**AWS Solutions Architect Professional Practice Test 1**

Top of Form

Question 1: **Incorrect**

An unusual API activity and intra-VPC port scanning have been identified by the security team of your department. They noticed that there are multiple port scans being triggered to your EC2 instances from a specific IP address. To fix the issue immediately, your team has decided to simply block the offending IP address. You are also instructed to fortify their existing cloud infrastructure security from the most frequently occurring network and transport layer DDoS attacks.

Which of the following is the most suitable method to satisfy the above requirement in AWS?

* ​

Block the offending IP address using Route 53. Use Amazon Macie to automatically discover, classify, and protect sensitive data in AWS, including DDoS attacks.

* ​

Deny access from the IP Address block in the Network ACL. Use AWS Shield Advanced to protect your cloud resources.

**(Correct)**

* ​

Deny access from the IP Address block by adding a specific rule to all of the Security Groups. Use a combination of AWS WAF and AWS Config to protect your cloud resources against common web attacks.

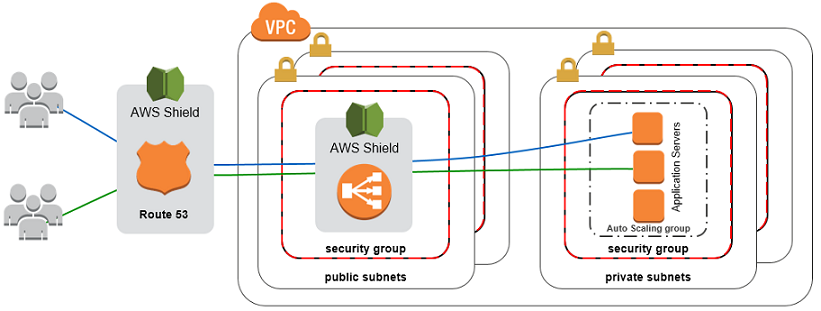
* ​

Change the Windows Firewall settings to deny access from the IP address block. Use Amazon GuardDuty to detect potentially compromised instances or reconnaissance by attackers, and AWS Systems Manager Patch Manager to properly apply the latest security patches to all of your instances.

**(Incorrect)**

**Explanation**

***AWS Shield***is a managed Distributed Denial of Service (DDoS) protection service that safeguards applications running on AWS. AWS Shield provides always-on detection and automatic inline mitigations that minimize application downtime and latency, so there is no need to engage AWS Support to benefit from DDoS protection. There are two tiers of AWS Shield - Standard and Advanced.



All AWS customers benefit from the automatic protections of AWS Shield Standard, at no additional charge. AWS Shield Standard defends against most common, frequently occurring network and transport layer DDoS attacks that target your web site or applications. When you use AWS Shield Standard with Amazon CloudFront and Amazon Route 53, you receive comprehensive availability protection against all known infrastructure (Layer 3 and 4) attacks.

A network access control list (ACL) is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets. You might set up network ACLs with rules similar to your security groups in order to add an additional layer of security to your VPC. A network ACL has separate inbound and outbound rules, and each rule can either allow or deny traffic. Network ACLs are stateless; responses to allowed inbound traffic are subject to the rules for outbound traffic (and vice versa).

The option that says: ***Block the offending IP address using Route 53. Use Amazon Macie to automatically discover, classify, and protect sensitive data in AWS, including DDoS attacks*** is incorrect because Amazon Macie is just a security service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. It does not provide security against DDoS attacks. In addition, you cannot block the offending IP address using Route 53. You should use Network ACL for this scenario.

The option that says: ***Change the Windows Firewall settings to deny access from the IP address block. Use Amazon GuardDuty to detect potentially compromised instances or reconnaissance by attackers, and AWS Systems Manager Patch Manager to properly apply the latest security patches to all of your instances***is incorrect because you have to use Network ACL to block the specific IP address to your network and not just change the firewall of your Windows server. Amazon GuardDuty and AWS Systems Manager Patch Manager are not suitable to fortify your AWS Cloud against DDoS attacks.

The option that says:***Deny access from the IP Address block by adding a specific rule to all of the Security Groups. Use a combination of AWS WAF and AWS Config to protect your cloud resources against common web attacks*** is incorrect because it is still better to block the offending IP address on the Network ACL level as you cannot directly deny an IP address in your Security Group. AWS WAF and AWS Config are helpful to improve the security of your cloud infrastructure in AWS but these services are not enough to protect your infrastructure against DDoS attacks. You have to use AWS Shield Advanced in this scenario.

**References:**

<https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html>

<https://aws.amazon.com/shield/>

**Check out this AWS WAF and AWS Shield Cheat Sheets:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-waf/>

<https://tutorialsdojo.com/aws-cheat-sheet-aws-shield/>

Bottom of Form

Top of Form

Question 2: Skipped

A data analytics startup has been chosen to develop a data analytics system that will track all statistics in the Fédération Internationale de Football Association (FIFA) World Cup which will also be used by other 3rd-party analytics sites. The system will record, store and provide statistical data reports about the top scorers, goal scores for each team, average goals, average passes and average yellow/red cards per match and many other details. FIFA fans all over the world will frequently access the statistics reports everyday and thus, it should be durably stored, highly available and highly scalable. In addition, the data analytics system will allow the users to vote for the best male and female FIFA player as well as the best male and female coach. Due to the popularity of the FIFA World Cup event, it is projected that there will be over 10 million queries on game day and could spike to 30 million queries over the course of time.

Which of the following is the most cost-effective solution that will meet these requirements?

* ​

1. Generate the FIFA reports from MySQL database in Multi-AZ RDS deployments configuration with Read Replicas.

2. Set up a batch job that put reports in an S3 bucket.

3. Launch a CloudFront distribution to cache the content with a TTL set to expire objects daily.

**(Correct)**

* ​

1. Launch a Multi-AZ MySQL RDS instance.

2. Query the RDS instance and store the results in a DynamoDB table.

3. Generate reports from DynamoDB table.

4. Delete the old DynamoDB tables everyday.

* ​

1. Launch a MySQL database in Multi-AZ RDS deployments configuration with Read Replicas.

2. Generate the FIFA reports by querying the Read Replica.

3. Configure a daily job that performs a daily table cleanup.

* ​

1. Launch a MySQL database in Multi-AZ RDS deployments configuration.

2. Configure the application to generate reports from ElastiCache to improve the read performance of the system.

3. Utilize the default expire parameter for items in ElastiCache.

**Explanation**

In this scenario, you are required to have the following:

A durable storage for the generated reports.

A database that is highly available and can scale to handle millions of queries.

A Content Delivery Network that can distribute the report files to the users all over the world.

Hence, the following option is the best solution that satisfies all of these requirements. S3 provides the durable storage; Multi-AZ RDS with Read Replicas provide a scalable and highly available database and CloudFront provides the CDN:

***1. Generate the FIFA reports from MySQL database in Multi-AZ RDS deployments configuration with Read Replicas.***

***2. Set up a batch job that put reports in an S3 bucket.***

***3. Launch a CloudFront distribution to cache the content with a TTL set to expire objects daily.***

The following option is incorrect because although the database is scalable and highly available, it neither has any durable data storage nor a CDN:

***1. Launch a MySQL database in Multi-AZ RDS deployments configuration with Read Replicas.***

***2. Generate the FIFA reports by querying the Read Replica.***

***3. Configure a daily job that performs a daily table cleanup.***

The following option is incorrect because although this option handles and provides a better read capability for the system, it is still lacking a durable storage and a CDN:

***1. Launch a MySQL database in Multi-AZ RDS deployments configuration.***

***2. Configure the application to generate reports from ElastiCache to improve the read performance of the system.***

***3. Utilize the default expire parameter for items in ElastiCache.***

The following option is incorrect because it is not a cost-effective solution to maintain both RDS and a DynamoDB instance:

***1. Launch a Multi-AZ MySQL RDS instance.***

***2. Query the RDS instance and store the results in a DynamoDB table.***

***3. Generate reports from DynamoDB table.***

***4. Delete the old DynamoDB tables everyday.***

**References:**

<https://aws.amazon.com/rds/details/multi-az/>

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_BestPractices.html>

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/Introduction.html>

**Check out this Amazon RDS Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-relational-database-service-amazon-rds/>

Bottom of Form

Top of Form

Question 3: Skipped

A Solutions Architect has been assigned to develop a workflow to ensure that the required patches of all of their Windows EC2 instances are properly identified and applied automatically. To maintain their system uptime requirements, it is of utmost importance to ensure that the EC2 instance reboots do not occur at the same time on all of their Windows instances. This is to avoid any loss of revenue that could be caused by any unavailability issues of their systems.

Which of the following will meet the above requirements?

* ​

Create two Patch Groups with unique tags that you will assign to all of your EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaselinebaseline on both patch groups. Create two CloudWatch Events rules which are configured to use a cron expression to automate the execution of patching for the two Patch Groups using the AWS Systems Manager Run command. Set up an AWS Systems Manager State Manager document to define custom commands which will be executed during patch execution.

* ​

Create a Patch Group with unique tags that you will assign to all of your EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaselinebaseline on your patch group. Set up a maintenance window and associate it with your patch group. Assign the AWS-RunPatchBaselinedocument as a task within your maintenance window.

* ​

Create two Patch Groups with unique tags that you will assign to all of your EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaseline baseline on both patch groups. Set up two non-overlapping maintenance windows and associate each with a different patch group. Using Patch Group tags, register targets with specific maintenance windows and lastly, assign the AWS-RunPatchBaseline document as a task within each maintenance window which has a different processing start time.

**(Correct)**

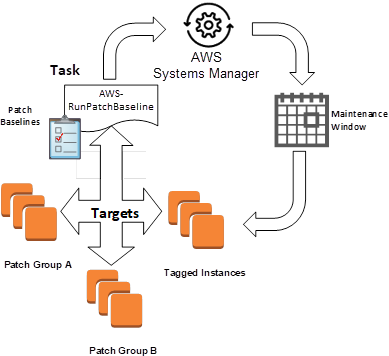
* ​

Create a Patch Group with unique tags that you will assign to all of your EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaselinebaseline on both patch groups. Create a CloudWatch Events rule configured to use a cron expression to automate the execution of patching in a given schedule using the AWS Systems Manager Run command. Set up an AWS Systems Manager State Manager document to define custom commands which will be executed during patch execution.

**Explanation**

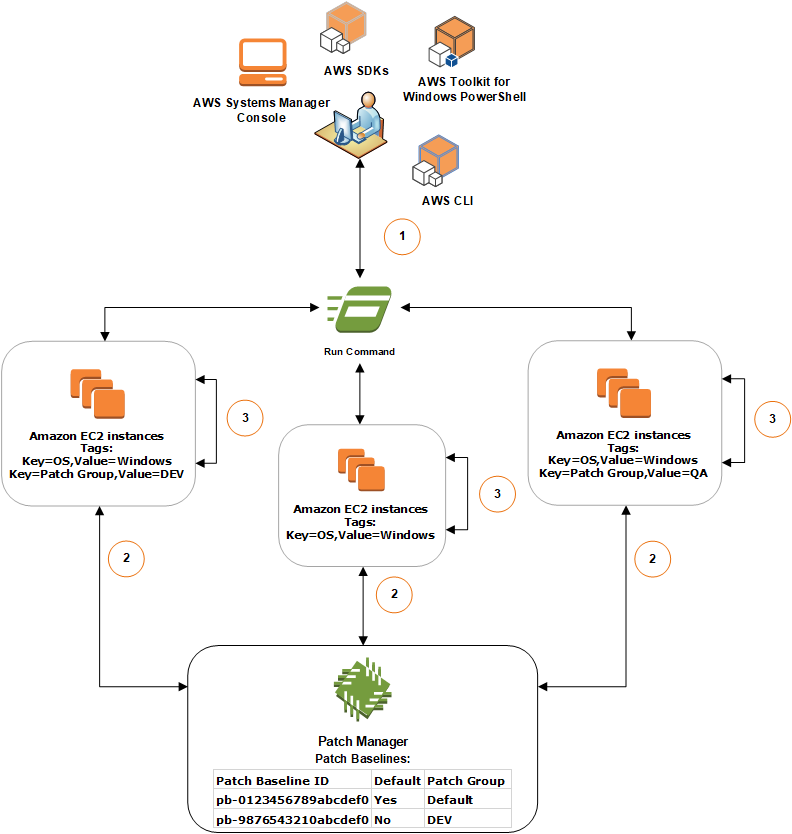
**AWS Systems Manager Patch Manager** automates the process of patching managed instances with both security-related and other types of updates. You can use Patch Manager to apply patches for both operating systems and applications.

You can patch fleets of Amazon EC2 instances or your on-premises servers and virtual machines (VMs) by operating system type. This includes supported versions of Windows Server, Ubuntu Server, Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), CentOS, Amazon Linux, and Amazon Linux 2. You can scan instances to see only a report of missing patches, or you can scan and automatically install all missing patches.



Patch Manager uses *patch baselines*, which include rules for auto-approving patches within days of their release, as well as a list of approved and rejected patches. You can install patches on a regular basis by scheduling patching to run as a Systems Manager maintenance window task. You can also install patches individually or to large groups of instances by using Amazon EC2 tags. You can add tags to your patch baselines themselves when you create or update them.

You can use a *patch group* to associate instances with a specific patch baseline. Patch groups help ensure that you are deploying the appropriate patches, based on the associated patch baseline rules, to the correct set of instances. Patch groups can also help you avoid deploying patches before they have been adequately tested. For example, you can create patch groups for different environments (such as Development, Test, and Production) and register each patch group to an appropriate patch baseline.



When you run AWS-RunPatchBaseline, you can target managed instances using their instance ID or tags. SSM Agent and Patch Manager will then evaluate which patch baseline to use based on the patch group value that you added to the instance.

You create a patch group by using Amazon EC2 tags. Unlike other tagging scenarios across Systems Manager, a patch group *must* be defined with the tag key: **Patch Group**. Note that the key is case-sensitive. You can specify any value, for example, "web servers," but the key must be **Patch Group**.

The AWS-DefaultPatchBaseline baseline is primarily used to approve all Windows Server operating system patches that are classified as "CriticalUpdates" or "SecurityUpdates" and that have an MSRC severity of "Critical" or "Important". Patches are auto-approved seven days after release.

Hence, the option that says: ***Create two Patch Groups with unique tags that you will assign to all of your EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaseline baseline on both patch groups. Set up two non-overlapping maintenance windows and associate each with a different patch group. Using Patch Group tags, register targets with specific maintenance windows and lastly, assign the AWS-RunPatchBaseline document as a task within each maintenance window which has a different processing start time*** is the correct answer as it properly uses two Patch Groups, non-overlapping maintenance windows and the AWS-DefaultPatchBaseline baseline to ensure that the EC2 instance reboots do not occur at the same time.

The option that says: ***Create a Patch Group with unique tags that you will assign to all of your EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaseline baseline on your patch group. Set up a maintenance window and associate it with your patch group. Assign the AWS-RunPatchBaseline document as a task within your maintenance window*** is incorrect because although it is correct to use a Patch Group, you must create another Patch Group to avoid any unavailability issues. Having two non-overlapping maintenance windows will ensure that there will be another set of running Windows EC2 instances while the other set is being patched.

The option that says: ***Create two Patch Groups with unique tags that you will assign to all of your EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaseline baseline on both patch groups. Create two CloudWatch Events rules which are configured to use a cron expression to automate the execution of patching for the two Patch Groups using the AWS Systems Manager Run command. Set up an AWS Systems Manager State Manager document to define custom commands which will be executed during patch execution*** is incorrect because the AWS Systems Manager Run Command is primarily used to remotely manage the configuration of your managed instances while AWS Systems Manager State Manager is just a configuration management service that automates the process of keeping your Amazon EC2 and hybrid infrastructure in a state that you define. These two services, including CloudWatch Events, are not suitable to be used in this scenario. The better solution would be to use AWS Systems Manager Maintenance Windows which lets you define a schedule for when to perform potentially disruptive actions on your instances such as patching an operating system, updating drivers, or installing software or patches.

