshot copy for an AWS KMS-encrypted cluster, you must configure a *snapshot copy grant* for a master key in the destination region so that Amazon Redshift can perform encryption operations in the destination region.

27. QUESTION

A company is running a stateless application on Amazon ECS with AWS Fargate. The Fargate cluster runs behind an Application Load Balancer (ALB) that serves internet traffic. The Amazon Aurora MySQL database cluster is on a private subnet in the same VPC. To connect to the database, the database hostname is passed to the application using the DB_HOST environment variable. The database password is encrypted in AWS Secrets Manager and is passed to the application using the DB_PASSWORD environment variable. Upon running the Fargate tasks, the application was unable to access the database.

Which of the following actions should be taken in order to resolve this issue? (Select THREE.)

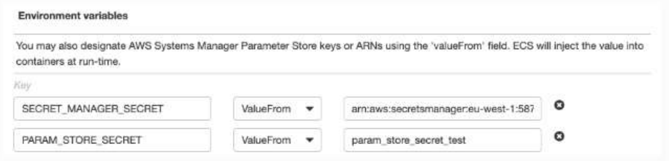
- [✓ Configure the Aurora MySQL database security group to allow inbound traffic from the AWS Fargate cluster on the TCP port 3306.] ✓
- [✓ Ensure that the "DB_HOST" environment variable is configured with the hostname of the Aurora database instance endpoint.]
- [✓ Set the "DB_PASSWORD" environment variable as "ValueFrom" with the secret name from AWS Secrets Manager.]
- Ensure that the "DB_HOST" environment variable is configured with the hostname of the Aurora database cluster endpoint.
- [✓ Set the "DB_PASSWORD" environment variable as "ValueFrom" with the ARN of the secret from AWS Secrets Manager.]
- [✓ Configure the AWS Fargate cluster security group to allow inbound traffic from the Amazon Aurora MySQL cluster on the TCP port 3306.]

Amazon Aurora typically involves a cluster of DB instances instead of a single instance. Each connection is handled by a specific DB instance. When you connect to an Aurora cluster, the host name and port that you specify point to an intermediate handler called an endpoint.

Each Aurora DB cluster has one cluster endpoint and one primary DB instance. A **cluster endpoint** (or writer endpoint) for an Aurora DB cluster connects to the current primary DB instance for that DB cluster. This endpoint is the only one that can perform write operations such as DDL statements. Because of this, the cluster endpoint is the one that you connect to when you first set up a cluster or when your cluster only contains a single DB instance.

An **instance endpoint** connects to a specific DB instance within an Aurora cluster. Each DB instance in a DB cluster has its own unique instance endpoint. So there is one instance endpoint for the current primary DB instance of the DB cluster, and there is one instance endpoint for each of the Aurora Replicas in the DB cluster.

Amazon ECS enables you to inject sensitive data into your containers by storing your sensitive data in **AWS Secrets Manager** secrets and then referencing them in your container definition. Sensitive data stored in Secrets Manager secrets can be exposed to a container as environment variables or as part of the log configuration.



Environment variables

You may also designate AWS Systems Manager Parameter Store keys or ARNs using the 'valueFrom' field. ECS will inject the value into containers at run-time.

Key		
SECRET_MANAGER_SECRET	ValueFrom	arn:aws:secretsmanager:eu-west-1:587
PARAM_STORE_SECRET	ValueFrom	param_store_secret_test

When you inject a secret as an environment variable, you can specify the full contents of a secret, a specific JSON key within a secret, or a specific version of a secret to inject.

Within your container definition, you can specify the following:

-The secrets object containing the name of the environment variable to set in the container.

-The Amazon Resource Name (ARN) of the Secrets Manager secret.

-Additional parameters that contain the sensitive data to present to the container.

