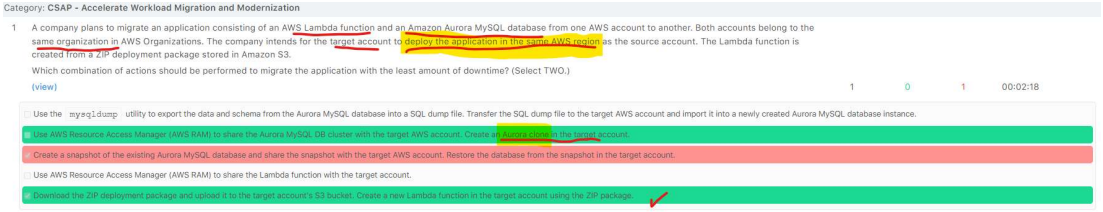
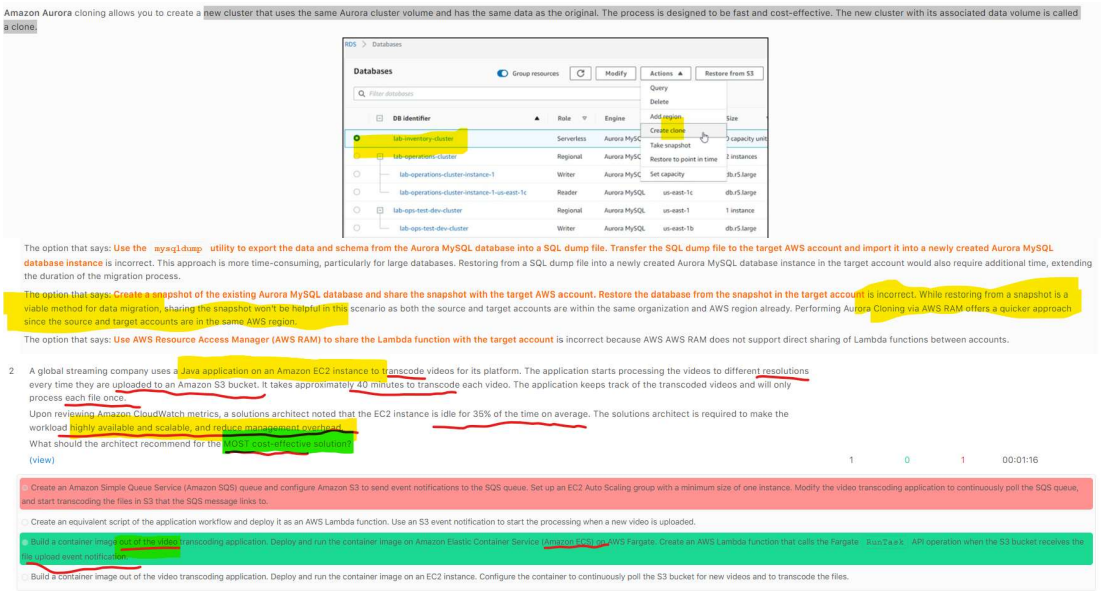
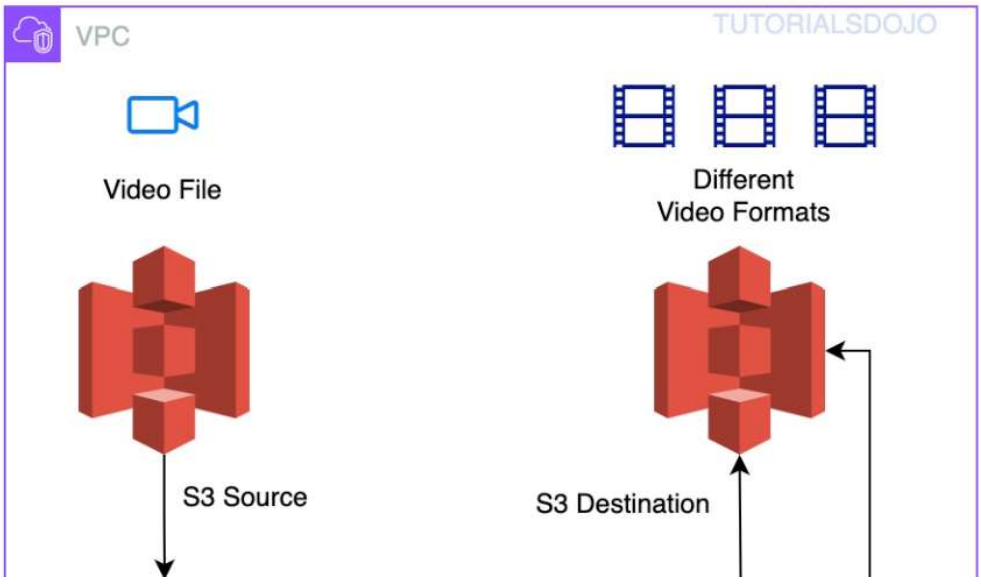
4-simulacoes profissionais