The option that says: ***Create a Patch Group with unique tags that you will assign to all of your EC2 Windows Instances. Associate the predefined AWS-DefaultPatchBaseline baseline on both patch groups. Create a CloudWatch Events rule configured to use a cron expression to automate the execution of patching in a given schedule using the AWS Systems Manager Run command. Set up an AWS Systems Manager State Manager document to define custom commands which will be executed during patch execution*** is incorrect because, just as what is mentioned in Option 3, you have to use Maintenance Windows for scheduling the patches and you also need to set up two Patch Groups in this scenario instead of one.

**References:**

<https://aws.amazon.com/blogs/mt/patching-your-windows-ec2-instances-using-aws-systems-manager-patch-manager/>

<https://docs.aws.amazon.com/systems-manager/latest/userguide/patch-manager-ssm-documents.html>

<https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-patch-scheduletasks.html>

**Check out this AWS Systems Manager Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-systems-manager/>

Bottom of Form

Top of Form

Question 4: Skipped

A company wants to implement a multi-account strategy that will be distributed across its several research facilities. There will be approximately 50 teams in total that will need their own AWS accounts. A solution is needed to simplify the DNS management as there is only one team that manages all the domains and subdomains for the whole organization. This means that the solution should allow private DNS to be shared among virtual private clouds (VPCs) in different AWS accounts.

Which of the following solutions has the LEAST complex DNS architecture and allows all VPCs to resolve the needed domain names?

* ​

On AWS Resource Access Manager (RAM), set up a shared services VPC on your central account. Create a peering from this VPC to each VPC on the other accounts. On Amazon Route 53, create a private hosted zone associated with the shared services VPC. Manage all domains and subdomains on this hosted zone. On each of the other AWS Accounts, create a Route 53 private hosted zone and configure the Name Server entry to use the DNS of the central account.

* ​

Set up a VPC peering connection among the VPC of each account. Ensure that the each VPC has the attributes enableDnsHostnames and enableDnsSupport set to “TRUE”. On Amazon Route 53, create a private hosted zone associated with the central account’s VPC. Manage all domains and subdomains on this hosted zone. On each of the other AWS Accounts, create a Route 53 private hosted zone and configure the Name Server entry to use the DNS of the central account.

* ​

On AWS Resource Access Manager (RAM), set up a shared services VPC on your central account. Set up VPC peering from this VPC to each VPC on the other accounts. On Amazon Route 53, create a private hosted zone associated with the shared services VPC. Manage all domains and subdomains on this zone. Programmatically associate the VPCs from other accounts with this hosted zone.

**(Correct)**

* ​

Set up Direct Connect connections among the VPCs of each account using private virtual interfaces. Ensure that each VPC has the attributes enableDnsHostnames and enableDnsSupport set to “FALSE”. On Amazon Route 53, create a private hosted zone associated with the central account’s VPC. Manage all domains and subdomains on this hosted zone. Programmatically associate the VPCs from other accounts with this hosted zone.

**Explanation**

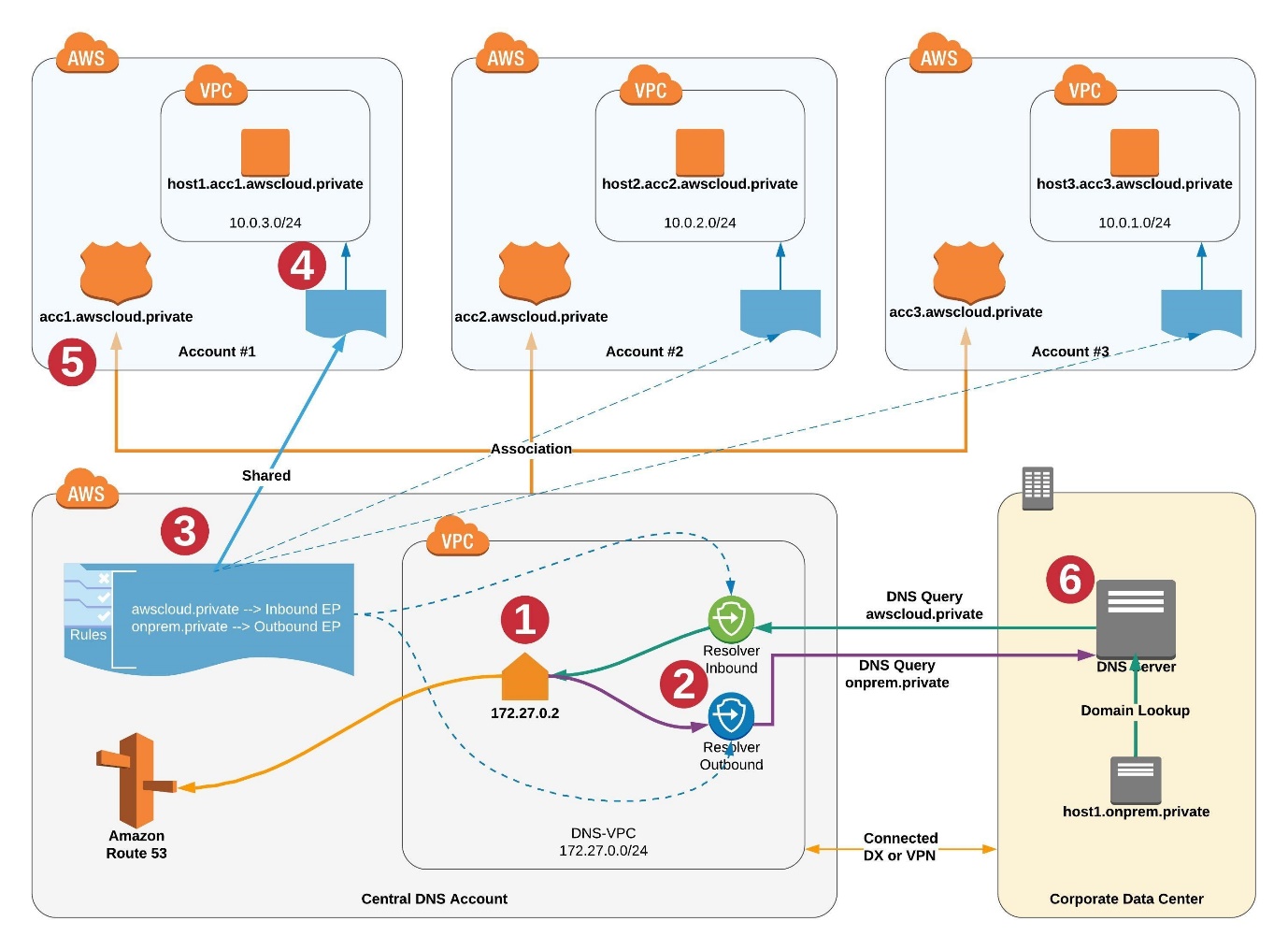
When you create a VPC using Amazon VPC, Route 53 Resolver automatically answers DNS queries for local VPC domain names for EC2 instances (ec2-192-0-2-44.compute-1.amazonaws.com) and records in private hosted zones (acme.example.com). For all other domain names, Resolver performs recursive lookups against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



VPC sharing allows customers to share subnets with other AWS accounts within the same AWS Organization. This is a very powerful concept that allows for a number of benefits:

- Separation of duties: centrally controlled VPC structure, routing, IP address allocation.

- Application owners continue to own resources, accounts, and security groups.

- VPC sharing participants can reference security group IDs of each other.

- Efficiencies: higher density in subnets, efficient use of VPNs and AWS Direct Connect.

- Hard limits can be avoided, for example, 50 VIFs per AWS Direct Connect connection through simplified network architecture.

- Costs can be optimized through reuse of NAT gateways, VPC interface endpoints, and intra-Availability Zone traffic.

Essentially, we can decouple accounts and networks. In this model, the account that owns the VPC (owner) shares one or more subnets with other accounts (participants) that belong to the same organization from AWS Organizations. After a subnet is shared, the participants can view, create, modify, and delete their application resources in the subnets shared with them. Participants cannot view, modify, or delete resources that belong to other participants or the VPC owner. You can simplify network topologies by interconnecting shared Amazon VPCs using connectivity features, such as AWS PrivateLink, AWS Transit Gateway, and Amazon VPC peering.

Therefore, the correct answer is: **On AWS Resource Access Manager (RAM), set up a shared services VPC on your central account. Set up VPC peering from this VPC to each VPC on the other accounts. On Amazon Route 53, create a private hosted zone associated with the shared services VPC. Manage all domains and subdomains on this zone. Programmatically associate the VPCs from other accounts with this hosted zone.**

The option that says: **Set up Direct Connect connections among the VPCs of each account using private virtual interfaces. Ensure that each VPC has the attributes enableDnsHostnames and enableDnsSupport set to “FALSE”. On Amazon Route 53, create a private hosted zone associated with the central account’s VPC. Manage all domains and subdomains on this hosted zone. Programmatically associate the VPCs from other accounts with this hosted zone** is incorrect because using AWS Direct Connect is not a suitable service to connect the various VPCs. In addition, attributes enableDnsHostnames and enableDnsSupport are set to “TRUE” by default and are needed for VPC resources to query Route 53 zone entries.

The option that says: **Set up a VPC peering connection among the VPC of each account. Ensure that each VPC has the attributes enableDnsHostnames and enableDnsSupport set to “TRUE”. On Amazon Route 53, create a private hosted zone associated with the central account’s VPC. Manage all domains and subdomains on this hosted zone. On each of the other AWS Accounts, create a Route 53 private hosted zone and configure the Name Server entry to use the DNS of the central account** is incorrect. You won't be able to resolve the hosted private zone entries even if you configure your Route 53 zone NS entry to use the central accounts' DNS servers.

The option that says: **On AWS Resource Access Manager (RAM), set up a shared services VPC on your central account. Create a peering from this VPC to each VPC on the other accounts. On Amazon Route 53, create a private hosted zone associated with the shared services VPC. Manage all domains and subdomains on this hosted zone. On each of the other AWS Accounts, create a Route 53 private hosted zone and configure the Name Server entry to use the DNS of the central account** is incorrect. Although creating the shared services VPC is a good solution, configuring Route 53 Name Server (NS) records to point to the shared services VPC’s Route 53 is not enough. You need to associate the VPCs from other accounts to the hosted zone on the central account.

**References:**

<https://docs.aws.amazon.com/vpc/latest/userguide/vpc-sharing.html>

<https://aws.amazon.com/blogs/networking-and-content-delivery/vpc-sharing-a-new-approach-to-multiple-accounts-and-vpc-management/>

<https://aws.amazon.com/blogs/security/simplify-dns-management-in-a-multiaccount-environment-with-route-53-resolver/>

**Check out these Amazon VPC and Route 53 Cheat Sheets:**

[https://tutorialsdojo.com/amazon-vpc/](https://tutorialsdojo.com/amazon-vpc/?src=udemy)

[https://tutorialsdojo.com/amazon-route-53/](https://tutorialsdojo.com/amazon-route-53/?src=udemy)

Bottom of Form

Top of Form

Question 5: Skipped

A company needs a deployment solution for its application that is hosted on the AWS cloud. The company has the following requirements for the application:

- The instances must have 500GB worth of static dataset that is accessible for the application upon boot up.

- The instances must be able to scale-out or scale-in depending on the traffic load of the application.

- The Development must have a quick and automated way to deploy their code updates several times during the day.

- Security patches for the vulnerabilities on the operating system (OS) must be installed within 48 hours of release.

Which of the following solutions should the Solutions Architect implement to meet the company requirements while being cost-effective?

* ​

Create an Auto Scaling group of EC2 instances using the Amazon Linux AMI. Install the application on the EC2 instances. Replace the existing instances as soon as AWS releases a new Amazon Linux AMI version. Write a user data script that will download the 500 GB static dataset from an Amazon S3 bucket. Deploy the new version of the application to the instances using AWS CodeDeploy.

* ​

Install OS patches and create a new AMI using AWS Systems Manager. Use this new AMI for the Auto Scaling group of EC2 instances and replace the existing instances. Deploy the new version of the application to the instances using AWS CodeDeploy. Mount an Amazon EFS volume containing the static dataset on the instances upon boot up.

**(Correct)**

* ​

Install OS patches and create a new AMI using AWS Systems Manager. Use this new AMI for the Auto Scaling group of EC2 instances and replace the existing instances. Create a scheduled batch job that will run every night to deploy the new application version and install the OS patches. Mount an Amazon EFS volume containing the static dataset on the instances upon boot up.

* ​

Create an Auto Scaling group of EC2 instances using the Amazon Linux AMI. Install the application on the EC2 instances. Write a user data script that will download the 500 GB static dataset from an Amazon S3 bucket. Use AWS Systems Manager to install the OS patches as soon as they are released. Deploy the new version of the application to the instances using AWS CodeDeploy.

**Explanation**

**AWS Systems Manager Patch Manager** automates the process of patching managed instances with both security-related and other types of updates. You can use Patch Manager to apply patches for both operating systems and applications. You can use Patch Manager to install Service Packs on Windows instances and perform minor version upgrades on Linux instances.

Patch Manager uses patch baselines, which include rules for auto-approving patches within days of their release, as well as a list of approved and rejected patches. You can install patches on a regular basis by scheduling patching to run as a Systems Manager maintenance window task.

**AWS Systems Manager Automation** simplifies common maintenance and deployment tasks of Amazon EC2 instances and other AWS resources. Automation enables you to do the following:

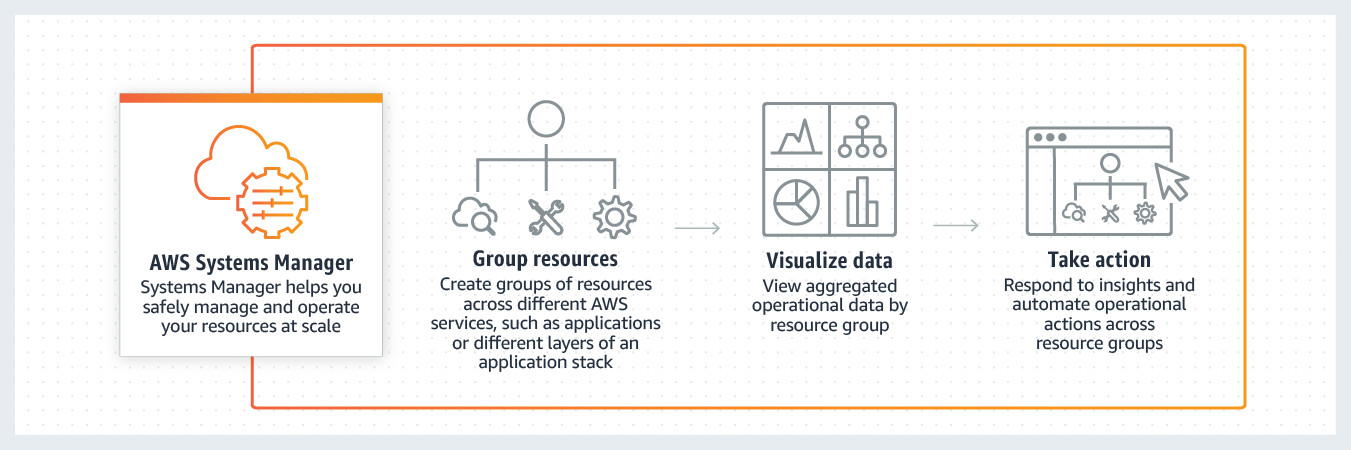
- Build automations to configure and manage instances and AWS resources.

- Create custom runbooks or use pre-defined runbooks maintained by AWS.

- Receive notifications about Automation tasks and runbooks by using Amazon EventBridge.