The option that says: **Set the "DB_PASSWORD" environment variable as "ValueFrom" with the ARN of the secret from AWS Secrets Manager** is correct. **The environment variable should contain the ARN from AWS Secrets Manager.**

The option that says: **Configure the Aurora MySQL database security group to allow inbound traffic from the AWS Fargate cluster on the TCP port 3306** is correct. **You need to ensure that the Aurora database is open to accept connections from the Fargate cluster.**

The option that says: **Ensure that the "DB_HOST" environment variable is configured with the hostname of the Aurora database cluster endpoint** is correct. **The cluster endpoint should be used as "Value" of the environment variable. This is the main endpoint that accepts read/write connections.**

The option that says: **Set the "DB_PASSWORD" environment variable as "ValueFrom" with the secret name from AWS Secrets Manager** is incorrect. **You should use the ARN of the secret from the Secrets Manager, not the secret name.**

The option that says: **Ensure that the "DB_HOST" environment variable is configured with the hostname of the Aurora database instance endpoint** is incorrect. **Amazon Aurora has different instance endpoints for write and reader nodes. It is recommended to use the cluster endpoint here because this is the one that can perform both read/write operations.**

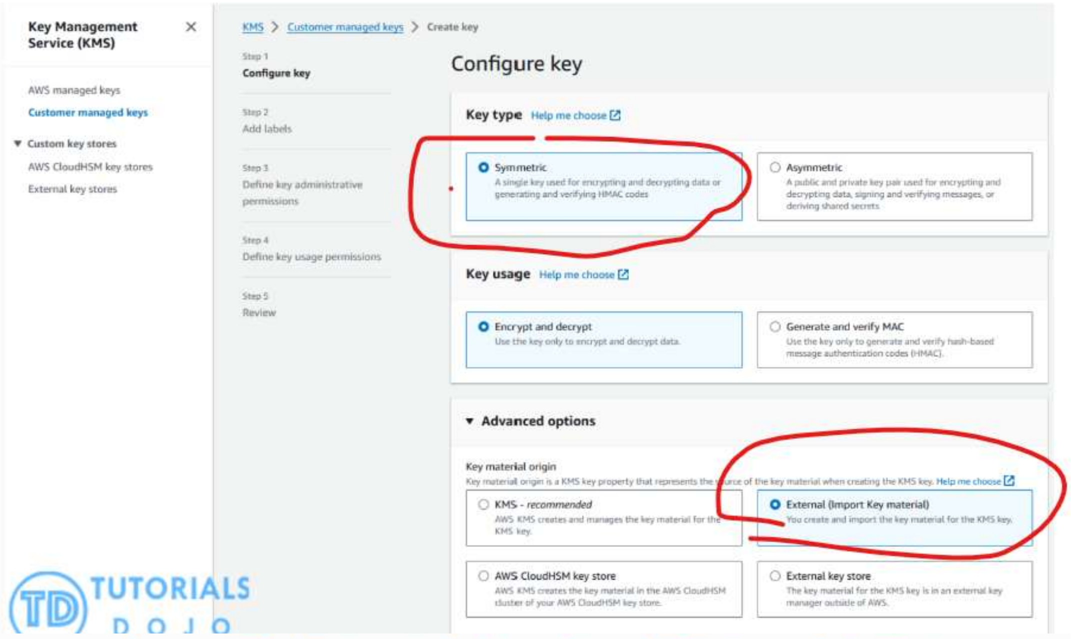
The option that says: **Configure the AWS Fargate cluster security group to allow inbound traffic from the Amazon Aurora MySQL cluster on the TCP port 3306** is incorrect. **The AWS Fargate cluster will connect to the database, therefore, the database must allow connections from the Aurora cluster.**

28. QUESTION

A financial services company uses hardware security modules (HSMs) to generate encryption master keys. Since the company application logs include personally identifiable information, encryption is required as part of regulatory compliance. The application logs are going to be stored on a central Amazon S3 bucket and should be encrypted at rest. The security team wants to use the company HSMs to generate the key material for encryption on the S3 bucket.

Which of the following options should the solutions architect implement to meet the company's requirements?