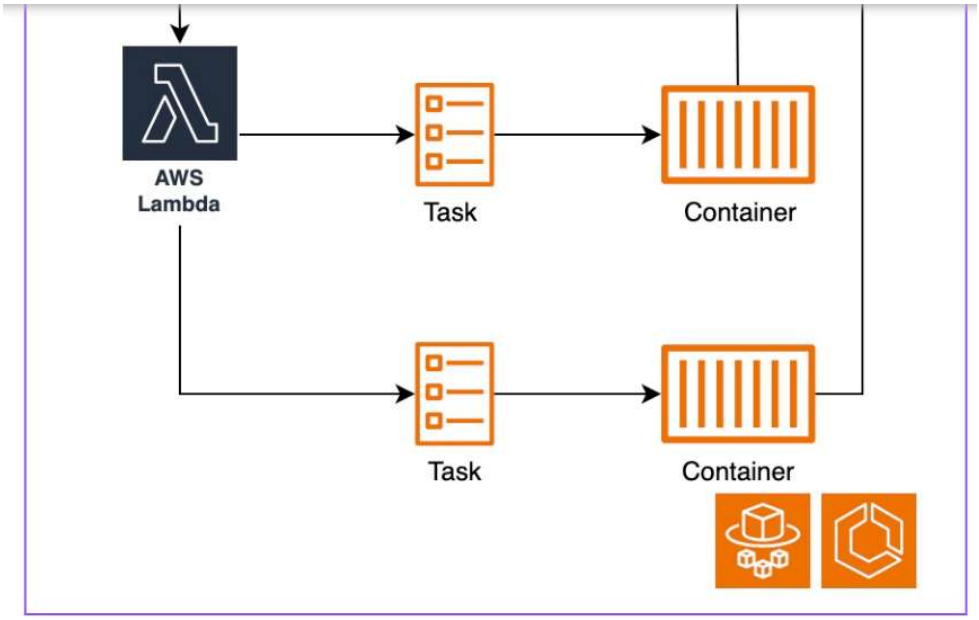
Notebook:	solutions profissional_ef8e1c0f-5d3c-42ab-ab04-9217e53e12cd		
Created:	15/10/2025 01:24	**Updated:**	16/10/2025 01:59
Author:	erikachen19@gmail.com		



The company intends for the target account to deploy the application in the same AWS region as the source account. 该公司希望目标账户在与源账户相同的 AWS 区域部署应用程序。







AWS Fargate is a service that you can use with Amazon ECS to run containers without having to manage servers or clusters of Amazon EC2 instances. With AWS Fargate, you no longer have to provision, configure, or scale clusters of virtual machines to run containers. This removes the need to choose server types, decide when to scale your clusters, or optimize cluster packing. Amazon ECS, using Fargate, allows you to run containers without managing the underlying infrastructure. It automatically scales based on demand.

One use case for ECS task definitions is on video transcoding workloads. To use video transcoding workloads on Amazon ECS, register Amazon EC2 VT1 instances. This will allow running live and pre-rendered video transcoding workloads as tasks on Amazon ECS.

Because of its scaling capability, AWS Fargate can be used to run event-driven workloads at scale. For use cases when a workload cannot be run by AWS Lambda alone because of limitations on runtime, such as Lambda's current 15-minute memory limit, a Lambda function can be triggered and invoke an ECS task.

Using S3 event notifications to trigger the Lambda function ensures that the transcoding process starts as soon as a new video is uploaded, improving efficiency.

Therefore, the correct answer is: **Build a container image out of the video transcoding application. Deploy and run the container image on Amazon Elastic Container Service (Amazon ECS) on AWS Fargate. Create an AWS Lambda function that calls the Fargate `RunTask API` operation when the S3 bucket receives the file upload event notification.**

The option that says: **Create an Amazon Simple Queue Service (Amazon SQS) queue and configure Amazon S3 to send event notifications to the SQS queue. Set up an EC2 Auto Scaling group with a minimum size of one instance. Modify the video transcoding application to continuously poll the SQS queue, and start transcoding the files in S3 that the SQS message links to** is incorrect. Although this approach decouples the components, is scalable, and ensures each file is processed only once, it is not cost-effective for the EC2 instances to continuously poll the SQS queue.