- Monitor Automation progress and details by using the AWS Systems Manager console.

AWS Systems Manager Automation provides several runbooks with pre-defined steps that you can use to perform common tasks like restarting one or more EC2 instances or creating an Amazon Machine Image (AMI). A Systems Manager Automation runbook defines the actions that Systems Manager performs on your managed instances and other AWS resources when an automation runs. A runbook contains one or more steps that run in sequential order. Each step is built around a single action. Output from one step can be used as input in a later step.



**Amazon Elastic File System (Amazon EFS)** provides a simple, scalable, fully managed elastic NFS file system for use with AWS Cloud services and on-premises resources. Amazon EFS has a simple web services interface that allows you to create and configure file systems quickly and easily. With Amazon EFS, you pay only for the storage used by your file system and there is no minimum fee or setup cost. Amazon EFS oﬀers two storage classes, Standard and Infrequent Access. The Standard storage class is used to store frequently accessed files. The Infrequent Access (IA) storage class is a lower-cost storage class that's designed for storing long-lived, infrequently accessed ﬁles cost-eﬀectively.

Amazon Elastic File System presents a standard file-system interface that supports full file-system access semantics. Using Network File System (NFS) version 4.1 (NFSv4.1), you can mount your Amazon EFS file system on any Amazon Elastic Compute Cloud (Amazon EC2) Linux-based instance. After your system is mounted, you can work with the files and directories just as you do with a local file system.

Therefore, the correct answer is: **Install OS patches and create a new AMI using AWS Systems Manager. Use this new AMI for the Auto Scaling group of EC2 instances and replace the existing instances. Deploy the new version of the application to the instances using AWS CodeDeploy. Mount an Amazon EFS volume containing the static dataset on the instances upon boot up.**

The option that says: **Install OS patches and create a new AMI using AWS Systems Manager. Use this new AMI for the Auto Scaling group of EC2 instances and replace the existing instances. Create a scheduled batch job that will run every night to deploy the new application version and install the OS patches. Mount an Amazon EFS volume containing the static dataset on the instances upon boot up** is incorrect. The OS patches can be installed every night, but it is not suitable for the application deployment. A batch job is not suitable for the application deployment as the Developers must deploy several times during the day.

The option that says: **Create an Auto Scaling group of EC2 instances using the Amazon Linux AMI. Install the application on the EC2 instances. Write a user data script that will download the 500 GB static dataset from an Amazon S3 bucket. Use AWS Systems Manager to install the OS patches as soon as they are released. Deploy the new version of the application to the instances using AWS CodeDeploy** is incorrect. Although Amazon S3 may seem more cost-effective than Amazon EFS in storing static contents, the Amazon EC2 instances will have to download the dataset on its local EBS volume. Attaching 500GB EBS volumes on each of the EC2 instances is more expensive compared to just using a single EFS volume mounted on all EC2 instances at boot up.

The option that says: **Create an Auto Scaling group of EC2 instances using the Amazon Linux AMI. Install the application on the EC2 instances. Replace the existing instances as soon as AWS releases a new Amazon Linux AMI version. Write a user data script that will download the 500 GB static dataset from an Amazon S3 bucket. Deploy the new version of the application to the instances using AWS CodeDeploy** is incorrect. The Amazon Linux AMI is patched for security vulnerabilities and OS minor versions. However, each new version usually takes weeks or months depending on Amazon’s release cycle.

**References:**

<https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-automation.html>

<https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-patch.html>

<https://docs.aws.amazon.com/systems-manager/latest/userguide/automation-documents.html>

<https://docs.aws.amazon.com/efs/latest/ug/whatisefs.html>

**Check out the AWS Systems Manager and Amazon EFS Cheat Sheets:**

[https://tutorialsdojo.com/aws-systems-manager/](https://tutorialsdojo.com/aws-systems-manager/?src=udemy)

[https://tutorialsdojo.com/amazon-efs/](https://tutorialsdojo.com/amazon-efs/?src=udemy)

Bottom of Form

Top of Form

Question 6: Skipped

You are working as a Senior Solutions Architect in a technology company and you have just created a new S3 bucket called “tutorialsdojo" in us-west-2 AWS region. You have enabled static website hosting and uploaded the static web content files in the S3 bucket including the index.html file. You registered a custom domain name called <code>www.tutorialsdojo.com</code> using Route 53 to connect it with the S3 bucket. The next day, a new record set has been created which points to the S3 website endpoint: http://tutorialsdojo.s3-website-us-west-2.amazonaws.com but when you tried to access the website, you cannot see any content. You tried going to tutorialsdojo.com and www.tutorialsdojo.com but neither of these links work properly.

Which of the following is the MOST likely cause of this issue that the Architect should fix?

* ​

Route 53 is still propagating the domain name changes. Wait for another 12 hours and then try again.

* ​

The site does not work because you have not set a value for the error.html file, which is a required step.

* ​

The S3 bucket does not have public read access which blocks the website visitors from seeing the content.

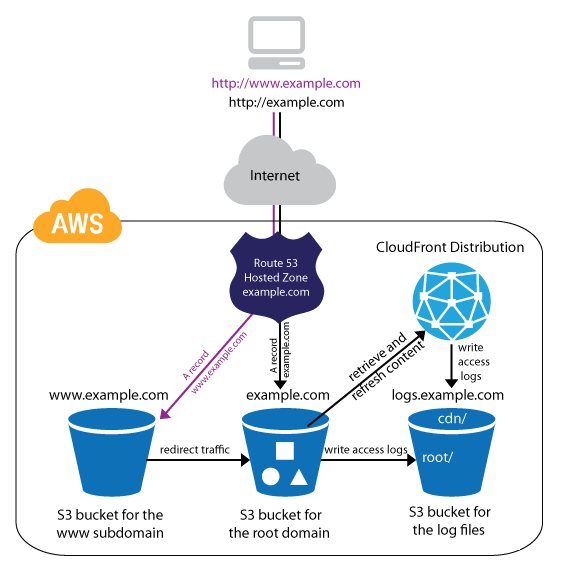
**(Correct)**

* ​

The site will not work because the URL does not include a file name at the end. This means that you need to use this URL instead: www.tutorialsdojo.com/index.html

**Explanation**

You can host a static website on Amazon Simple Storage Service (Amazon S3). On a static website, individual webpages include static content. They might also contain client-side scripts. To host a static website, you configure an Amazon S3 bucket for website hosting, and then upload your website content to the bucket. This bucket must have public read access. It is intentional that everyone in the world will have read access to this bucket.



In this scenario, the static S3 website does not work because the bucket does not have a public read access.

The option that says: **The S3 bucket does not have public read access which blocks the website visitors from seeing the content** is correct as this is the root cause why the static S3 website is inaccessible.

The option that says:**The site does not work because you have not set a value for the error.html file, which is a required step** is incorrect as the error.html is not required and won't affect the availability of the static S3 website.

The option that says: **The site will not work because the URL does not include a file name at the end. This means that you need to use this URL instead: www.tutorialsdojo.com/index.html** is incorrect as it is not required to manually append the exact filename in S3.

The option that says: **Route 53 is still propagating the domain name changes. Wait for another 12 hours and then try again** is incorrect as the Route 53 domain name propagation does not take that long. Remember that Amazon Route 53 is designed to propagate updates you make to your DNS records to its world-wide network of authoritative DNS servers within 60 seconds under normal conditions.

**References:**

<http://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html>

<https://docs.aws.amazon.com/AmazonS3/latest/dev/website-hosting-custom-domain-walkthrough.html>

**Check out this Amazon S3 Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-s3/>

Bottom of Form

Top of Form

Question 7: Skipped

**You currently operate a sports web portal that covers the latest cricket news in Australia. You manage the main AWS account which has multiple AWS regions. The online application is hosted on a fleet of on-demand EC2 instances and an RDS database which are also deployed to other AWS regions. Your IT Security Compliance Officer has given you the task of developing a reliable and durable logging solution to track changes made to all of your EC2, IAM, and RDS resources in all of the AWS regions. The solution must ensure the integrity and confidentiality of your log data. Which of the following solutions would be the best option to choose?**

* ​

Create three new CloudTrail trails, each with its own S3 bucket to store the logs: one for the AWS Management console, one for AWS SDKs, and one for command line tools. Then create IAM roles and S3 bucket policies for the S3 buckets storing your logs.

* ​

Create a new trail in AWS CloudTrail with the global services option selected, and assign it an existing S3 bucket to store the logs. Create S3 ACLs and enable Multi Factor Authentication (MFA) delete on the S3 bucket storing your logs.

* ​

Create a new trail in CloudTrail and assign it a new S3 bucket to store the logs. Configure AWS SNS to send delivery notifications to your management system. Secure the S3 bucket that stores your logs using IAM roles and S3 bucket policies.

* ​

Create a new trail in AWS CloudTrail with the global services option selected, and create one new Amazon S3 bucket to store the logs. Create IAM roles, S3 bucket policies, and enable Multi Factor Authentication (MFA) Delete on the S3 bucket storing your logs.

**(Correct)**

**Explanation**

For most services, events are recorded in the region where the action occurred to its respective AWS CloudTrail. For global services such as AWS Identity and Access Management (IAM), AWS STS, Amazon CloudFront, and Route 53, events are delivered to any trail that includes global services (IncludeGlobalServiceEvents flag). AWS CloudTrail service should be your top choice for the scenarios where the application is tracking the changes made by any AWS service, resource, or API.

AWS Identity and Access Management (IAM) is integrated with AWS CloudTrail, a service that logs AWS events made by or on behalf of your AWS account. CloudTrail logs authenticated AWS API calls and also AWS sign-in events, and collects this event information in files that are delivered to Amazon S3 buckets.

The option that says: ***Create a new trail in AWS CloudTrail with the global services option selected, and create one new Amazon S3 bucket to store the logs. Create IAM roles, S3 bucket policies, and enable Multi Factor Authentication (MFA) Delete on the S3 bucket storing your logs*** is correct because it uses AWS CloudTrail with (includeGlobalServiceEvents flag) Global Option enabled, a single new S3 bucket and IAM Roles so that it has the confidentiality, MFA on Delete on S3 bucket so that it maintains the data integrity.

The option that says: ***Create a new trail in AWS CloudTrail with the global services option selected, and assign it an existing S3 bucket to store the logs. Create S3 ACLs and enable Multi Factor Authentication (MFA) delete on the S3 bucket storing your logs***is incorrect because as an existing S3 bucket is used, it may already be accessed by the user, hence not maintaining the confidentiality, and it is not using IAM roles.

The option that says: ***Create three new CloudTrail trails, each with its own S3 bucket to store the logs: one for the AWS Management console, one for AWS SDKs, and one for command line tools. Then create IAM roles and S3 bucket policies for the S3 buckets storing your logs*** is incorrect because although it uses AWS CloudTrail, the Global Option is not enabled, and three S3 buckets are not needed.

The option that says: ***Create a new trail in CloudTrail and assign it a new S3 bucket to store the logs. Configure AWS SNS to send delivery notifications to your management system. Secure the S3 bucket that stores your logs using IAM roles and S3 bucket policies*** is incorrect because although it uses AWS CloudTrail, the Global Option is not enabled.

**References**:

<https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-concepts.html#cloudtrail-concepts-global-service-events>

<http://docs.aws.amazon.com/IAM/latest/UserGuide/cloudtrail-integration.html>

**Check out this AWS CloudTrail and IAM Cheat Sheets:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudtrail/>

<https://tutorialsdojo.com/aws-cheat-sheet-aws-identity-and-access-management-iam/>

Bottom of Form

Top of Form

Question 8: Skipped

A company has several IoT enabled devices and sells them to customers around the globe. Every 5 minutes, each IoT device sends back a data file that includes the device status and other information to an Amazon S3 bucket. Every midnight, a Python cron job runs from an Amazon EC2 instance to read and process each data file on the S3 bucket and loads the values on a designated Amazon RDS database. The cron job takes about 10 minutes to process a day’s worth of data. After each data file is processed, it is eventually deleted from the S3 bucket. The company wants to expedite the process and access the processed data on the Amazon RDS as soon as possible.

Which of the following actions would you implement to achieve this requirement with the LEAST amount of effort?

* ​

Convert the Python script cron job to an AWS Lambda function. Configure the Amazon S3 bucket event notifications to trigger the Lambda function whenever an object is uploaded to the bucket.

**(Correct)**

* ​

Increase the Amazon EC2 instance size and spawn more instances to speed up the processing of the data files. Set the Python script cron job schedule to a 1-minute interval to further improve the access time.

* ​

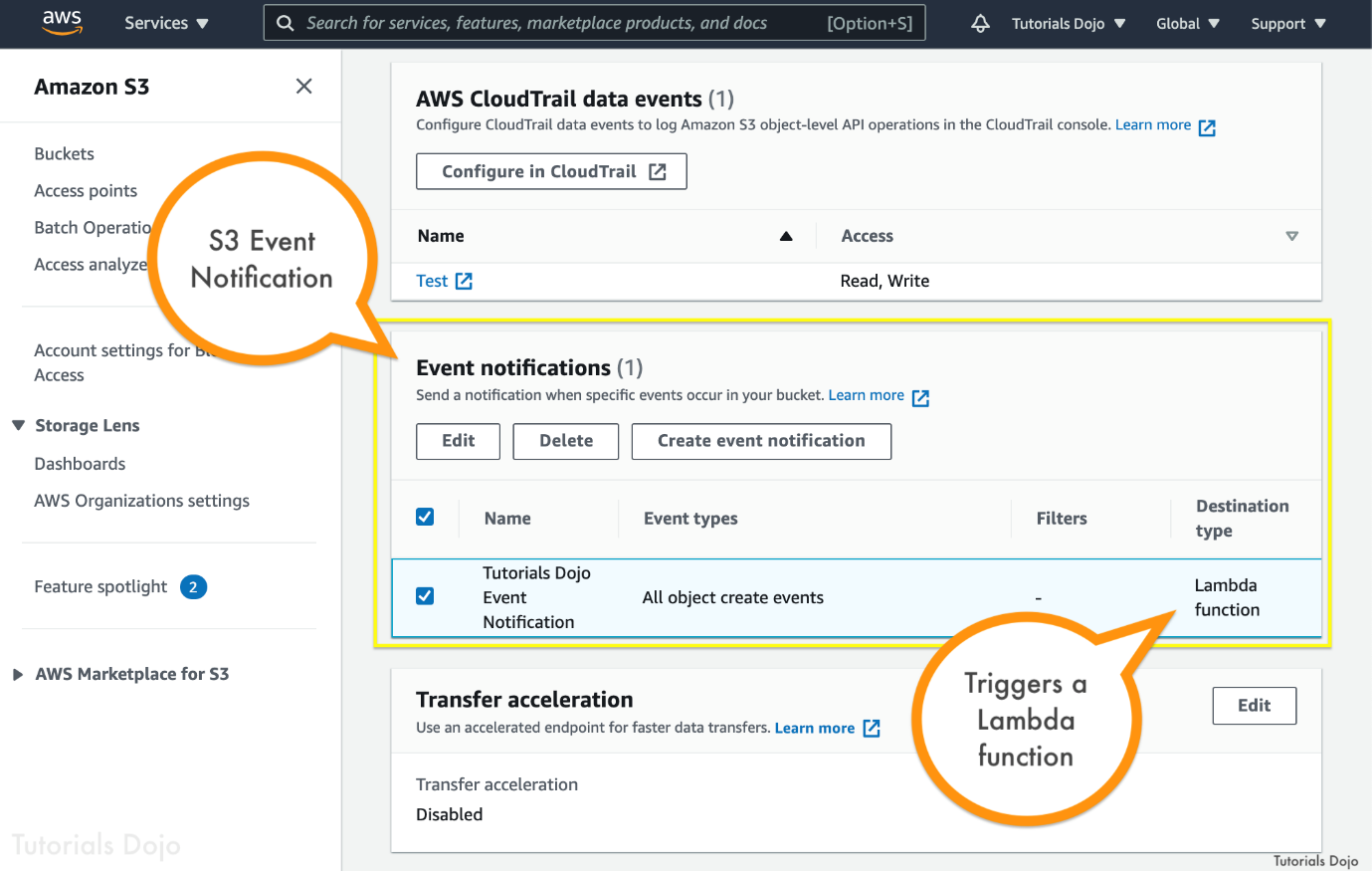
Convert the Python script cron job to an AWS Lambda function. Create an AWS CloudWatch Events rule scheduled at 1-minute intervals and trigger the Lambda function. Create parallel CloudWatch rules that trigger the same Lambda function to further reduce the processing time.