- [✓] Using AWS CLI, create a new KMS key with no key material and use EXTERNAL as the origin of the key. Generate a key from the on-premises HSMs and import it as KMS key using the public key and import token from AWS. Apply an Amazon S3 bucket policy on the central logging bucket to require AWS KMS as the encryption source and deny unencrypted object uploads.
- [ ] Request to provision an AWS Direct Connect connection from the on-premises data center to AWS VPC. Ensure that the network addresses do not overlap. Apply an Amazon S3 bucket policy on the central logging bucket to allow only encrypted object uploads. Configure the application to generate a unique KMS key for each logging event by querying the on-premises HSMs through the Direct Connection network.
- [ ] Using AWS CLI, create a new KMS key with AWS-provided key material and use AWS_KMS as the origin of the key. Overwrite this KMS key with a generated key from the on-premises HSMs by using the public key and import token provided by AWS. Set a 1-year duration for the KMS key automatic key rotation. Apply an Amazon S3 bucket policy on the central logging bucket to require AWS KMS as the encryption source and deny unencrypted object uploads.
- [ ] Create a new AWS CloudHSM cluster and set it as the key material source in AWS Key Management Service (KMS) when you generate a new KMS key. Set a 1-year duration for the KMS key automatic key rotation. Apply an Amazon S3 bucket policy on the central logging bucket to require AWS KMS as the encryption source and deny unencrypted object uploads.



Key Management Service (KMS)

AWS managed keys
Customer managed keys
▼ Custom key stores
AWS CloudHSM key stores
External key stores

KMS > Customer managed keys > Create key

Step 1 Configure key
Step 2 Add labels
Step 3 Define key administrative permissions
Step 4 Define key usage permissions
Step 5 Review

Configure key

Key type Help me choose

● Symmetric — A single key used for encrypting and decrypting data or generating and verifying HMAC codes
○ Asymmetric — A public and private key pair used for encrypting and decrypting data, signing and verifying messages, or deriving shared secrets

Key usage Help me choose

● Encrypt and decrypt — Use the key only to encrypt and decrypt data.
○ Generate and verify MAC — Use the key only to generate and verify hash-based message authentication codes (HMAC).

▼ Advanced options

Key material origin
Key material origin is a KMS key property that represents the source of the key material when creating the KMS key. Help me choose

○ KMS - *recommended* — AWS KMS creates and manages the key material for the KMS key.
● External (Import Key material) — You create and import the key material for the KMS key.
○ AWS CloudHSM key store — AWS KMS creates the key material in the AWS CloudHSM cluster of your AWS CloudHSM key store.
○ External key store — The key material for the KMS key is in an external key manager outside of AWS.

TUTORIALS DOJO

The option that says: **Create a new AWS CloudHSM cluster and set it as the key material source in AWS Key Management Service (KMS) when you generate a new KMS key. Set a 1-year duration for the KMS key automatic key rotation. Apply an Amazon S3 bucket policy on the central logging bucket to require AWS KMS as the encryption source and deny unencrypted object uploads** is incorrect because it uses an AWS CloudHSM cluster instead of the company's existing on-premises HSMs to generate the key material.

The option that says: **Request to provision an AWS Direct Connect connection from the on-premises data center to AWS VPC. Ensure that the network addresses do not overlap. Apply an Amazon S3 bucket policy on the central logging bucket to allow only encrypted object uploads. Configure the application to generate a unique KMS key for each logging event by querying the on-premises HSMs through the Direct Connection network** is incorrect because it involves setting up an AWS Direct Connect connection from the on-premises data center to AWS VPC, which may not be necessary if the on-premises HSMs can securely communicate with AWS over the internet. It also requires generating a unique KMS key for each logging event by querying the on-premises HSMs through the Direct Connect network. This approach is not scalable or efficient, as it would result in a large number of KMS keys being created and managed. Lastly, it does not explicitly enforce encryption on the central Amazon S3 bucket using the KMS key with the imported key material from the on-premises HSMs.