The option that says: **Create an equivalent script of the application workflow and deploy it as an AWS Lambda function. Use an S3 event notification to start the processing when a new video is uploaded** is incorrect. AWS Lambda currently has a function execution time limit of 900 seconds or 15 minutes before it is automatically terminated. The scenario talks about a long-running workload, and using AWS Lambda cannot fulfill this requirement.

The option that says: **Build a container image out of the video transcoding application. Deploy and run the container image on an EC2 instance. Configure the container to continuously poll the S3 bucket for new videos and to transcode the files** is incorrect. This solution allows for full control of the container application in the EC2 image, but it is not cost-effective as it requires the EC2 instance to be constantly running. There is also an additional management overhead of scaling the EC2 instances, which is not the best practice.

O **AWS Site-to-Site VPN** é um **túnel seguro e criptografado** entre a sua **rede local (on-premises)** e uma **VPC na AWS**, através da Internet pública.

Ele usa o protocolo **IPsec (Internet Protocol Security)**.

4 A logistics company is strategizing to transition their on-premises Microsoft server environments to the AWS cloud. After conducting successful preliminary tests on AWS and establishing a secure AWS Site-to-Site VPN connection to an Amazon VPC, the company's solutions architect is tasked with producing a comprehensive Total Cost of Ownership (TCO) analysis for the proposed migration.

Each server in the company's on-premises data center has Simple Network Management Protocol (SNMP) activated. However, due to company policies, they are unable to expand their server count or install new software on existing servers. The data gathered must be seamlessly integrated with AWS Migration Hub.

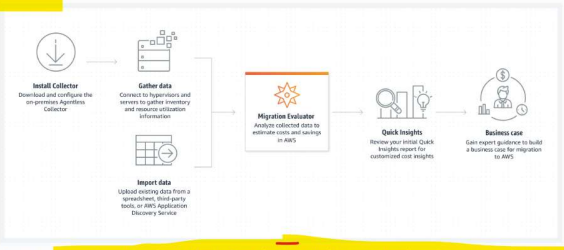
What approach should be taken to meet these requirements?

(view) 1 0 1 00:00:27

- Deploy a Windows Amazon EC2 instance and install the Migration Evaluator agentless collector on the EC2 instance to collect the data from the on-premises servers via SNMP. Use the Migration Evaluator to generate the comprehensive Total Cost of Ownership (TCO) analysis for the proposed migration.
- Use AWS Application Migration Service for agentless replication from the company's on-premises servers to AWS. Then, use AWS Migration Hub Strategy Recommendations to generate a comprehensive Total Cost of Ownership (TCO) analysis for the proposed migration.
- Deploy a Windows Amazon EC2 instance and install the Migration Evaluator agentless collector on the EC2 instance to collect the data from the on-premises servers via SNMP. Use AWS Migration Hub to generate the comprehensive Total Cost of Ownership (TCO) analysis for the proposed migration.
- Use AWS Snowball to physically transfer the data from the company's on-premises servers to AWS. Then, use AWS Cost Explorer to generate a comprehensive Total Cost of Ownership (TCO) analysis for the proposed migration.

microsoft-> **windows**, ec2+**SNMP(** Simple Network Management Protocol**)**+TCO(Total cost of ownership) + migration hub-> **Migration Evaluator Agentless Collector**

The Migration Evaluator Agentless Collector is a tool provided by AWS that can be deployed on a Windows Amazon EC2 instance. This tool is designed to collect data from on-premises servers via the Simple Network Management Protocol (SNMP). The types of data collected by the Migration Evaluator Agentless Collector include server configuration, utilization, annual costs to operate, eligibility for bring-your-own-license, and hundreds of other parameters. This data is crucial for understanding the current state of the on-premises servers and planning for their migration to the AWS cloud.



costs

Once the data is collected, it is then analyzed by the Migration Evaluator to generate a comprehensive Total Cost of Ownership (TCO) analysis. The TCO analysis provides a clear baseline of what your organization is running today and projects AWS costs based on measured on-premises provisioning and utilization. This analysis is essential for understanding the financial impact of migrating the on-premises servers to the AWS cloud. The Migration Evaluator service analyzes an enterprise's compute footprint, including server configuration, utilization, annual costs to operate, eligibility for bring-your-own-license, and hundreds of other parameters. This allows the company to make informed decisions about using AWS.