* ​

Convert the Python script cron job to an AWS Lambda function. Configure AWS CloudTrail to log data events of the Amazon S3 bucket. Set up a CloudWatch Events rule to trigger the Lambda function whenever an upload event on the S3 bucket occurs.

**Explanation**

The **Amazon S3 notification feature** enables you to receive notifications when certain events happen in your bucket. To enable notifications, you must first add a notification configuration that identifies the events you want Amazon S3 to publish and the destinations where you want Amazon S3 to send the notifications. You store this configuration in the notification subresource that is associated with a bucket. Amazon S3 event notifications are designed to be delivered at least once. Typically, event notifications are delivered in seconds but can sometimes take a minute or longer.



Currently, Amazon S3 can publish notifications for the following events:

**New object created events** — Amazon S3 supports multiple APIs to create objects. You can request notification when only a specific API is used (for example, s3:ObjectCreated:Put), or you can use a wildcard (for example, s3:ObjectCreated:\*) to request notification when an object is created regardless of the API used.

**Object removal events** — Amazon S3 supports deletes of versioned and unversioned objects. For information about object versioning, see Object Versioning and Using versioning.

**Restore object events** — Amazon S3 supports the restoration of objects archived to the S3 Glacier storage classes. You request to be notified of object restoration completion by using s3:ObjectRestore:Completed. You use s3:ObjectRestore:Post to request notification of the initiation of a restore.

**Reduced Redundancy Storage (RRS) object lost events** — Amazon S3 sends a notification message when it detects that an object of the RRS storage class has been lost.

**Replication events** — Amazon S3 sends event notifications for replication configurations that have S3 Replication Time Control (S3 RTC) enabled. It sends these notifications when an object fails replication, when an object exceeds the 15-minute threshold, when an object is replicated after the 15-minute threshold, and when an object is no longer tracked by replication metrics. It publishes a second event when that object replicates to the destination Region.

Enabling notifications is a bucket-level operation; that is, you store notification configuration information in the notification subresource associated with a bucket. After creating or changing the bucket notification configuration, typically you need to wait 5 minutes for the changes to take effect. Amazon S3 supports the following destinations where it can publish events - Amazon Simple Notification Service (Amazon SNS) topic, Amazon Simple Queue Service (Amazon SQS) queue, and AWS Lambda.

Therefore, the correct answer is: **Convert the Python script cron job to an AWS Lambda function. Configure the Amazon S3 bucket event notifications to trigger the Lambda function whenever an object is uploaded to the bucket** because this provides the best processing and access time. Each of the data files will be processed almost immediately once uploaded on the S3 bucket.

The option that says: **Convert the Python script cron job to an AWS Lambda function. Configure AWS CloudTrail to log data events of the Amazon S3 bucket. Setup a CloudWatch Events rule to trigger the Lambda function whenever an upload event on the S3 bucket occurs**is incorrect. Although this is possible, you do not have to use CloudTrail and CloudWatch Events to satisfy the given requirement. This solution entails a lot of steps. You can simply use the Amazon S3 event notification feature that can trigger the Lambda function directly.

The option that says: **Increase the Amazon EC2 instance size and spawn more instances to speed up the processing of the data files. Set the Python script cron job schedule to a 1-minute interval to further improve the access time**is incorrect. This solution is unreliable since the Amazon EC2 instances can process the same data file at the same time, and because of the limitations of cron, the minimum interval for processing is only 1 minute.

The option that says: **Convert the Python script cron job to an AWS Lambda function. Create an AWS CloudWatch Events rule scheduled at 1-minute intervals and trigger the Lambda function. Create parallel CloudWatch rules that trigger the same Lambda function to further reduce the processing time**is incorrect. The scheduled CloudWatch events rule can only have a minimum of 1-minute intervals. Using Amazon S3 Event notifications as triggers will result in almost near real-time processing of the data files.

**References:**

<https://docs.aws.amazon.com/AmazonS3/latest/dev/NotificationHowTo.html>

<https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html#with-s3-example-configure-event-source>

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/events/ScheduledEvents.html>

**Check out this Amazon S3 Cheat Sheet:**

[https://tutorialsdojo.com/amazon-s3/](https://tutorialsdojo.com/amazon-s3/?src=udemy)

Bottom of Form

Top of Form

Question 9: Skipped

A company is hosting its production environment on its on-premises servers. Most of the applications are packed as Docker containers that are manually run on self-managed virtual machines. The web servers are using the latest commercial Oracle Java SE suite which costs the company thousands of dollars in licensing costs. The MySQL databases are installed on separate servers configured on a master-slave setup for high availability. The company wants to migrate the whole environment to AWS Cloud to take advantage of its flexibility and agility, as well as use OpenJDK to save licensing costs without major changes in their applications.

Which of the following application migration strategies meet this requirement?

* ​

Re-host the environment on the AWS Cloud platform by creating EC2 instances that mirror the current web servers and database servers. Host the Docker instances on Amazon EC2 and test the new OpenJDK Docker containers on these instances. Create a dump of the on-premises MySQL databases and upload it to an Amazon S3 bucket. Launch a new Amazon EC2 instance with a MySQL database and import the data from Amazon S3.

* ​

Retire the current on-premises environment and create a brand new environment on the AWS Cloud platform using AWS AppSync. In this way, you can optimize the application and modify it to be more cost-effective and highly available.

* ​

Re-platform the environment on the AWS Cloud platform by running the Docker containers on Amazon ECS. Test the new OpenJDK Docker containers and upload them on Amazon Elastic Container Registry. Migrate the MySQL database to Amazon RDS using AWS Database Migration Service.

**(Correct)**

* ​

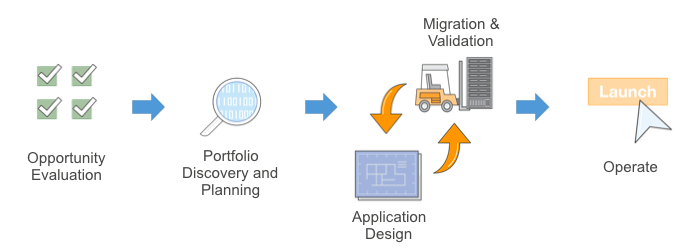
Re-factor/re-architect the environment on AWS Cloud by converting the Docker containers to run on AWS Lambda Functions. Convert the MySQL database to Amazon DynamoDB using the AWS Schema Conversion Tool (AWS SCT) to save on costs.

**Explanation**

The six most common application migration strategies are:

**Rehosting** — Otherwise known as “lift-and-shift”. Many early cloud projects gravitate toward net new development using cloud-native capabilities, but in a large legacy migration scenario where the organization is looking to scale its migration quickly to meet a business case, applications can be rehosted. Most rehosting can be automated with tools (e.g. CloudEndure Migration, AWS VM Import/Export), although you can do this manually to apply changes on legacy systems to the new cloud platform.

**Replatforming** — Sometimes, this is called “lift-tinker-and-shift.” Here you might make a few cloud (or other) optimizations in order to achieve some tangible benefit, but you aren’t otherwise changing the core architecture of the application. You may be looking to reduce the amount of time you spend managing database instances by migrating to a database-as-a-service platform like Amazon Relational Database Service (Amazon RDS), or migrating your application to a fully managed platform like Amazon Elastic Beanstalk.



**Repurchasing** — Moving to a different product. Repurchasing is a move to a SaaS platform. Moving a CRM to Salesforce.com, an HR system to Workday, a CMS to Drupal, etc.

**Refactoring / Re-architecting** — Re-imagining how the application is architected and developed, typically using cloud-native features. This is typically driven by a strong business need to add features, scale, or performance that would otherwise be difficult to achieve in the application’s existing environment. For example, migrating from a monolithic architecture to a service-oriented (or server-less) architecture to boost agility.

**Retire** — This strategy basically means: "Get rid of." Once you’ve discovered everything in your environment, you might ask each functional area who owns each application and see that some of the applications are no longer used. You can save costs by retiring these applications.

**Retain** — Usually this means “revisit” or do nothing (for now). Maybe you aren’t ready to prioritize an application that was recently upgraded or is otherwise not inclined to migrate some applications. You can retain these applications and revisit your migration strategy.

Therefore, the correct answer is:**Re-platform the environment on the AWS Cloud platform by running the Docker containers on Amazon ECS. Test the new OpenJDK Docker containers and upload them on Amazon Elastic Container Registry. Migrate the MySQL database to Amazon RDS using AWS Database Migration Service.**

The option that says: **Re-host the environment on the AWS Cloud platform by creating EC2 instances that mirror the current web servers and database servers. Host the Docker instances on Amazon EC2 and test the new OpenJDK Docker images on these instances. Create a dump of the on-premises MySQL databases and upload it to an Amazon S3 bucket. Launch a new Amazon EC2 instance with a MySQL database and import the data from Amazon S3**is incorrect. Although this is possible, simply re-hosting your applications by mirroring your current on-premises setup does not take advantage of the cloud’s elasticity and agility. A better approach is to use Amazon ECS to run the Docker containers and migrate the MySQL database to Amazon RDS.

The option that says:**Re-factor/re-architect the environment on AWS Cloud by converting the Docker containers to run on AWS Lambda Functions. Convert the MySQL database to Amazon DynamoDB using the AWS Schema Conversion Tool (AWS SCT) to save on costs**is incorrect because this solution requires major changes on the current application to execute successfully. In addition, there is nothing mentioned in the scenario that warrants the conversion of the MySQL database to Amazon DynamoDB.

The option that says:**Retire the current on-premises environment and create a brand new environment on the AWS Cloud platform using AWS AppSync. In this way, you can optimize the application and modify it to be more cost-effective and highly available** is incorrect. This will also require a lot of effort since you will be starting from scratch. Moreover, AWS AppSync is just a service that accelerates application development with scalable GraphQL APIs. The best approach for this scenario is to re-platform the environment using Amazon ECS and Amazon RDS.

**References:**

<https://aws.amazon.com/blogs/enterprise-strategy/6-strategies-for-migrating-applications-to-the-cloud/>

<https://aws.amazon.com/blogs/enterprise-strategy/214-2/>

<https://aws.amazon.com/blogs/enterprise-strategy/considering-a-mass-migration-to-the-cloud/>

**AWS Migration Strategies Cheat Sheet:**

[https://tutorialsdojo.com/aws-migration-strategies-the-6-rs/](https://tutorialsdojo.com/aws-migration-strategies-the-6-rs/?src=udemy)

Bottom of Form

Top of Form

Question 10: Skipped

An online medical record system is using a fleet of Windows EC2 instances with several EBS volumes attached to it. Since the records that they are storing are confidential health files of their patients, there is a need to ensure that the latest security patches are installed to the EC2 instances. In addition, you also have to implement a system in your cloud architecture which checks all of your EC2 instances if they are using an approved Amazon Machine Image (AMI). The system that you will implement should not impede developers from launching instances using an unapproved AMI, but you still have to be notified if there are non-compliant EC2 instances in your VPC.

Which of the following should you implement to protect and monitor all of your instances as required above? (Select TWO.)

* ​

Create an IAM policy that will restrict the developers from launching EC2 instances with an unapproved AMI.

* ​

Use AWS Shield Advanced to automatically patch all of your EC2 instances and detect uncompliant EC2 instances which do not use approved AMIs.

* ​

Use the AWS Config Managed Rule which automatically checks whether your running EC2 instances are using approved AMIs. Set up CloudWatch Alarms to notify you if there are any non-compliant instances running in your VPC.

**(Correct)**

* ​

Set up Amazon GuardDuty that continuously monitors your instances if the latest security patches are installed and if there is an instance that is using an unapproved AMI. Use CloudWatch Alarms to notify you if there are any non-compliant instances running in your VPC.

* ​

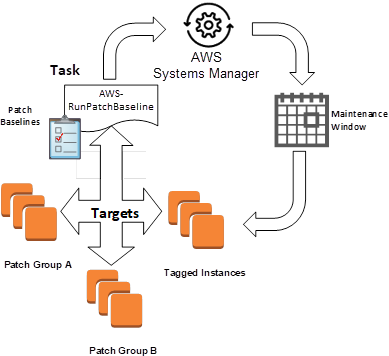
Set up a patch baseline which defines which patches are approved for installation on your instances using AWS Systems Manager Patch Manager.

**(Correct)**

**Explanation**

In this scenario, you can use a combination of AWS Config Managed Rules and AWS Systems Manager Patch Manager to meet the requirements. Hence, the options that say: ***Set up a patch baseline which defines which patches are approved for installation on your instances using AWS Systems Manager Patch Manager***and ***Use the AWS Config Managed Rule which automatically checks whether your running EC2 instances are using approved AMIs. Set up CloudWatch Alarms to notify you if there are any non-compliant instances running in your VPC*** are the correct answers.

AWS Systems Manager Patch Manager automates the process of patching managed instances with security-related updates. For Linux-based instances, you can also install patches for non-security updates. You can patch fleets of Amazon EC2 instances or your on-premises servers and virtual machines (VMs) by operating system type. This includes supported versions of Windows Server, Ubuntu Server, Red Hat Enterprise Linux (RHEL), SUSE Linux Enterprise Server (SLES), CentOS, Amazon Linux, and Amazon Linux 2. You can scan instances to see only a report of missing patches, or you can scan and automatically install all missing patches.



Patch Manager uses *patch baselines*, which include rules for auto-approving patches within days of their release, as well as a list of approved and rejected patches. You can install patches on a regular basis by scheduling patching to run as a Systems Manager Maintenance Window task. You can also install patches individually or to large groups of instances by using Amazon EC2 tags. You can add tags to your patch baselines themselves when you create or update them.

AWS Config provides *AWS managed rules*, which are predefined, customizable rules that AWS Config uses to evaluate whether your AWS resources comply with common best practices. For example, you could use a managed rule to quickly start assessing whether your Amazon Elastic Block Store (Amazon EBS) volumes are encrypted or whether specific tags are applied to your resources. You can set up and activate these rules without writing the code to create an AWS Lambda function, which is required if you want to create custom rules. The AWS Config console guides you through the process of configuring and activating a managed rule. You can also use the AWS Command Line Interface or AWS Config API to pass the JSON code that defines your configuration of a managed rule.

***Creating an IAM policy that will restrict the developers from launching EC2 instances with an unapproved AMI***is incorrect because although you can use an IAM policy to prohibit your developers to launch unapproved AMIs, this will impede their work which violates what the scenario requires. Remember that the scenario says that the system that you will implement should not impede developers to launch instances using an unapproved AMI.

The option that says: ***Set up Amazon GuardDuty that continuously monitors your instances if the latest security patches are installed and if there is an instance that is using an unapproved AMI. Use CloudWatch Alarms to notify you if there are any non-compliant instances running in your VPC*** is incorrect because Amazon GuardDuty is primarily used as a threat detection service that continuously monitors for malicious or unauthorized behavior to help you protect your AWS accounts and workloads. It monitors for activity such as unusual API calls or potentially unauthorized deployments that indicate a possible account compromise however, it does not check if your EC2 instances are using an approved AMI or not.