The option that says: **Using AWS CLI, create a new KMS key with AWS-provided key material and use AWS_KMS as the origin of the key. Overwrite this KMS key with a generated key from the on-premises HSMs by using the public key and import token provided by AWS. Set a 1-year duration for the KMS key automatic key rotation. Apply an Amazon S3 bucket policy on the central logging bucket to require AWS KMS as the encryption source and deny unencrypted object uploads** is incorrect because it creates a KMS key with AWS-provided key material (origin set to AWS_KMS) and then overwrites this key material with the key generated from the on-premises HSMs. This approach is not recommended, as overwriting an existing key material can lead to security risks and potential data loss if not handled correctly. It is better to create a new KMS key with no key material (origin set to EXTERNAL) and import the key material from the on-premises HSMs.

29. QUESTION

A company is planning to build its new customer relationship management (CRM) portal in AWS. The application architecture will be using a containerized microservices hosted on an Amazon ECS cluster. A Solutions Architect has been tasked to set up the architecture and comply with the AWS security best practice of granting the least privilege. The architecture should also support the use of security groups and standard network monitoring tools at the container level to comply with the company's strict IT security policies.

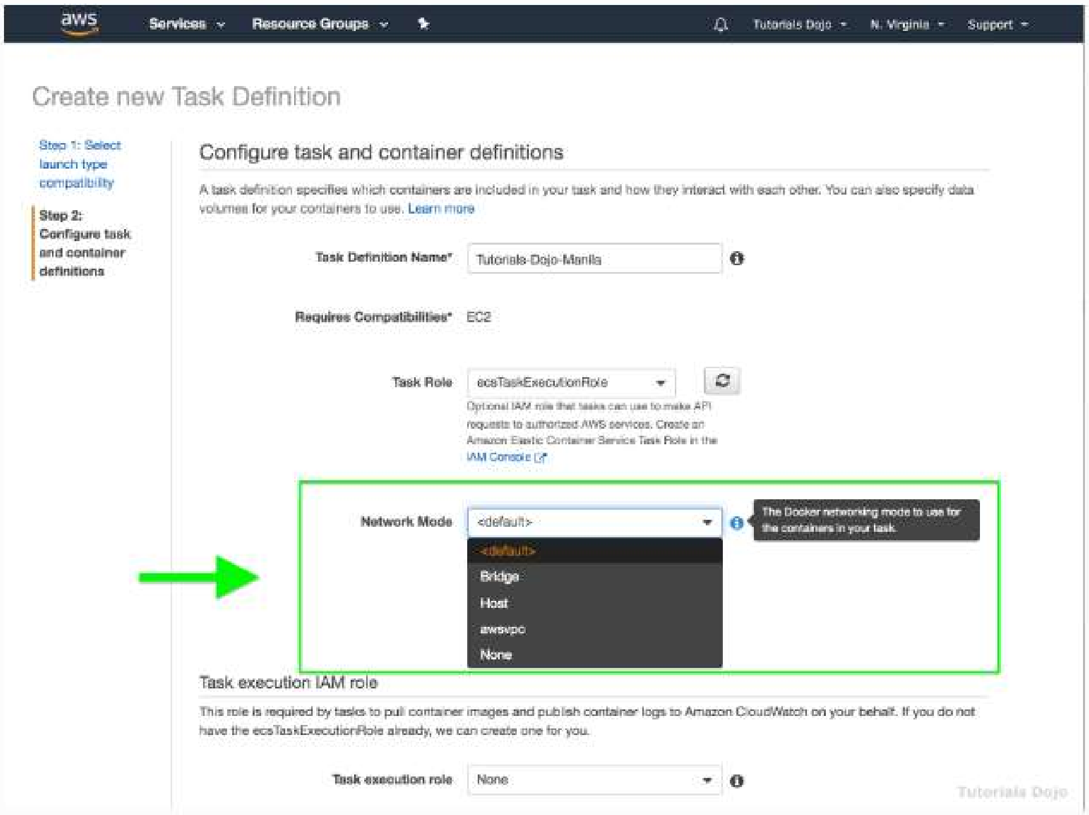
Which of the following provides the MOST secure configuration for the CRM portal?