In summary, the Migration Evaluator Agentless Collector and the Migration Evaluator work together to comprehensively analyze the costs and benefits of migrating on-premises servers to the AWS cloud. This process helps companies make data-driven decisions and build a strong business case for their migration efforts.

5 A company uses Lightweight Directory Access Protocol (LDAP) for its employee authentication and authorization. The company plans to release a mobile app that can be installed on employee's smartphones. The mobile application will allow users to have federated access to AWS resources. Due to strict security and compliance requirements, the mobile application must use a custom-built solution for user authentication. It must also use IAM roles for granting user permissions to AWS resources. The Solutions Architect was tasked to create a solution that meets these requirements.

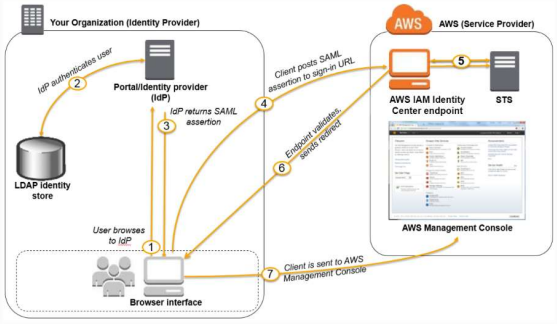
Which of the following options should the Solutions Architect implement to enable authentication and authorization for the application? (Select TWO.)

(view) 1 0 1 00:00:00

- Build a custom SAML-compatible solution for user authentication. Leverage AWS IAM Identity Center for authorizing access to AWS resources.
- Build a custom LDAP connector using Amazon API Gateway with AWS Lambda function for user authentication. Use Amazon DynamoDB to store user authorization tokens. Write another Lambda function that will validate user authorization requests based on the token stored on DynamoDB.
- Build a custom OpenID Connect-compatible solution in combination with AWS IAM Identity Center to create authentication and authorization functionality for the application.
- **Build a custom SAML-compatible solution to handle authentication and authorization. Configure the solution to use LDAP for user authentication and use SAML assertion to perform authorization to the IAM identity provider.**
- **Build a custom OpenID Connect-compatible solution for the user authentication functionality. Use Amazon Cognito Identity Pools for authorizing access to AWS resources.**

AWS supports **identity federation with SAML 2.0** (Security Assertion Markup Language 2.0), an open standard that many identity providers (IdPs) use. This feature enables federated single sign-on (SSO), so users can log in to the AWS Management Console or call the AWS API operations without having to create an IAM user for everyone in your organization. By using SAML, you can simplify the process of configuring federation with AWS because you can use the IdP's service instead of writing custom identity proxy code.

You can use a role to configure your SAML 2.0-compliant identity provider (IdP) and AWS to permit your federated users to access the AWS Management Console. The role grants the user permissions to carry out tasks in the console. The following diagram illustrates the flow for SAML-enabled single sign-on.



The diagram illustrates the following steps:

1. The user browses your organization's portal and selects the option to go to the AWS Management Console. In your organization, the portal is typically a function of your IdP that handles the exchange of trust between your organization and AWS.
2. The portal verifies the user's identity in your organization.
3. The portal generates a SAML authentication response that includes assertions that identify the user and include attributes about the user. The portal sends this response to the client's browser.
4. The client browser is redirected to the AWS IAM Identity Center endpoint and posts the SAML assertion.
5. The endpoint requests temporary security credentials on behalf of the user and creates a console sign-in URL that uses those credentials.
6. AWS sends the sign-in URL back to the client as a redirect.
7. The client browser is redirected to the AWS Management Console. If the SAML authentication response includes attributes that map to multiple IAM roles, the user is first prompted to select the role for accessing the console.

Amazon Cognito provides authentication, authorization, and user management for your web and mobile apps. Your users can sign in directly with a username 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.

Amazon Cognito identity pools provide temporary AWS credentials for users who are guests (unauthenticated) and for users who have been authenticated and have received a token.

OpenID Connect is an open standard for authentication that is supported by a number of login providers. Amazon Cognito supports the linking of identities with OpenID Connect providers that are configured through AWS Identity and Access Management. Once you've created an OpenID Connect provider in the IAM Console, you can associate it with an identity pool.