***Using AWS Shield Advanced to automatically patch all of your EC2 instances and detecting uncompliant EC2 instances which do not use approved AMIs***is incorrect because the AWS Shield Advanced service is most suitable to be used to prevent DDoS attacks in your AWS resources. It cannot check the specific AMIs that your EC2 instances are using.

**References:**

<https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-patch.html>

<https://docs.aws.amazon.com/config/latest/developerguide/evaluate-config_use-managed-rules.html>

<https://docs.aws.amazon.com/config/latest/developerguide/approved-amis-by-id.html>

**Check out these cheat sheets on AWS Config and AWS Systems Manager:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-config/>

<https://tutorialsdojo.com/aws-cheat-sheet-aws-systems-manager/>

Bottom of Form

Top of Form

Question 11: Skipped

Your team had developed an online voting application for a photo competition in AWS using CloudFormation. The application accepts high-quality images of each contestant and stores them in S3 then records the information about the image as well as the contestant's profile in RDS. After the competition, the CloudFormation stack is not used anymore and to save resources, the stack can be terminated. Your manager instructed you to back up the RDS database and the S3 bucket so the data can still be used even after the CloudFormation template is deleted.

Which of the following options is the MOST suitable solution to fulfill this requirement?

* ​

Set the DeletionPolicy on the S3 bucket to snapshot.

* ​

Set the DeletionPolicy to retain on both the RDS and S3 resource types on the CloudFormation template.

* ​

Set the DeletionPolicy on the RDS resource to snapshot and set the S3 bucket to retain.

**(Correct)**

* ​

Set the DeletionPolicy for the RDS instance to snapshot and then enable S3 bucket replication on the source bucket to a destination bucket to maintain a copy of all the S3 objects.

**Explanation**

There are 3 types of DeletionPolicy Options:

Delete

Retain

Snapshot

For Delete, CloudFormation deletes the resource and all its contents if applicable during stack deletion. For Retain, CloudFormation keeps the resource without deleting the resource or its contents when its stack is deleted. For Snapshot, CloudFormation creates a snapshot of the resource before deleting it.

***Setting the DeletionPolicy on the RDS resource to snapshot and setting the S3 bucket to retain*** is correct because it correctly sets the DeletionPolicy of retain on S3 bucket and snapshot on RDS instance.

***Setting the DeletionPolicy for the RDS instance to snapshot and then enabling S3 bucket replication on the source bucket to a destination bucket to maintain a copy of all the S3 objects*** is incorrect because even if the images are backed up to another bucket, the original bucket would be deleted if the CloudFormation stack is deleted. Although this option is valid, it is certainly not the most suitable solution because you don't need to back up the data to another region in the first place. You simply have to set the DeletionPolicy of the S3 bucket to retain.

***Setting the DeletionPolicy to retain on both the RDS and S3 resource types on the CloudFormation template*** is incorrect because DeletionPolicy attribute for RDS should be set to Snapshot and not Retain because with the snapshot option, the backup of the RDS instance would be stored in the form of snapshots. With the retain option, CloudFormation will keep the RDS instance running.

***Setting the DeletionPolicy on the S3 bucket to snapshot*** is incorrect because the DeletionPolicy of the S3 bucket should be retained, not snapshot.

**Reference:**

<http://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-deletionpolicy.html>

**Check out this AWS CloudFormation Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-cloudformation/>

Bottom of Form

Top of Form

Question 12: Skipped

A company has several development teams using AWS CodeCommit to store their source code. With the number of code updates every day, the management is having difficulty tracking if the developers are adhering to company security policies. On a recent audit, the security team found several IAM access keys and secret keys in the CodeCommit repository. This is a big security risk so the company wants to have an automated solution that will scan the CodeCommit repositories for committed IAM credentials and delete/disable the IAM keys for those users.

Which of the following options will meet the company requirements?

* ​

Using a development instance, use the AWS Systems Manager Run Command to scan the AWS CodeCommit repository for IAM credentials on a daily basis. If credentials are found, rotate them using AWS Secrets Manager. Notify the user of the violation.

* ​

Scan the CodeCommit repositories for IAM credentials using Amazon Macie. Using machine learning, Amazon Macie can scan your repository for security violations. If violations are found, invoke an AWS Lambda function to notify the user and delete the IAM keys.

* ​

Write a custom AWS Lambda function to search for credentials on new code submissions. Set the function trigger as AWS CodeCommit push events. If credentials are found, notify the user of the violation, and disable the IAM keys.

**(Correct)**

* ​

Download and scan the source code from AWS CodeCommit using a custom AWS Lambda function. Schedule this Lambda function to run daily. If credentials are found, notify the user of the violation, generate new IAM credentials and store them in AWS KMS for encryption.

**Explanation**

**AWS CodeCommit** is a version control service hosted by Amazon Web Services that you can use to privately store and manage assets (such as documents, source code, and binary files) in the cloud.

You can configure a CodeCommit repository so that code pushes or other events trigger actions, such as sending a notification from Amazon Simple Notification Service (Amazon SNS) or invoking a function in AWS Lambda. You can create up to 10 triggers for each CodeCommit repository.

Triggers are commonly configured to:

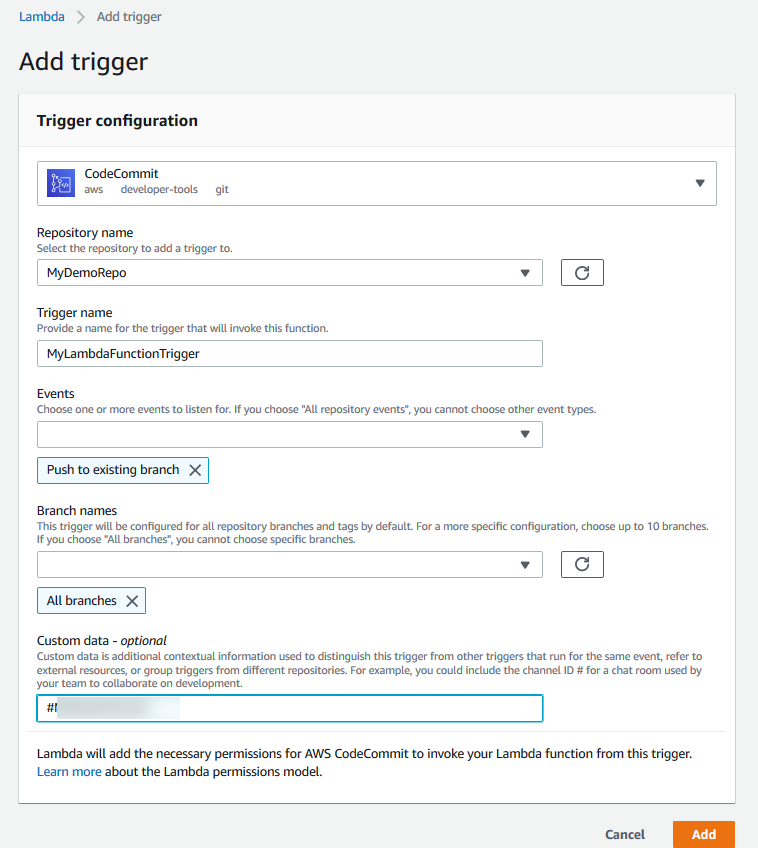
- Send emails to subscribed users every time someone pushes to the repository.

- Notify an external build system to start a build after someone pushes to the main branch of the repository.

Scenarios like notifying an external build system require writing a Lambda function to interact with other applications. The email scenario simply requires creating an Amazon SNS topic. You can create a trigger for a CodeCommit repository so that events in that repository trigger notifications from an Amazon Simple Notification Service (Amazon SNS) topic.

You can also create an AWS Lambda trigger for a CodeCommit repository so that events in the repository invoke a Lambda function. For example, you can create a Lambda function that will scan the CodeCommit code submissions for IAM credentials, and then send out notifications or perform corrective actions.

When you use the Lambda console to create the function, you can create a CodeCommit trigger for the Lambda function. Here is an example of the trigger for all push events:



Therefore, the correct answer is: **Write a custom AWS Lambda function to search for credentials on new code submissions. Set the function trigger as AWS CodeCommit push events. If credentials are found, notify the user of the violation, and disable the IAM keys.**

The option that says: **Using a development instance, use the AWS Systems Manager Run Command to scan the AWS CodeCommit repository for IAM credentials on a daily basis. If credentials are found, rotate them using AWS Secrets Manager. Notify the user of the violation** is incorrect. You cannot rotate IAM keys on AWS Secrets Manager. Using the Run Command on a development instance just for scanning the repository is costly. It is cheaper to just write your own Lambda function to do the scanning.

The option that says: **Download and scan the source code from AWS CodeCommit using a custom AWS Lambda function. Schedule this Lambda function to run daily. If credentials are found, notify the user of the violation, generate new IAM credentials and store them in AWS KMS for encryption** is incorrect. You store encryption keys on AWS KMS, not IAM keys.

The option that says: **Scan the CodeCommit repositories for IAM credentials using Amazon Macie. Using machine learning, Amazon Macie can scan your repository for security violations. If violations are found, invoke an AWS Lambda function to notify the user and delete the IAM keys** is incorrect. Amazon Macie is designed to use machine learning and pattern matching to discover and protect your sensitive data in AWS. Macie primarily scans Amazon S3 buckets for data security and data privacy.

**References:**

<https://docs.aws.amazon.com/codecommit/latest/userguide/how-to-notify-lambda.html>

<https://docs.aws.amazon.com/codecommit/latest/userguide/how-to-notify-sns.html>

<https://docs.aws.amazon.com/codecommit/latest/userguide/how-to-notify.html>

**Check out the AWS CodeCommit Cheat Sheet:**

[https://tutorialsdojo.com/aws-codecommit/](https://tutorialsdojo.com/aws-codecommit/?src=udemy)

Bottom of Form

Top of Form

Question 13: Skipped

A company regularly processes large product catalogs for its online retail platform, which is needed to index and extract metadata of its items. These are done in batches and are sent out to a small team to process them using the Amazon Mechanical Turk service. The Solutions Architect has been tasked to design a workflow orchestration process to allow multiple concurrent Mechanical Turk operations while dealing with the result assessment process and the ability to reprocess the failed jobs.

Which of the following solutions will allow the company to visualize and control the state of every workflow with the LEAST amount of effort?

* ​

Create a workflow on Amazon SWF that will handle a single batch of catalog records. Create multiple worker tasks that will extract and transform the data before sending it through Amazon Mechanical Turk. To visualize the workflow states, process the logs using AWS Lambda functions and use Amazon Elasticsearch and Kibana.

**(Correct)**

* ​

Run a Lambda function to regularly poll for status changes in an Amazon RDS database that stores the workflow information. Create worker Lambda functions that will then process the next workflow steps. Visualize the workflow states using Amazon QuickSight on the Amazon RDS database.

* ​

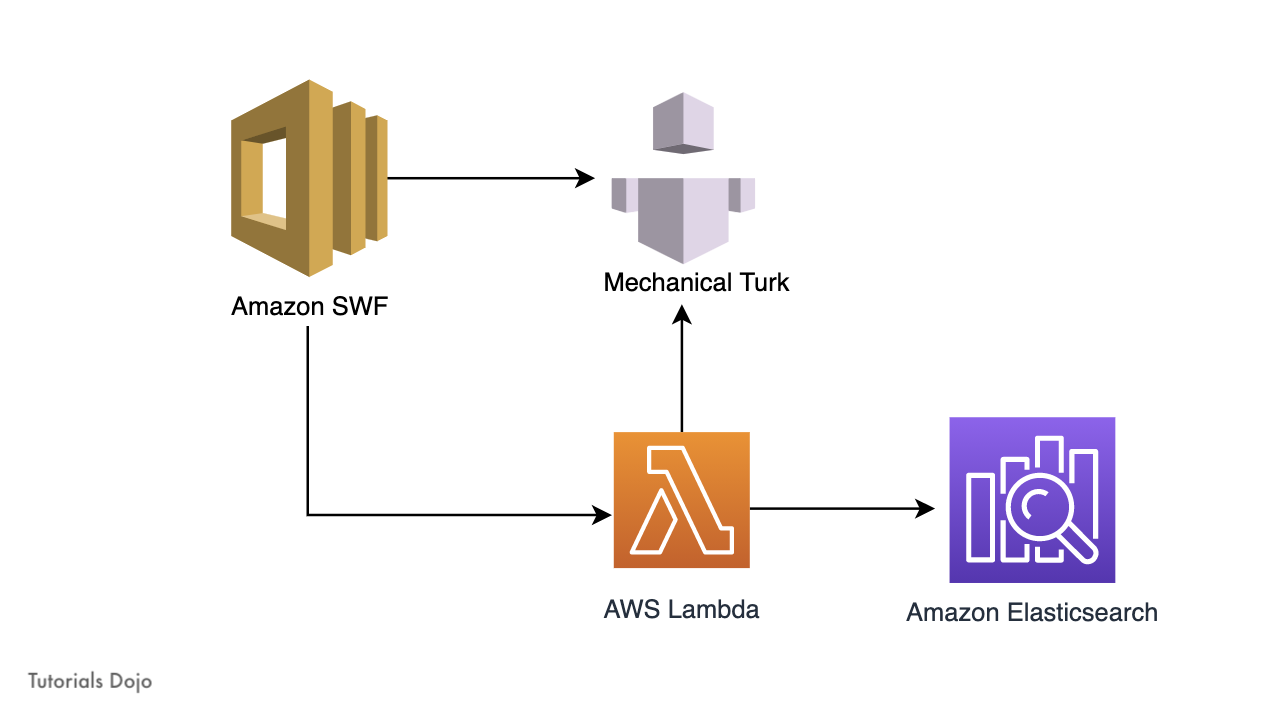
With AWS Step Functions, create a workflow that will orchestrate multiple concurrent workflows. Visualize each workflow status on the AWS Management Console and write the historical data on an Amazon S3 bucket. Use Amazon QuickSight to visualize the data on the S3 bucket.

* ​

Create a queue per workflow stage on Amazon SQS and trigger an Amazon CloudWatch Alarm based on the message visibility on each queue. Send messages using Amazon SNS to trigger the AWS Lambda functions that process the next step. Visualize the Lambda processing logs for each workflow state using Amazon Elasticsearch and Kibana.

**Explanation**

While validating data in large catalogs, the products in the catalog are processed in batches. Different batches can be processed concurrently. For each batch, the product data is extracted from servers in the datacenter and transformed into CSV (Comma Separated Values) files required by Amazon Mechanical Turk’s Requester User Interface (RUI). The CSV is uploaded to populate and run the HITs (Human Intelligence Tasks). When HITs complete, the resulting CSV file is reverse transformed to get the data back into the original format. The results are then assessed and Amazon Mechanical Turk workers are paid for acceptable results. Failures are weeded out and reprocessed, while the acceptable HIT results are used to update the catalog. As batches are processed, the system needs to track the quality of the Amazon Mechanical Turk workers and adjust the payments accordingly. Failed HITs are re-batched and sent through the pipeline again.



With Amazon SWF: The use case above is implemented as a set of workflows. A **BatchProcess** workflow handles the processing for a single batch. It has workers that extract the data, transform it, and send it through Amazon Mechanical Turk. The **BatchProcess** workflow outputs the acceptable HITs and the failed ones. This is used as the input for three other workflows: **MTurkManager**, **UpdateCatalogWorkflow**, and **RerunProducts**.

The **MTurkManager** workflow makes payments for acceptable HITs, responds to the human workers who produced failed HITs, and updates its own database for tracking results quality. The **UpdateCatalogWorkflow** updates the master catalog based on acceptable HITs. The **RerunProducts** workflow waits until there is a large enough batch of products with failed HITs. It then creates a batch and sends it back to the **BatchProcess** workflow. The entire end-to-end catalog processing is performed by a **CleanupCatalog** workflow that initiates child executions of the above workflows. Having a system of well-defined workflows enables this use case to be architected, audited, and run systematically for catalogs with several million products.