- [✓ Use the `bridge` network mode in the task definition in your Amazon ECS Cluster. Attach security groups to Amazon EC2 instances then use IAM roles for EC2 instances to access other resources.]
- Use AWS App Runner to run the containerized application instead to improve security and reduce operational overhead. Select VPC and security groups accordingly for deployment. Add IAM credentials to the environment variables when launching the service.
- Use the `awsvpc` network mode in the task definition in your Amazon ECS Cluster. Attach security groups to the ECS tasks then pass IAM credentials into the container at launch time to access other AWS resources.
- Use the `awsvpc` network mode in the task definition in your Amazon ECS Cluster. Attach security groups to the ECS tasks then use IAM roles for tasks to access other resources.

Task definitions are split into separate parts: the task family, the IAM task role, the network mode, container definitions, volumes, task placement constraints, and launch types. The family and container definitions are required in a task definition, while task role, network mode, volumes, task placement constraints, and launch type are optional.

You can configure various Docker networking modes that will be used by containers in your ECS task. The valid values are `none` , `bridge` , `awsvpc` , and `host` . The default Docker network mode is `bridge` .

With IAM roles for Amazon ECS tasks, you can specify an IAM role that can be used by the containers in a task. Applications must sign their AWS API requests with AWS credentials, and this feature provides a strategy for managing credentials for your applications to use, similar to the way that Amazon EC2 instance profiles provide credentials to EC2 instances. Instead of creating and distributing your AWS credentials to the containers or using the EC2 instance's role, you can associate an IAM role with an ECS task definition or `RunTask` API operation. The applications in the task's containers can then use the AWS SDK or CLI to make API requests to authorized AWS services.



Create new Task Definition

Configure task and container definitions

A task definition specifies which containers are included in your task and how they interact with each other. You can also specify data volumes for your containers to use. Learn more

Task Definition Name* Tutorials-Dojo-Manila

Requires Compatibilities* EC2

Task Role ecsTaskExecutionRole

Optional IAM role that tasks can use to make API requests to authorized AWS services. Create an Amazon Elastic Container Service Task Role in the IAM Console

Network Mode <default>

The Docker networking mode to use for the containers in your task.

Task execution IAM role

This role is required by tasks to pull container images and publish container logs to Amazon CloudWatch on your behalf. If you do not have the ecsTaskExecutionRole already, we can create one for you.

Task execution role None

If the network mode is **`host`**, the task bypasses Docker's built-in virtual network and maps container ports directly to the EC2 instance's network interface directly. In this mode, you can't run multiple instantiations of the same task on a single container instance when port mappings are used.

如果网络模式为 host，则任务将绕过 Docker 的内置虚拟网络，并将容器端口直接映射到 EC2 实例的网络接口。在此模式下，使用端口映射时，无法在单个

容器实例上运行同一任务的多个实例。

If the network mode is set to `none`, the task's containers do not have external connectivity, and port mappings can't be specified in the container definition.

If the network mode is `bridge`, the task utilizes Docker's built-in virtual network which runs inside each container instance.

If the network mode is `host`, the task bypasses Docker's built-in virtual network and maps container ports directly to the EC2 instance's network interface directly. In this mode, you can't run multiple instantiations of the same task on a single container instance when port mappings are used.

If the network mode is `awsvpc`, the task is allocated an elastic network interface, and you must specify a `NetworkConfiguration` when you create a service or run a task with the task definition. When you use this network mode in your task definitions, every task that is launched from that task definition gets its own elastic network interface (ENI) and a primary private IP address. The task networking feature simplifies container networking and gives you more control over how containerized applications communicate with each other and other services within your VPCs.

Task networking also provides greater security for your containers by allowing you to use security groups and network monitoring tools at a more granular level within your tasks. Because each task gets its own ENI, you can also take advantage of other Amazon EC2 networking features like VPC Flow Logs so that you can monitor traffic to and from your tasks. Additionally, containers that belong to the same task can communicate over the `localhost` interface. A task can only have one ENI associated with it at a given time.

30. QUESTION

A company recently migrated a legacy application from a data center to AWS. The application, which is hosted on an EC2 instance, uses a Simple Mail Transfer Protocol (SMTP) server for sending promotional emails. The application communicates with the SMTP server over port 25 without TLS encryption. As part of its replatforming strategy, the company has opted to replace the SMTP server with Amazon Simple Email Service (SES). The SES domain is already created and has been validated.