6 A company is using AWS Managed Active Directory Service to host the company AD in the AWS Cloud with a custom AD domain name private.tutorialsdojo.com. A pair of domain controllers are launched with the default configuration inside the VPC. A VPC interface endpoint was also created for the Amazon Kinesis using AWS Private Link to allow instances to connect to Kinesis service endpoints from inside the VPC. The solutions architect launched several EC2 instances in the VPC, however, the instances were not able to resolve the company's custom AD domain name.

Which of the following steps should the Solutions Architect implement to allow the instances to resolve both AWS VPC endpoints and the AWS Managed Microsoft AD domain's FQDN? (Select TWO.)

(view)

- **Create an outbound endpoint on the Amazon Route 53 console. Set the AmazonProvidedDNS as the DNS resolver for the VPC.**
- **Create a forwarding rule inside the endpoint to forward any queries for private.tutorialsdojo.com to the IP addresses of the two domain controllers.**
- Create an inbound endpoint on the Amazon Route 53 console. Set the AmazonProvidedDNS as the DNS resolver for the VPC.
- Reconfigure the DNS service on every client on the VPC to split DNS queries. Use the Active Directory servers for the custom AD domain and the VPC resolver for all other DNS queries.
- Create a conditional forwarder inside the endpoint to forward any queries for private.tutorialsdojo.com to the IP addresses of the two domain controllers.

Outbound Resolver Endpoints host Forwarding Rules that forward queries for specified domain names to specific IP addresses. You create forwarding rules when you want to forward DNS queries for specified domain names to DNS resolvers on your network.

To forward selected queries, you create Resolver rules that specify the domain names for the DNS queries that you want to forward (such as example.com), and the IP addresses of the DNS resolvers on your network that you want to forward the queries to. If a query matches multiple rules (tutorialsdojo.com, portal.tutorialsdojo.com), the Resolver chooses the rule with the most specific match (portal.tutorialsdojo.com) and forwards the query to the IP addresses that you specified in that rule.

What is a Domain Controller (DC)

A **Domain Controller** is a **server** in a **Windows network** (specifically an **Active Directory** environment) that **authenticates and authorizes users and computers** in a **domain**.

Think of it as the **"identity and security brain"** of your Windows network.

🏢 In simple terms

When someone logs into their company laptop with a corporate account — 👉 the **Domain Controller** checks their username and password.

👉 if correct, it gives them access to resources like file shares, printers, and applications.

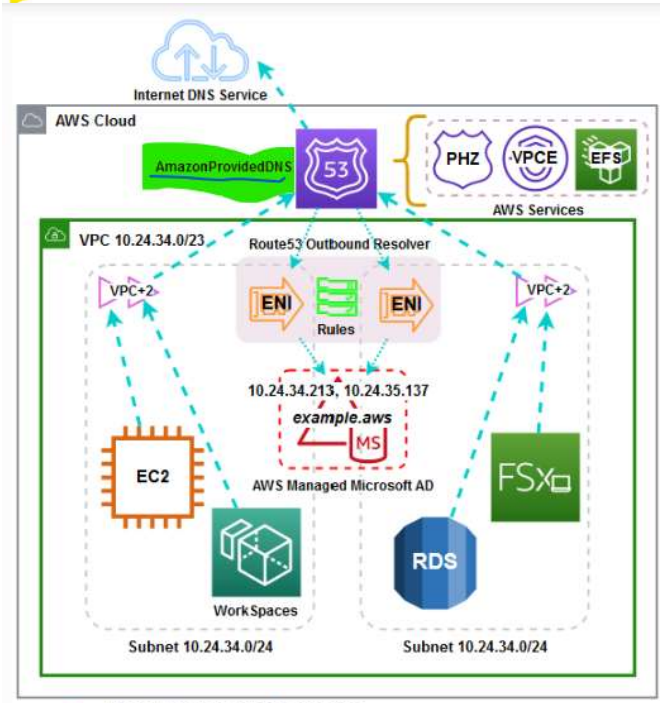
When you create a VPC using Amazon VPC, **Route 53 Resolver** automatically answers DNS queries for local VPC domain names for EC2 instances (ec2-192-0-2-44.compute-1.amazonaws.com) and records in private hosted zones (private.tutorialsdojo.com). For all other domain names, Resolver performs recursive lookups against public name servers.

You also can integrate DNS resolution between Resolver and DNS resolvers on your network by configuring forwarding rules. Your network can include any network that is reachable from your VPC, such as the following:

- The VPC itself
- Another peered VPC
- An on-premises network that is connected to AWS with AWS Direct Connect, a VPN, or a network address translation (NAT) gateway.