The scenario for this question is similar to the Amazon Simple Workflow Service FAQs. See the FAQs Use Case #2: [*Processing large product catalogs using Amazon Mechanical Turk.*](https://aws.amazon.com/swf/faqs/#:~:text=Processing%20large%20product%20catalogs%20using%20Amazon%20Mechanical%20Turk)

Therefore, the correct answer is:**Create a workflow on Amazon SWF that will handle a single batch of catalog records. Create multiple worker tasks that will extract and transform the data before sending it through Amazon Mechanical Turk. To visualize the workflow states, process the logs using AWS Lambda functions and use Amazon Elasticsearch and Kibana.**

The option that says: **Create a queue per workflow stage on Amazon SQS and trigger an Amazon CloudWatch Alarm based on the message visibility on each queue. Send messages using Amazon SNS to trigger the AWS Lambda functions that process the next step. Visualize the Lambda processing logs for each workflow state using Amazon Elasticsearch and Kibana**is incorrect because it requires a lot of effort to configure each component on various stages. The reliance on multiple AWS services adds complexity and a chance for the workflow to encounter an error when running.

The option that says: **Run a Lambda function to regularly poll for status changes in an Amazon RDS database that stores the workflow information. Create worker Lambda functions that will then process the next workflow steps. Visualize the workflow states using Amazon QuickSight on the Amazon RDS database** is incorrect. This is inefficient as the Lambda function constantly polls the RDS database for changes. Additionally, this incurs unnecessary costs as you keep the RDS instance running even if there are no batch jobs to be processed.

The option that says: **With AWS Step Functions, create a workflow that will orchestrate multiple concurrent workflows. Visualize each workflow status on the AWS Management Console, and write the historical data on an Amazon S3 bucket. Use Amazon QuickSight to visualize the data on the S3 bucket**is incorrect because Step Functions do not directly support Mechanical Turk. You will need to use Amazon SWF for this scenario.

**References:**

<https://aws.amazon.com/swf/faqs/>

<https://aws.amazon.com/swf/>

<https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMechanicalTurkGettingStartedGuide/SvcIntro.html>

<https://docs.aws.amazon.com/AWSMechTurk/latest/AWSMechanicalTurkGettingStartedGuide/NextSteps.html>

**Check out these Cheat Sheets:**

[https://tutorialsdojo.com/amazon-simple-workflow-amazon-swf/](https://tutorialsdojo.com/amazon-simple-workflow-amazon-swf/%20?src=udemy)

[https://tutorialsdojo.com/amazon-mechanical-turk/](https://tutorialsdojo.com/amazon-mechanical-turk/?src=udemy)

Bottom of Form

Top of Form

Question 14: Skipped

A company has a fitness tracking app that accompanies its smartwatch. The primary customers are North American and Asian users. The application is read-heavy as it pings the servers at regular intervals for user-authorization. The company wants the infrastructure to have the following capabilities:

- The application must be fault-tolerant to problems in any Region.

- The database writes must be highly-available in a single Region.

- The application tier must be able to read the database on multiple Regions.

- The application tier must be resilient in each Region.

- Relational database semantics must be reflected in the application.

Which of the following options must the Solutions Architect implement to meet the company requirements? (Select TWO.)

* ​

Deploy the application tier on an Auto Scaling group of EC2 instances for each Region. Create an RDS for MySQL database on each region. Configure the application to perform read/write operations on the local RDS. Enable cross-Region replication for the database servers. Create snapshots of the application and database servers regularly. Store the snapshots in Amazon S3 buckets in both regions.

* ​

Create a geoproximity routing policy on Amazon Route 53 to control traffic and direct users to their closest regional endpoint. Combine this with a multivalue answer routing policy with health checks to direct users to a healthy region at any given time.

* ​

Deploy the application tier on an Auto Scaling group of EC2 instances for each Region in an active-active configuration. Create a cluster of Amazon Aurora global database in both Regions. Configure the application to use the in-Region Aurora database endpoint for the read/write operations. Create snapshots of the application servers regularly. Store the snapshots in Amazon S3 buckets in both regions.

**(Correct)**

* ​

Deploy the application tier on an Auto Scaling group of EC2 instances for each Region. Create an RDS for MySQL database on each region. Configure the application to perform read/write operations on the local RDS. Enable Multi-AZ failover support for the EC2 Auto Scaling group and the RDS database. Enable cross-Region replication for the database servers.

* ​

Create a geolocation routing policy on Amazon Route 53 to point the global users to their designated regions. Combine this with a failover answer routing policy with health checks to direct users to a healthy region at any given time.

**(Correct)**

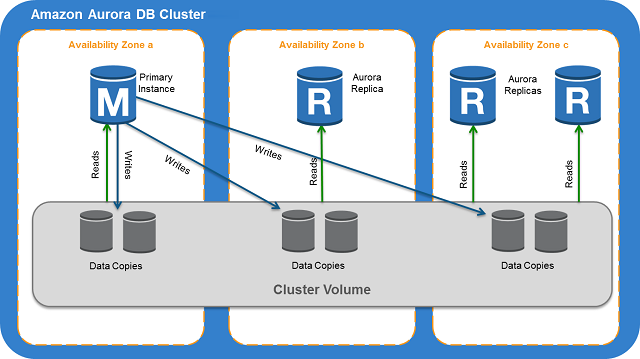
**Explanation**

**Amazon Aurora Global Database** is designed for globally distributed applications, allowing a single Amazon Aurora database to span multiple AWS regions. It replicates your data with no impact on database performance, enables fast local reads with low latency in each region, and provides disaster recovery from region-wide outages.

By using an Amazon Aurora global database, you can have a single Aurora database that spans multiple AWS Regions to support your globally distributed applications.

An Aurora global database consists of one primary AWS Region where your data is mastered, and up to five read-only secondary AWS Regions. You issue write operations directly to the primary DB cluster in the primary AWS Region. Aurora replicates data to the secondary AWS Regions using dedicated infrastructure, with latency typically under a second.

You can also change the configuration of your Aurora global database while it's running to support various use cases. For example, you might want the read/write capabilities to move from one Region to another, say, in different time zones, to 'follow the sun.' Or, you might need to respond to an outage in one Region. With Aurora global database, you can promote one of the secondary Regions to the primary role to take full read/write workloads in under a minute.



On **Amazon Route 53**, after you create a hosted zone for your domain, such as tutorialsdojo.com, you can create records to tell the Domain Name System (DNS) how you want traffic to be routed for that domain. You can create a record that points to the DNS name of your Application Load Balancer on AWS.

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

**Simple routing policy** – Use for a single resource that performs a given function for your domain, for example, a web server that serves content for the example.com website.

**Failover routing policy** – Use when you want to configure active-passive failover.

**Geolocation routing policy** – Use when you want to route traffic based on the location of your users.

**Geoproximity routing policy** – Use when you want to route traffic based on the location of your resources and, optionally, shift traffic from resources in one location to resources in another.

**Latency routing policy**– Use when you have resources in multiple AWS Regions and you want to route traffic to the region that provides the best latency.

**Multivalue answer routing policy** – Use when you want Route 53 to respond to DNS queries with up to eight healthy records selected at random.

**Weighted routing policy** – Use to route traffic to multiple resources in proportions that you specify.

You can use Route 53 health checks to configure active-active and active-passive failover configurations. You configure active-active failover using any routing policy (or combination of routing policies) other than failover, and you configure active-passive failover using the failover routing policy.

The option that says: **Create a geolocation routing policy on Amazon Route 53 to point the global users to their designated regions. Combine this with a failover answer routing policy with health checks to direct users to a healthy region at any given time** is correct. You can use geolocation routing policy to direct the North American users to your servers on the North America region and configure failover routing to the Asia region in case the North America region fails. You can configure the same for the Asian users pointed to the Asia region servers and have the North America region as its backup.

The option that says: **Deploy the application tier on an Auto Scaling group of EC2 instances for each Region in an active-active configuration. Create a cluster of Amazon Aurora global database in both Regions. Configure the application to use the in-Region Aurora database endpoint for the read/write operations. Create snapshots of the application servers regularly. Store the snapshots in Amazon S3 buckets in both regions** is correct. The Amazon Aurora global database solves the problem on read/write as well as syncing the data across all the regions. With both regions in an active-active configuration, each region can accept the traffic from users around the world.

The option that says: **Create a geoproximity routing policy on Amazon Route 53 to control traffic and direct users to their closest regional endpoint. Combine this with a multivalue answer routing policy with health checks to direct users to a healthy region at any given time** is incorrect. Geoproximity routing policy is good to control the user traffic to specific regions. However, a multivalue answer routing policy may cause the users to be randomly sent to other healthy regions that may be far away from the user’s location.

The option that says: **Deploy the application tier on an Auto Scaling group of EC2 instances for each Region. Create an RDS for MySQL database on each region. Configure the application to perform read/write operations on the local RDS. Enable cross-Region replication for the database servers. Create snapshots of the application and database servers regularly. Store the snapshots in Amazon S3 buckets in both regions** is incorrect. You can’t sync two MySQL master databases that both accept writes on their respective regions.

The option that says: **Deploy the application tier on an Auto Scaling group of EC2 instances for each Region. Create an RDS for MySQL database on each region. Configure the application to perform read/write operations on the local RDS. Enable Multi-AZ failover support for the EC2 Auto Scaling group and the RDS database. Enable cross-Region replication for the database servers** is incorrect. You can’t sync two MySQL master databases that both accept writes on their respective regions. Enabling Multi-AZ on the RDS MySQL server does not protect you from AWS Regional failures.

**References:**

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

<https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-global-database.html#aurora-global-database-overview>

<https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Overview.html>

<https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/aurora-global-database-connecting.html>

**Check out the Amazon Aurora and Route 53 Cheat Sheets:**

[https://tutorialsdojo.com/amazon-aurora/](https://tutorialsdojo.com/amazon-aurora/?src=udemy)

[https://tutorialsdojo.com/amazon-route-53/](https://tutorialsdojo.com/amazon-route-53/?src=udemy)

Bottom of Form

Top of Form

Question 15: Skipped

A company is hosting its flagship product page on a three-tier web application in its on-premises data center. The popularity of the last product launch attracted a sudden surge of traffic to their site, which caused some downtime that resulted in a significant impact on the product’s sales volume. The management decided to move the application to AWS. The application uses a MySQL database and is written in .NET framework. The Solutions Architect must design a highly available and scalable infrastructure to handle the demand of 300,000 peak users.

Which of the following design options would satisfy the above requirements while being cost-effective?

* ​

Create an AWS Elastic Beanstalk application with an Auto Scaling group of EC2 instances as web servers that spans two separate regions. Put the EC2 instances behind an Application Load Balancer in each region. Launch a Multi-AZ Amazon Aurora MySQL database with cross-region read replica to the other region. Create zone entries in Route 53 with geoproximity routing policy to direct the traffic between the two regions.

* ​

Launch a CloudFormation stack that contains an Auto Scaling Group of Amazon EC2 instances spanning multiple Availability Zones that are behind an Application Load Balancer. Use the stack to launch an Amazon Aurora MySQL database cluster in a Multi-AZ configuration with a “retain” deletion policy. Create a Route 53 zone entry for the company’s domain name with an Alias-record pointed to the ALB.

**(Correct)**

* ​

Create an AWS Elastic Beanstalk application that contains a web server tier and an Amazon RDS MySQL Multi-AZ database tier. The web server tier should launch a fleet of Amazon EC2 Auto Scaling Group spanning multiple Availability Zones and behind a Network Load Balancer. Create a Route 53 zone entry for the company’s domain name with an Alias-record pointed to the NLB.

* ​

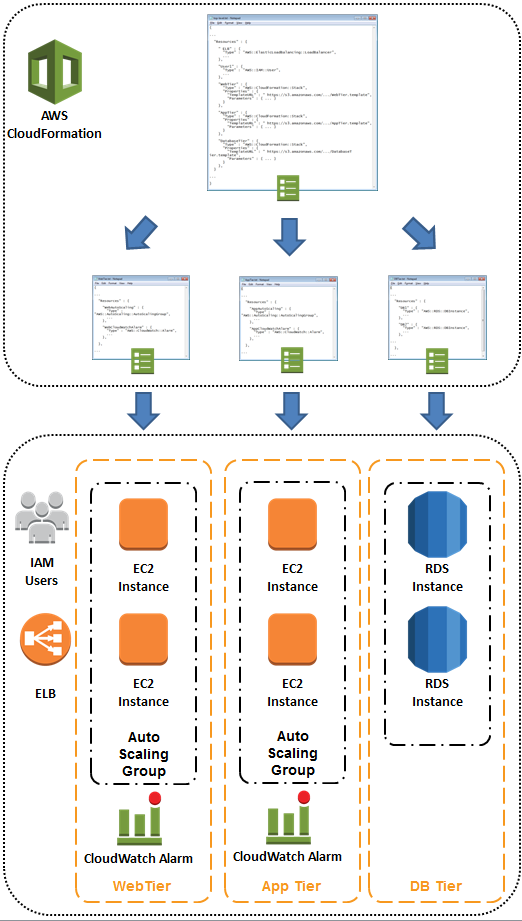
Launch a CloudFormation stack that contains an Amazon ECS cluster that spans multiple Availability Zones using Spot Instances. Create an Application Load Balancer in front of the ECS cluster. Use the stack to launch an Amazon RDS MySQL database in Multi-AZ configuration with a “snapshot” deletion policy. Create a Route 53 zone entry for the company’s domain name with an Alias-record pointed to the ALB.

**Explanation**

**AWS CloudFormation** gives you an easy way to model a collection of related AWS resources, provision them quickly and consistently, and manage them throughout their lifecycles, by treating infrastructure as code. A CloudFormation template describes your desired resources and their dependencies so you can launch and configure them together as a stack.

In CloudFormation, the AWS::AutoScaling::AutoScalingGroup resource defines an Amazon EC2 Auto Scaling group, which is a collection of Amazon EC2 instances that are treated as a logical grouping for the purposes of automatic scaling and management.

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.



To use Elastic Load Balancing with your Auto Scaling group, you attach the load balancer to your Auto Scaling group to register the group with the load balancer. Your load balancer acts as a single point of contact for all incoming web traffic to your Auto Scaling group. An Application Load Balancer is ideal for this scenario as routes and load balances at the application layer (HTTP/HTTPS) and supports path-based routing.

On CloudFormation you can set the RDS DeletionPolicy as “Retain” which keeps the resource without deleting it or its contents when its stack is deleted. This is helpful in the event that the stack is deleted and you need to quickly provision a new stack. You can quickly use the retained RDS instance from the old stack.

Therefore, the correct answer is:**Launch a CloudFormation stack that contains an Auto Scaling Group of Amazon EC2 instances spanning multiple Availability Zones that are behind an Application Load Balancer. Use the stack to launch an Amazon Aurora MySQL database cluster in a Multi-AZ configuration with a “retain” deletion policy. Create a Route 53 zone entry for the company’s domain name with an Alias-record pointed to the ALB.**

The option that says: **Create an AWS Elastic Beanstalk application that contains a web server tier and an Amazon RDS MySQL Multi-AZ database tier. The web server tier should launch a fleet of Amazon EC2 Auto Scaling Group spanning multiple Availability Zones and behind a Network Load Balancer. Create a Route 53 zone entry for the company’s domain name with an Alias-record pointed to the NLB**is incorrect because you do not need an expensive Network Load Balancer (NLB) to handle the expected peak traffic. An NLB is a good choice if you expect millions of requests per second.

The option that says: **Create an AWS Elastic Beanstalk application with an Auto Scaling group of EC2 instances as web servers that spans two separate regions. Put the EC2 instances behind an Application Load Balancer in each region. Launch a Multi-AZ Amazon Aurora MySQL database with cross-region read replica to the other region. Create zone entries in Route 53 with geoproximity routing policy to direct the traffic between the two regions** is incorrect because creating two Auto Scaling groups on separate regions is unnecessary and expensive. Distributing the EC2 instance in multiple Availability Zones is enough to handle the traffic. In this setup, the database on the second region is a read-replica only so any writes to the database will have to be sent on the main region’s RDS instance.