What steps should be taken next to enable communication between the application and Amazon SES?

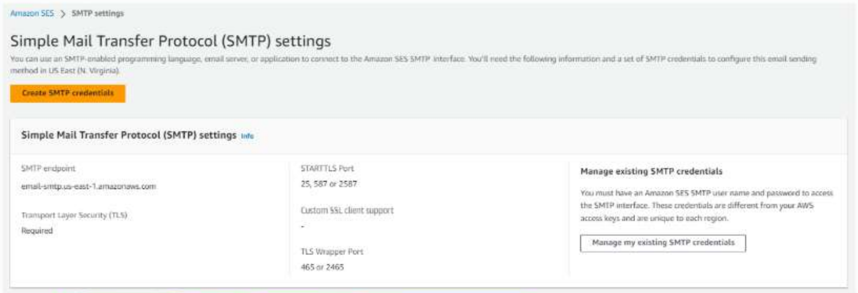
1) 2)

- [✓] Create an SMTP credential through the Amazon SES console. Configure the application to use the SMTP credential and update the connection to the Amazon SES SMTP endpoint. Change the port settings to 587 to use STARTTLS.
- Create an IAM Role for the application with `ses:SendEmail` and `ses:SendRawEmail` permissions and attach it to the EC2 instance. Update the SMTP settings in the application to connect to the Amazon SES SMTP. Update the port number to 465 to use TLS Wrapper.
- Create an SMTP credential through the Amazon SES console. Configure the application to use the SMTP credential. Use the AWS SDK for Amazon SES to send emails.
- Create an IAM user and grant it permission to send emails through Amazon SES. Generate access keys for this IAM user. Update the application to use these access keys and connect to the Amazon SES SMTP endpoint. Change the port to 587 to use STARTTLS.

Correct

Amazon Simple Email Service (Amazon SES) is a scalable and secure cloud-based email-sending service designed to help businesses send marketing, transactional, and other types of professional emails.

The Amazon SES SMTP endpoint requires that all connections be encrypted using Transport Layer Security (TLS). Amazon SES supports two mechanisms for establishing a TLS-encrypted connection: STARTTLS and TLS Wrapper.


Amazon SES > SMTP settings
Create SMTP credentials
Info
Manage my existing SMTP credentials

Simple Mail Transfer Protocol (SMTP) settings

You can use an SMTP-enabled programming language, email server, or application to connect to the Amazon SES SMTP interface. You'll need the following information and a set of SMTP credentials to configure this email sending method in US East (N. Virginia).

Simple Mail Transfer Protocol (SMTP) settings

SMTP endpoint
email-smtp.us-east-1.amazonaws.com

Transport Layer Security (TLS)
Required

STARTTLS Port
25, 587 or 2587

Custom SSL client support
-

TLS Wrapper Port
465 or 2465

Manage existing SMTP credentials

You must have an Amazon SES SMTP user name and password to access the SMTP interface. These credentials are different from your AWS access keys and are unique to each region.

STARTTLS is a means of upgrading an unencrypted connection to an encrypted connection. To set up a STARTTLS connection, the SMTP client connects to the Amazon SES SMTP endpoint on **port 25, 587, or 2587**, issues an EHLO command, and waits for the server to announce that it supports the STARTTLS SMTP extension. The client then issues the STARTTLS command, initiating TLS negotiation. When negotiation is complete, the client issues an EHLO command over the new encrypted connection, and the SMTP session proceeds normally.

Amazon SES supports sending emails through two main interfaces: the Simple Mail Transfer Protocol (SMTP) and the SES API.

In the scenario, since the company's legacy application is already configured to connect to an SMTP server, they can simply replace the existing SMTP settings with those of Amazon SES. This involves generating new SMTP credentials via the Amazon SES console and updating the application to use them. Additionally, it is recommended to switch from the currently used port 25 to port 587 for STARTTLS encryption. This is beneficial not only for enhanced security but also because Amazon EC2 instances are throttled on port 25, and using port 587 avoids the need to request the lifting of these limitations.