A Route 53 Resolver Endpoint is a customer-managed resolver consisting of one or more Elastic Network Interfaces (ENIs) deployed on your VPC. Resolver Endpoints are classified into two types:

- **Inbound Endpoint** - provides DNS resolution for AWS resources, such as EC2 instances, for your corporate network
- **Outbound Endpoint** - provides resolution of specific DNS names that you configure using forwarding rules for your VPC.



When **Outbound Endpoint** and **Forwarding Rule** are created, any resource in the VPC that queries the AmazonProvidedDNS as its DNS resolver is able to seamlessly resolve for AWS Managed Microsoft AD domain's FQDN, as well as any AWS resources in the VPC such as (interface) VPC Endpoints.

The option that says: **Create an outbound endpoint on the Amazon Route 53 console. Set the AmazonProvidedDNS as the DNS resolver for the VPC** is correct. You need an outbound endpoint to forward and resolve custom domain names inside your VPC.

The option that says: **Create a forwarding rule inside the endpoint to forward any queries for private.tutorialsdojo.com to the IP addresses of the two domain controllers** is correct. The forwarding rules will handle queries for a given DNS domain and forward them to the AD server to resolve them.

The option that says: **Create an inbound endpoint on the Amazon Route 53 console. Set the AmazonProvidedDNS as the DNS resolver for the VPC** is incorrect. An inbound endpoint is used for DNS resolution of AWS services. You need an outbound endpoint for this scenario.

The option that says: **Reconfigure the DNS service on every client on the VPC to split DNS queries. Use the Active Directory servers for the custom AD domain and the VPC resolver for all other DNS queries** is incorrect. It is not recommended to manually configure resources to split DNS queries. This entails a lot of management overhead. You just need to set the Active Directory servers as the DNS servers and the requests will be forwarded to the VPC resolver accordingly.

The option that says: **Create a conditional forwarder inside the endpoint to forward any queries for private.tutorialsdojo.com to the IP addresses of the two domain controllers** is incorrect. A conditional forwarder is configured inside the AD servers, not on the Route 53 resolver endpoint.

8 A fashion company in France sells bags, clothes, and other luxury items in its online web store. The online store is currently hosted on the company's on-premises data center. The company has recently decided to move all of its on-premises infrastructure to the AWS cloud. The main application is running on an NGINX web server and a database with an Oracle Real Application Clusters (RAC) One Node configuration.

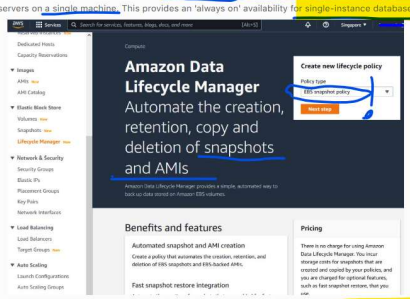
Which of the following is the best way to migrate the application to AWS and set up an automated backup?

(view) 1 0

- Launch an EC2 instance and run an NGINX server to host the application. Deploy an RDS instance and enable automated backups on the RDS RAC cluster.
- Launch an On-Demand EC2 instance and run an NGINX server to host the application. Deploy an RDS instance with a Multi-AZ deployment configuration and enable automated backups on the RDS RAC cluster.
- Launch an EC2 instance for both the NGINX server as well as for the database. Attach EBS Volumes on the EC2 instance of the database and then write a shell script that runs the manual snapshot of the volumes.
- Launch an EC2 instance for both the NGINX server as well as for the database. Attach EBS volumes to the EC2 instance of the database and then use the Data Lifecycle Manager to automatically create scheduled snapshots against the EBS volumes.

Oracle RAC is not supported by RDS. That is why you need to deploy the database in an EC2 instance and then either create a shell script to automate the backup or use the Data Lifecycle Manager to automate the process.

An Oracle Real Application Clusters (RAC) One Node option provides virtualized servers on a single machine. This provides an 'always on' availability for single-instance databases for a fraction of a cost.



Amazon Data Lifecycle Manager (DLM) for EBS Snapshots provides a simple, automated way to back up data stored on Amazon EBS volumes. You can define backup and retention schedules for EBS snapshots by creating lifecycle policies based on tags. With this feature, you no longer have to rely on custom scripts to create and manage your backups.

The option that says: **Launch an EC2 instance for both the NGINX server as well as for the database. Attach EBS Volumes on the EC2 instance of the database and then write a shell script that runs the manual snapshot of the volumes** is incorrect. Although this approach is valid, a more suitable option is to use the Data Lifecycle Manager (DLM) to automatically take the snapshot of the EC2 instance. The DLM can also reduce storage costs by deleting outdated backups.