The option that says: **Launch a CloudFormation stack that contains an Amazon ECS cluster that spans multiple Availability Zones using Spot Instances. Create an Application Load Balancer in front of the ECS cluster. Use the stack to launch an Amazon RDS MySQL database in Multi-AZ configuration with a “snapshot” deletion policy. Create a Route 53 zone entry for the company’s domain name with an Alias-record pointed to the ALB** is incorrect. Although the Spot instance provides good cost savings for the web tier, the reliability of the site will suffer as the Spot instances are usually reclaimed by AWS based on the supply and demand of its global computing capacity. The “snapshot” deletion policy on the database tier is also not ideal as this will require a significant time to restore if you delete the CloudFormation stack.

**References:**

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

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-properties-as-group.html>

<https://docs.aws.amazon.com/autoscaling/ec2/userguide/autoscaling-load-balancer.html>

<https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/aws-attribute-deletionpolicy.html>

<https://aws.amazon.com/blogs/aws/parallel-stack-processing-and-nested-stack-updates-for-aws-cloudformation/>

**Check out these Cheat Sheets:**

[https://tutorialsdojo.com/amazon-relational-database-service-amazon-rds/](https://tutorialsdojo.com/amazon-relational-database-service-amazon-rds/?src=udemy)

[https://tutorialsdojo.com/aws-elastic-load-balancing-elb/](https://tutorialsdojo.com/aws-elastic-load-balancing-elb/?src=udemy)

[https://tutorialsdojo.com/aws-cloudformation/](https://tutorialsdojo.com/aws-cloudformation/?src=udemy)

Bottom of Form

Top of Form

Question 16: Skipped

A game development startup hired you to design the cloud architecture of an online role-playing game, which has an extra leaderboard that displays the top 10 scorers of the day. The leaderboard is updated real-time using Redis sorted sets which sorts and re-ranks the data. In case of an outage, a Lambda function can re-generate all of the data to the cluster. A Solutions Architect is instructed to build a cost-effective and fault tolerant caching layer that enhances the data durability of your ElastiCache cluster.

Which of the following is the MOST suitable and cost-effective solution that the Architect must implement to meet the above requirement?

* ​

Purchase extra large reserved nodes for your new ElastiCache Redis cluster and enable the Multi-AZ with Auto-Failover option upon launch.

* ​

Enable automatic backups and set the backup retention period to maximum.

* ​

Integrate your ElastiCache with Amazon ES then enable Kibana and LogStash to re-generate all of the data to the Amazon ElastiCache cluster in the event of an outage.

* ​

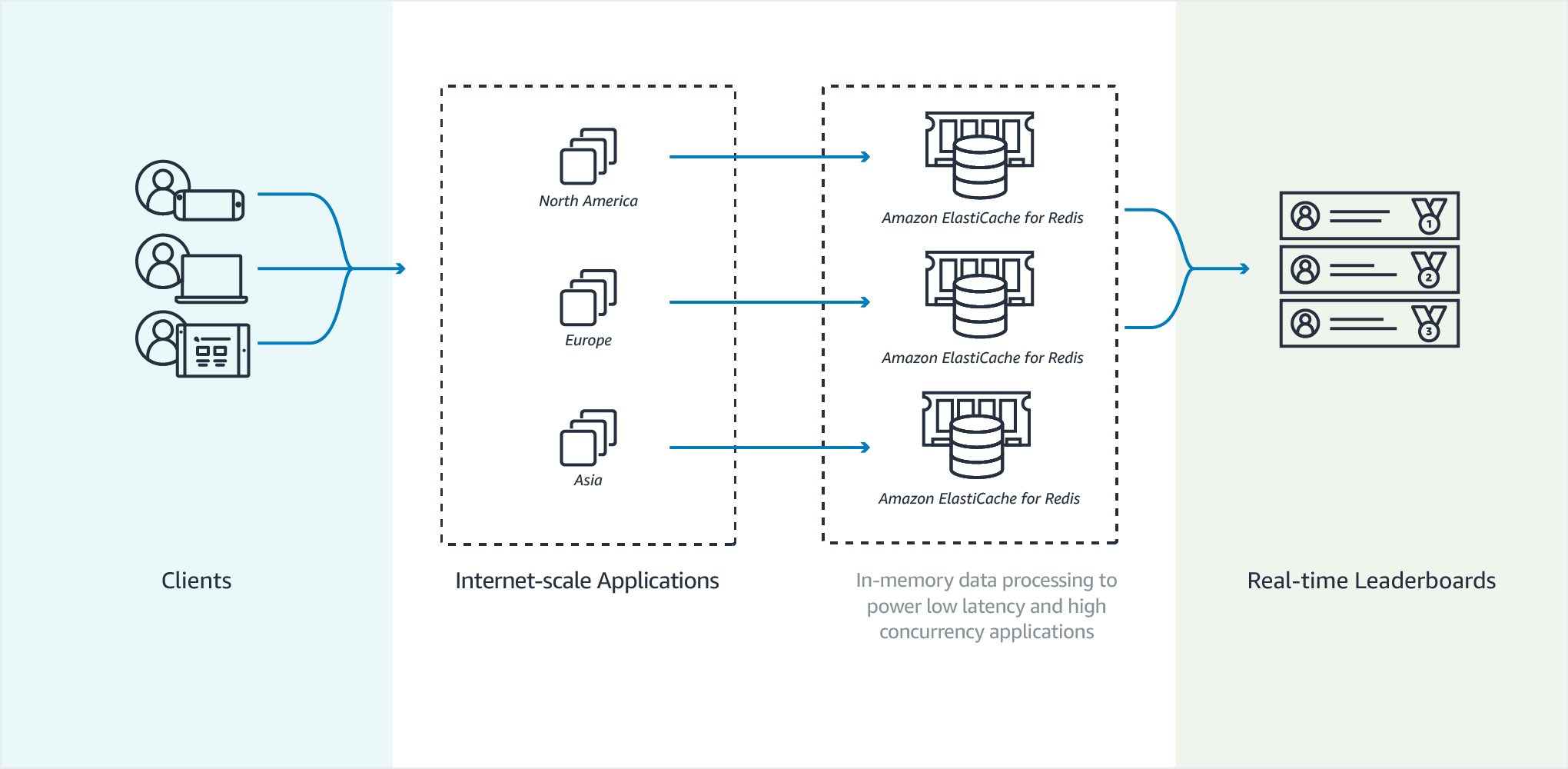
Use the Redis append-only file feature (AOF) to record all of the commands that change cache data to an append-only file. When a node is rebooted and the cache engine starts, the AOF is "replayed" and the result is a warm Redis cache with all of the data intact.

**(Correct)**

**Explanation**

An ElastiCache Redis cluster provides varying levels of data durability, performance, and cost for implementing disaster recovery or fault tolerance of your cached data. You can choose the following options to improve the data durability of your ElastiCache cluster:

- Daily automatic backups  
- Manual backups using Redis append-only file (AOF)  
- Setting up a Multi-AZ with Automatic Failover



By default, the data in a Redis node on ElastiCache resides only in memory and is not persistent. If a node is rebooted, or if the underlying physical server experiences a hardware failure, the data in the cache is lost.

If you require data durability, you can **enable the Redis append-only file feature (AOF)**. When this feature is enabled, the node writes all of the commands that change cache data to an append-only file. When a node is rebooted and the cache engine starts, the AOF is "replayed"; the result is a warm Redis cache with all of the data intact.

AOF is disabled by default. To enable AOF for a cluster running Redis, you must create a parameter group with the appendonly parameter set to yes, and then assign that parameter group to your cluster. You can also modify the appendfsync parameter to control how often Redis writes to the AOF file.

Hence, the correct answer is: **Use the Redis append-only file feature (AOF) to record all of the commands that change cache data to an append-only file. When a node is rebooted and the cache engine starts, the AOF is "replayed" and the result is a warm Redis cache with all of the data intact.**

**Purchasing extra large reserved nodes for your new ElastiCache Redis cluster and enabling the Multi-AZ with Auto-Failover option upon launch** is incorrect because although setting up a Multi-AZ with automatic failover can optimally improve the data durability of your ElastiCache layer, this option is the most expensive one since you bought reserved nodes with extra large EC2 instances which will increase your overall cost. The top scorers data can easily be regenerated by the Lambda function so simply using Redis AOF is enough to meet the requirement.

**Enabling automatic backups and setting the backup retention period to maximum** is incorrect because the data loss potential for daily scheduled backups is high (up to a day’s worth of data). Since you will only backup your ElastiCache once a day, there will be hours of gap in between the scheduled backup. If you scheduled the backup every 1 AM and the outage happened in the middle of the day (1 PM), then the data loss will be up to 12 hours.

**Integrating your ElastiCache with Amazon ES then enabling Kibana and LogStash to re-generate all of the data to the Amazon ElastiCache cluster in the event of an outage**is incorrect because integrating ElastiCache and Amazon ES (ElasticSearch) will not improve the data durability of your cluster. Logstash simply provides a convenient way to use the bulk API to upload data into your Amazon ES domain with the S3 plugin while Kibana is just an open-source visualization tool designed to work with Elasticsearch.

**References:**

<https://aws.amazon.com/premiumsupport/knowledge-center/fault-tolerance-elasticache/>

<https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/AutoFailover.html>

<https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/RedisAOF.html>

**Check out this Amazon Elasticache Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-elasticache/>

**Redis Append-Only Files vs Redis Replication:**

<https://tutorialsdojo.com/aws-cheat-sheet-redis-append-only-files-vs-redis-replication/>

**Comparison of AWS Services Cheat Sheets:**

<https://tutorialsdojo.com/comparison-of-aws-services-for-udemy-students/>

Bottom of Form

Top of Form

Question 17: Skipped

A media company has a suite of internet-facing web applications hosted in US West (N. California) region in AWS. The architecture is composed of several On-Demand Amazon EC2 instances behind an Application Load Balancer, which is configured to use public SSL/TLS certificates. The Application Load Balancer also enables incoming HTTPS traffic through the fully qualified domain names (FQDNs) of the applications for SSL termination. A Solutions Architect has been instructed to upgrade the corporate web applications to a multi-region architecture that uses various AWS Regions such as ap-southeast-2, ca-central-1, eu-west-3, and so forth.

Which of the following approach should the Architect implement to ensure that all HTTPS services will continue to work without interruption?

* ​

In each new AWS Region, request for SSL/TLS certificates using the AWS Certificate Manager for each FQDN. Associate the new certificates to the corresponding Application Load Balancer of the same AWS Region.

**(Correct)**

* ​

Use the AWS Certificate Manager service in the US West (N. California) region to request for SSL/TLS certificates for each FQDN which will be used to all regions. Associate the new certificates to the new Application Load Balancer on each new AWS Region that the Architect will add.

* ​

In each new AWS Region, request for SSL/TLS certificates using AWS KMS for each FQDN. Associate the new certificates to the corresponding Application Load Balancer of the same AWS Region.

* ​

Use the AWS KMS in the US West (N. California) region to request for SSL/TLS certificates for each FQDN which will be used to all regions. Associate the new certificates to the new Application Load Balancer on each new AWS Region that the Architect will add.

**Explanation**

**AWS Certificate Manager** is a service that lets you easily provision, manage, and deploy public and private Secure Sockets Layer/Transport Layer Security (SSL/TLS) certificates for use with AWS services and your internal connected resources. SSL/TLS certificates are used to secure network communications and establish the identity of websites over the Internet as well as resources on private networks. AWS Certificate Manager removes the time-consuming manual process of purchasing, uploading, and renewing SSL/TLS certificates.

With AWS Certificate Manager, you can quickly request a certificate, deploy it on ACM-integrated AWS resources, such as Elastic Load Balancers, Amazon CloudFront distributions, and APIs on API Gateway, and let AWS Certificate Manager handle certificate renewals. It also enables you to create private certificates for your internal resources and manage the certificate lifecycle centrally. Public and private certificates provisioned through AWS Certificate Manager for use with ACM-integrated services are free. You pay only for the AWS resources you create to run your application. With AWS Certificate Manager Private Certificate Authority, you pay monthly for the operation of the private CA and for the private certificates you issue.

You can use the same SSL certificate from ACM in more than one AWS Region but it depends on whether you’re using Elastic Load Balancing or Amazon CloudFront. To use a certificate with Elastic Load Balancing for the same site (the same fully qualified domain name, or FQDN, or set of FQDNs) in a different Region, you must request a new certificate for each Region in which you plan to use it. To use an ACM certificate with Amazon CloudFront, you must request the certificate in the US East (N. Virginia) region. ACM certificates in this region that are associated with a CloudFront distribution are distributed to all the geographic locations configured for that distribution.

Hence, the correct answer is the option that says: **In each new AWS Region, request for SSL/TLS certificates using the AWS Certificate Manager for each FQDN. Associate the new certificates to the corresponding Application Load Balancer of the same AWS Region.**

The option that says: **In each new AWS Region, request for SSL/TLS certificates using AWS KMS for each FQDN. Associate the new certificates to the corresponding Application Load Balancer of the same AWS Region** is incorrect because AWS KMS is not the right service to use to generate the SSL/TLS certificates. You have to utilize ACM instead.

The option that says: **Use the AWS KMS in the US West (N. California) region to request for SSL/TLS certificates for each FQDN which will be used to all regions. Associate the new certificates to the new Application Load Balancer on each new AWS Region that the Architect will add**is incorrect because you have to use the AWS Certificate Manager (ACM) service to generate the certificates and not AWS KMS as this service is primarily used for data encryption. Moreover, you have to associate the certificates that were generated from the same AWS Region where the load balancer is launched.

The option that says: **Use the AWS Certificate Manager service in the US West (N. California) region to request for SSL/TLS certificates for each FQDN which will be used to all regions. Associate the new certificates to the new Application Load Balancer on each new AWS Region that the Architect will add**is incorrect because you can only use the same SSL certificate from ACM in more than one AWS Region if you are attaching it to your CloudFront distribution only, and not to your Application Load Balancer. To use a certificate with Elastic Load Balancing for the same site (the same fully qualified domain name, or FQDN, or set of FQDNs) in a different Region, you must request a new certificate for each Region in which you plan to use it.

**References:**

<https://aws.amazon.com/certificate-manager/faqs/>

<https://aws.amazon.com/certificate-manager/>

Bottom of Form

Top of Form

Question 18: Skipped

Company XYZ has hired you as a Solutions Architect for their Flight Deals web application which is currently hosted on their on-premises data center. The website hosts high-resolution photos of top tourist destinations in the world and uses a third-party payment platform to accept payments. Recently, they have heavily invested on their global marketing campaign and there is a high probability that the incoming traffic to their Flight Deals website will increase in the coming days. Due to a tight deadline, the company does not have the time to fully migrate the website to AWS. A set of security rules that block common attack patterns, such as SQL injection and cross-site scripting should also be implemented to improve the website security.

Which of the following options will maintain the website's functionality despite the massive amount of incoming traffic?

* ​

Use the AWS Server Migration Service to easily migrate the website from your on-premises data center to your VPC. Create an Auto Scaling group to automatically scale the web tier based on the incoming traffic. Deploy AWS WAF on the Amazon CloudFront distribution to protect the website from common web attacks.

* ​

Create and configure an S3 bucket as a static website hosting. Move the web domain of the website from your on-premises data center to Route 53 then route the newly created S3 bucket as the origin. Enable Amazon S3 server-side encryption with AWS Key Management Service managed keys.

* ​

Use CloudFront to cache and distribute the high resolution images and other static assets of the website. Deploy AWS WAF on the Amazon CloudFront distribution to protect the website from common web attacks.