The following options are incorrect as these two use Amazon RDS, which doesn't natively support Oracle RAC:

- **Launch an EC2 instance and run an NGINX server to host the application. Deploy an RDS instance and enable automated backups on the RDS RAC cluster.**
- **Launch an On-Demand EC2 instance and run an NGINX server to host the application. Deploy an RDS instance with a Multi-AZ deployment configuration and enable automated backups on the RDS RAC cluster.**

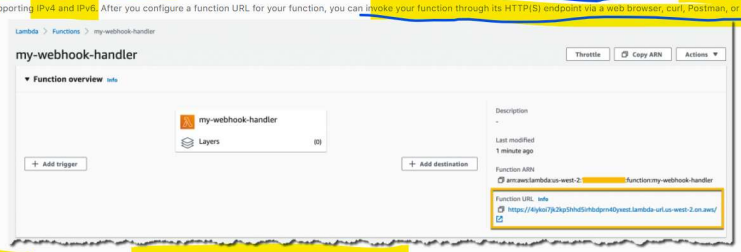
11 A company has a marketing service that schedules and posts content on a social media platform. The service currently runs on a collection of Amazon EC2 instances within an Auto Scaling group behind an Application Load Balancer (ALB). This ALB responds to webhook calls from the social media platform when specific events occur, such as mentions, shares, or specific hashtag uses. The company wants to migrate this service to a serverless architecture to lessen administrative overhead.

Which action would be the most operationally efficient way to meet the requirements?

(view) 1

- Run the service using AWS Lambda. Generate a Lambda Function URL for each webhook. Configure the platform to send webhook calls to the function URLs.
- Use AWS Lambda to run the service. Create a single AWS AppSync GraphQL API endpoint and connect it to the Lambda functions. Configure the platform to send webhook calls to the AppSync endpoint.
- Replace the Amazon EC2 instances with AWS Fargate tasks. Create an Amazon Gateway HTTP API and integrate it with the Application Load Balancer. Configure the platform to send webhook calls to the HTTP API endpoint.
- Run the service using AWS Lambda. Create an Amazon Gateway REST API that uses proxy integration with AWS Lambda. Create an API resource for each webhook. Configure the platform to send webhook calls to the REST API endpoint.

Lambda function URLs are HTTP(S) endpoints dedicated to your Lambda function. You can easily create and set up a function URL using the Lambda console or API. Once created, Lambda generates a unique URL endpoint for your use. Function URLs are dual stack-enabled, supporting IPv4 and IPv6. After you configure a function URL for your function, you can invoke your function through its HTTP(S) endpoint via a web browser, curl, Postman, or any HTTP client.



In the scenario, exposing Lambda functions directly using function URLS eliminates the need for intermediaries like Amazon API Gateway. This approach significantly simplifies the architecture, reduces latency, and lessens administrative overhead. Each webhook can be configured to send calls to specific Lambda function URLs, providing a response mechanism for various social media events.

在此场景中，直接使用函数 URL 公开 Lambda 函数，无需 Amazon API Gateway 等中介。这种方法显著简化了架构，降低了延迟，并减少了管理开销。每个 Webhook 都可以配置为将调用发送到特定的 Lambda 函数 URL，从而为各种社交媒体事件提供响应机制。

n the scenario, exposing Lambda functions directly using function URLS eliminates the need for intermediaries like Amazon API Gateway. This approach significantly simplifies the architecture, reduces latency, and lessens administrative overhead. Each webhook can be configured to send calls to specific Lambda function URLs, providing a response mechanism for various social media events.

Hence, the correct answer is: **Run the service using AWS Lambda. Generate a Lambda Function URL for each webhook. Configure the platform to send webhook calls to the function URLs.**

The option that says: **Use AWS Lambda to run the service. Create a single AWS AppSync GraphQL API endpoint and connect it to the Lambda functions. Configure the platform to send webhook calls to the AppSync endpoint** is incorrect. AWS AppSync is a just managed service that uses GraphQL to make it easy for applications to get exactly the data they need. While you can use it with Lambda, this solution is not as straightforward as directly triggering Lambda functions using function URLs. Using AppSync might introduce unnecessary complexity for this use case which is about responding to webhook calls from a social media platform, not handling GraphQL queries from a client application.

The option that says: **Replace the Amazon EC2 instances with AWS Fargate tasks. Create an Amazon Gateway HTTP API and integrate it with the Application Load Balancer. Configure the platform to send webhook calls to the HTTP API endpoint** is incorrect. While AWS Fargate is a serverless compute engine for containers, it wouldn't be the most efficient choice in the scenario. Using Fargate would require you to containerize the application, which adds an additional layer of development and management overhead

The option that says: **Run the service using AWS Lambda. Create an Amazon Gateway REST API that uses proxy integration with AWS Lambda. Create an API resource for each webhook. Configure the platform to send webhook calls to the REST API endpoint** is incorrect. Although this option could work, it introduces additional complexity by using Amazon API Gateway as an intermediary. In this scenario, a fully-fledged REST API is not necessary. Instead of creating an API resource for each webhook, it would be simpler and more efficient to directly trigger specific AWS Lambda functions thru Lambda function URLs.

9 A multinational consumer goods company is currently using a VMWare vCenter Server to manage their virtual machines, multiple ESXi hosts, and all dependent components from a single centralized location. To save costs and to avail the benefits of cloud computing, the company decided to move its virtual machines to AWS. The Solutions Architect is required to generate new AMIs of the existing virtual machines which can then be launched as an EC2 instance in the company VPC.

Which combination of steps should the Solutions Architect do to properly execute the cloud migration? (Select TWO.)

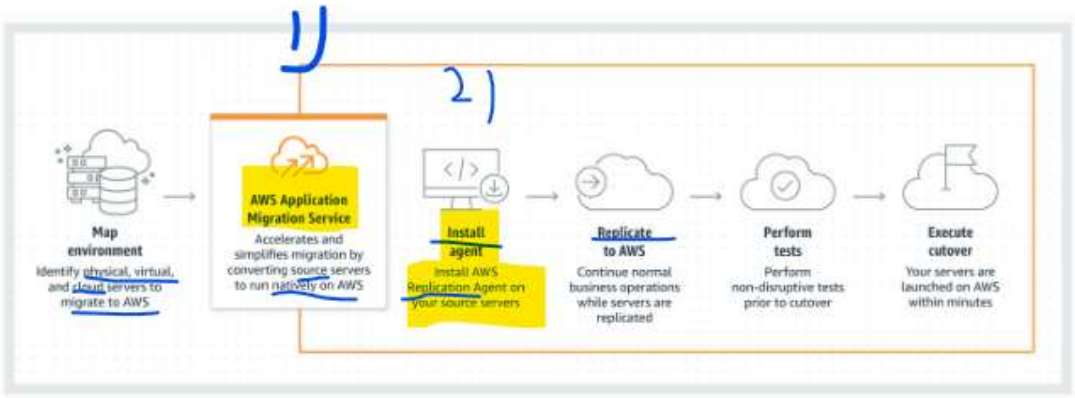
(view)

- Use the AWS Application Migration Service to migrate your on-premises workloads to the AWS cloud.
- Install the AWS Replication Agent in your on-premises virtualization environment.
- Use Serverless Application Model (SAM) to migrate the virtual machines (VMs) to AWS and automatically launch an Amazon ECS Cluster to host the VMs.
- Create an AWS CloudFormation template that mirrors the on-premises virtualized environment. Deploy the stack to the AWS cloud.
- Establish a Direct Connect connection between your data center and your VPC. Use AWS Service Catalog to centrally manage all your IT services and to quickly migrate virtual machines to your virtual private cloud.

AWS Application Migration Service (MGN) is a highly automated lift-and-shift (rehost) solution that simplifies, expedites, and reduces the cost of migrating applications to AWS. It enables companies to lift and shift a large number of physical, virtual, or cloud servers without compatibility issues, performance disruption, or long cutover windows.

MGN replicates source servers into your AWS account. When you're ready, it automatically converts and launches your servers on AWS so you can quickly benefit from the cost savings, productivity, resilience, and agility of the Cloud.

Once your applications are running on AWS, you can leverage AWS services and capabilities to quickly and easily re-platform or refactor those applications – which makes lift-and-shift a fast route to modernization.


1)
2)
Map environment
Identify physical, virtual, and cloud servers to migrate to AWS
AWS Application Migration Service
Accelerates and simplifies migration by converting source servers to run natively on AWS
Install agent
Install AWS Replication Agent on your source servers
Replicate to AWS
Continue normal business operations while servers are replicated
Perform tests
Perform non-disruptive tests prior to cutover
Execute cutover
Your servers are launched on AWS within minutes