**(Correct)**

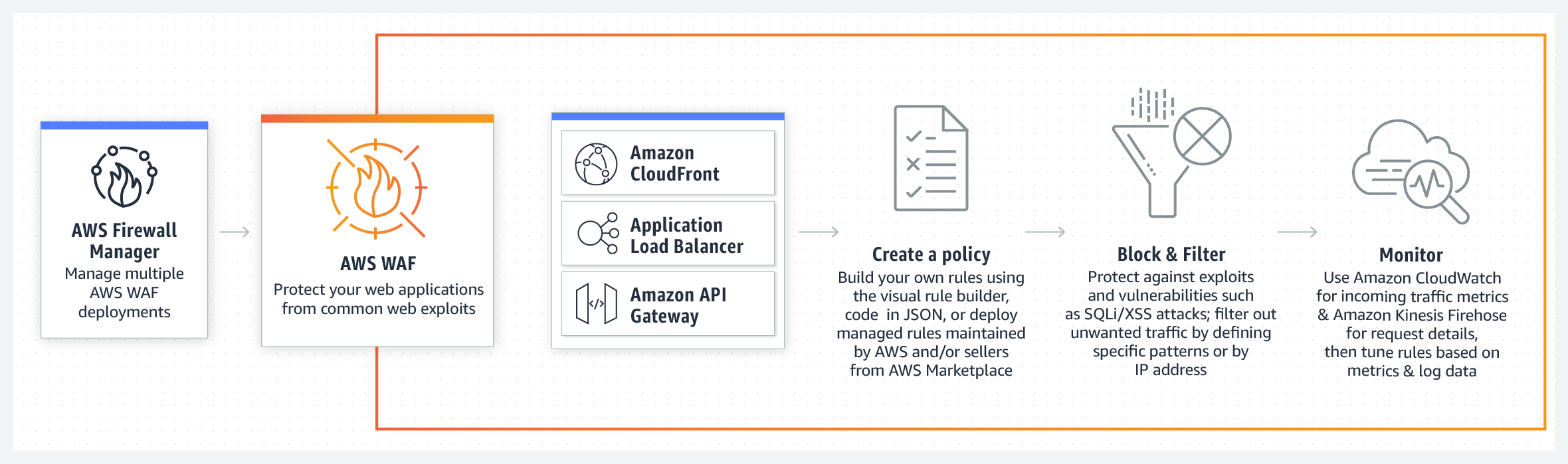
* ​

Generate an AMI based on the existing Flight Deals website. Launch the AMI to a fleet of EC2 instances with Auto Scaling group enabled, for it to automatically scale up or scale down based on the incoming traffic. Place these EC2 instances behind an ALB which can balance traffic between the web servers in the on-premises data center and the web servers hosted in AWS.

**Explanation**

**Amazon CloudFront** is a web service that speeds up distribution of your static and dynamic web content, such as .html, .css, .js, and image files, to your users. CloudFront delivers your content through a worldwide network of data centers called edge locations. When a user requests content that you're serving with CloudFront, the user is routed to the edge location that provides the lowest latency (time delay), so that content is delivered with the best possible performance.

**AWS WAF** is a web application firewall that helps protect your web applications or APIs against common web exploits that may affect availability, compromise security, or consume excessive resources. AWS WAF gives you control over how traffic reaches your applications by enabling you to create security rules that block common attack patterns, such as SQL injection or cross-site scripting, and rules that filter out specific traffic patterns you define. You can get started quickly using Managed Rules for AWS WAF, a pre-configured set of rules managed by AWS or AWS Marketplace Sellers. The Managed Rules for WAF address issues like the OWASP Top 10 security risks. These rules are regularly updated as new issues emerge. AWS WAF includes a full-featured API that you can use to automate the creation, deployment, and maintenance of security rules.



AWS WAF is easy to deploy and protect applications deployed on either Amazon CloudFront as part of your CDN solution, the Application Load Balancer that fronts all your origin servers, or Amazon API Gateway for your APIs. There is no additional software to deploy, DNS configuration, SSL/TLS certificate to manage, or need for a reverse proxy setup. With AWS Firewall Manager integration, you can centrally define and manage your rules, and reuse them across all the web applications that you need to protect.

Hence, the option that says: **Use CloudFront to cache and distribute the high resolution images and other static assets of the website. Deploy AWS WAF on the Amazon CloudFront distribution to protect the website from common web attacks** is correct because CloudFront will provide the scalability the website needs without doing major infrastructure changes. Take note that the website has a lot of high-resolution images which can easily be cached using CloudFront to alleviate the massive incoming traffic going to the on-premises web server and also provide a faster page load time for the web visitors.

The option that says: **Use the AWS Server Migration Service to easily migrate the website from your on-premises data center to your VPC. Create an Auto Scaling group to automatically scale the web tier based on the incoming traffic. Deploy AWS WAF on the Amazon CloudFront distribution to protect the website from common web attacks** is incorrect as migrating to AWS would be time-consuming compared with simply using CloudFront. Although this option can provide a more scalable solution, the scenario says that the company does not have ample time to do the migration.

The option that says:**Create and configure an S3 bucket as a static website hosting. Move the web domain of the website from your on-premises data center to Route53 then route the newly created S3 bucket as the origin. Enable Amazon S3 server-side encryption with AWS Key Management Service managed keys** is incorrect because the website is a dynamic website that accepts payments and bookings. Migrating your web domain to Route 53 may also take some time.

The option that says: **Generate an AMI based on the existing Flight Deals website. Launch the AMI to a fleet of EC2 instances with Auto Scaling group enabled, for it to automatically scale up or scale down based on the incoming traffic. Place these EC2 instances behind an ALB which can balance traffic between the web servers in the on-premises data center and the web servers hosted in AWS** is incorrect because it didn't mention any existing AWS Direct Connect or VPN connection. Although an Application Load Balancer can load balance the traffic between the EC2 instances in AWS Cloud and web servers located in the on-premises data center, your systems should be connected via Direct Connect or VPN connection first. In addition, the application seems to be used around the world because the company launched a global marketing campaign. Hence, CloudFront is a more suitable option for this scenario.

**References:**

<https://aws.amazon.com/cloudfront/>

<https://aws.amazon.com/waf/>

**Check out this Amazon CloudFront Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudfront/>

Bottom of Form

Top of Form

Question 19: Skipped

An Internet-Of-Things (IoT) startup is developing a health monitoring pet collar for dogs and cats. The startup has hired an electrical engineer to build a smart pet collar that collects biometric information of the pet every second and then sends it to a web portal through a POST API request. A Solutions Architect has been tasked to set up the API services and the web portal that will accept and process the biometric data as well as provide complete trends and health reports to pet owners around the globe. The portal should be highly durable, available, and scalable with an additional feature for showing real-time biometric data analytics and monitoring.

Which of the following is the best architecture that the Solutions Architect should implement to meet the above requirement?

* ​

1. Create an Amazon SQS queue to collect the incoming biometric data.

2. Analyze the data from SQS with Amazon Kinesis.

3. Store the results to an Amazon RDS for MySQL database.

* ​

1. Create an Amazon S3 bucket to collect the incoming biometric data from the smart pet collar.

2. Use Amazon Data Pipeline to run a data analysis task in the S3 bucket every day.

3. Use Amazon Redshift as the online analytic processing (OLAP) database for the web portal.

* ​

1. Use Amazon Kinesis Data Streams to collect the incoming biometric data.

2. Analyze the data using Amazon Kinesis and show the results in a real-time dashboard.

3. Set up a simple data aggregation process and pass the results to Amazon S3.

4. Store the data to Amazon Redshift, configured with automated backups, to handle complex analytics.

**(Correct)**

* ​

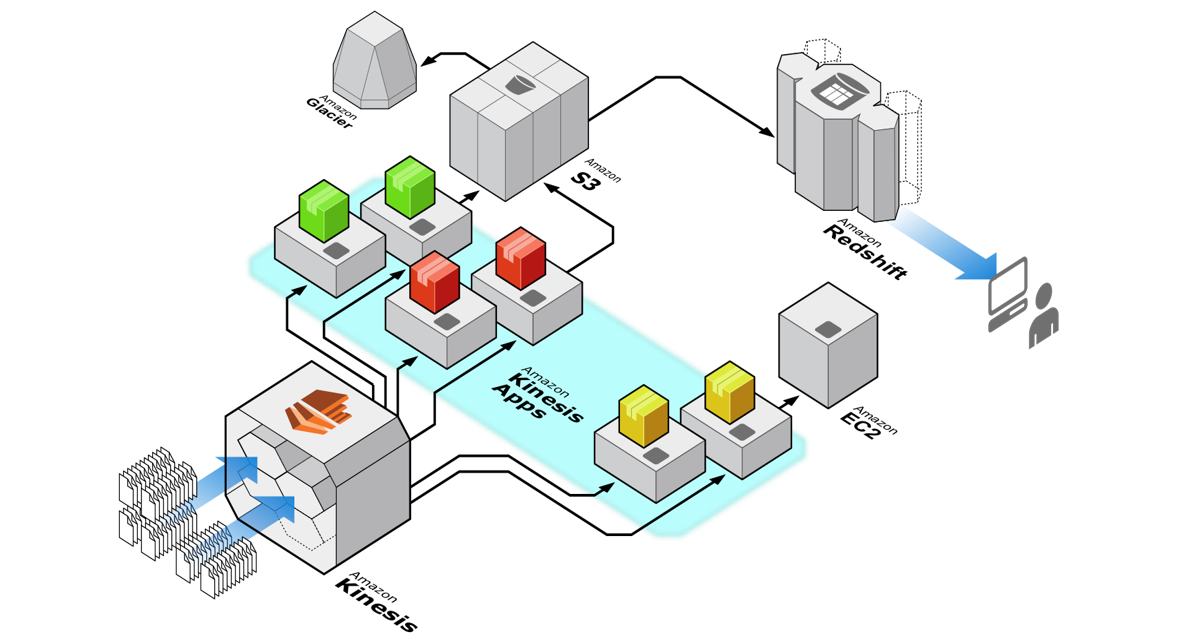
1. Launch an Amazon Elastic MapReduce instance to collect the incoming biometrics data.

2. Use Amazon Kinesis to analyze the data.

3. Save the results to an Amazon DynamoDB table.

**Explanation**

**Amazon Kinesis Data Streams** enable you to build custom applications that process or analyze streaming data for specialized needs. Kinesis Data Streams can continuously capture and store terabytes of data per hour from hundreds of thousands of sources such as website clickstreams, financial transactions, social media feeds, IT logs, and location-tracking events. With the Kinesis Client Library (KCL), you can build Kinesis Applications and use streaming data to power real-time dashboards, generate alerts, implement dynamic pricing and advertising, and more. You can also emit data from Kinesis Data Streams to other AWS services such as Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon EMR, and AWS Lambda.



**Kinesis Data Streams** can be used to collect log and event data from sources such as servers, desktops, and mobile devices. You can then build Kinesis Applications to continuously process the data, generate metrics, power live dashboards, and emit aggregated data into stores such as Amazon S3.

You can have your Kinesis Applications run real-time analytics on high frequency event data such as sensor data collected by Kinesis Data Streams, which enables you to gain insights from your data at a frequency of minutes instead of hours or days.

Hence, the following option is the correct answer as Amazon Kinesis is the one used here to collect the streaming data:

**1. Use Amazon Kinesis Data Streams to collect the incoming biometric data.**

**2. Analyze the data using Amazon Kinesis and show the results in a real-time dashboard.**

**3. Set up a simple data aggregation process and pass the results to Amazon S3.**

**4. Store the data to Amazon Redshift, configured with automated backups, to handle complex analytics.**

The following option is incorrect because S3, Data Pipeline, and Redshift do not provide real-time data analytics:

**1. Create an Amazon S3 bucket to collect the incoming biometric data from the smart pet collar.**

**2. Use Amazon Data Pipeline to run a data analysis task in the S3 bucket every day.**

**3. Use Amazon Redshift as the online analytic processing (OLAP) database for the web portal.**

The following option is incorrect because an SQS queue is not appropriate to use to accept all of the incoming biometric data. You should use Amazon Kinesis instead:

**1. Create an Amazon SQS queue to collect the incoming biometric data.**

**2. Analyze the data from SQS with Amazon Kinesis.**

**3. Store the results to an Amazon RDS for MySQL database*.***

The following option is incorrect because just like in the above, it does not use Amazon Kinesis to accept the incoming data:

**1. Launch an Amazon Elastic MapReduce instance to collect the incoming biometrics data.**

**2. Use Amazon Kinesis to analyze the data.**

**3. Save the results to an Amazon DynamoDB table.**

**References:**

<https://aws.amazon.com/kinesis/data-streams/details/>

<https://docs.aws.amazon.com/streams/latest/dev/introduction.html>

**Check out this Amazon Kinesis Cheat Sheet:**

[https://tutorialsdojo.com/amazon-kinesis/](https://tutorialsdojo.com/amazon-kinesis/?src=udemy)

Bottom of Form

Top of Form

Question 20: Skipped

A company has several virtual machines on its on-premises data center hosting its three-tier web application. The company wants to migrate the application to AWS to take advantage of the benefits of cloud computing. The following are the company requirements for the migration process:

- The virtual machine images from the on-premises data center must be imported to AWS.

- The changes on the on-premises servers must be synchronized to the AWS servers until the production cutover is completed.

- Have minimal downtime during the production cutover.

- The root volumes and data volumes (containing Terabytes of data) of the VMs must be migrated to AWS.

- The migration solution must have minimal operational overhead.

Which of the following options is the recommended solution to meet the company requirements?

* ​

Leverage both AWS Application Discovery Service and AWS Migration Hub to group the on-premises VMs as an application. Write an AWS CLI script that uses VM Import/Export to import the VMs as AMIs. Schedule the script to run at regular intervals to synchronize the changes from the on-premises environment to AWS. Launch Amazon EC2 instances based on the images created from VM Import/Export. After successful testing, perform a final virtual machine import before the cutover. Launch new instances based on the updated AMIs.

* ​

Create a job on AWS Server Migration Service (SMS) to migrate the virtual machines to AWS. Create a replication job for each application tier to sync the changes from the on-premises environment to AWS. Launch Amazon EC2 instances based on the images created from AWS SMS. After successful testing, perform a final replication before the cutover and launch new instances based on the updated AMIs.

**(Correct)**

* ​

Create a job on AWS Server Migration Service (SMS) to migrate the root volumes of the virtual machines to AWS. Import the data volumes using the AWS CLI import-snapshot command. Launch Amazon EC2 instances based on the images created from AWS SMS and attach the imported data volumes. After successful testing, perform a final replication before the cutover. Launch new instances based on the updated AMIs and attach the corresponding data volumes.

* ​

Write an AWS CLI script that uses VM Import/Export to migrate the virtual machines. Schedule the script to run at regular intervals to synchronize the changes from the on-premises environment to AWS. Launch Amazon EC2 instances based on the images created from VM Import/Export. After successful testing, re-run the script to perform a final replication before the cutover. Launch new instances based on the updated AMIs.

**Explanation**

**AWS Server Migration Service** automates the migration of your on-premises VMware vSphere, Microsoft Hyper-V/SCVMM, and Azure virtual machines to the AWS Cloud. AWS SMS incrementally replicates your server VMs as cloud-hosted Amazon Machine Images (AMIs) ready for deployment on Amazon EC2. Working with AMIs, you can easily test and update your cloud-based images before deploying them in production.

By using AWS SMS to manage your server migrations, you can:

**- Simplify the cloud migration process.** You can begin migrating a group of servers with just a few clicks in the AWS Management Console. After the migration has initiated, AWS SMS manages all the complexities of the migration process, including automatically replicating volumes of live servers to AWS and creating new AMIs periodically. You can quickly launch EC2 instances from AMIs in the console.

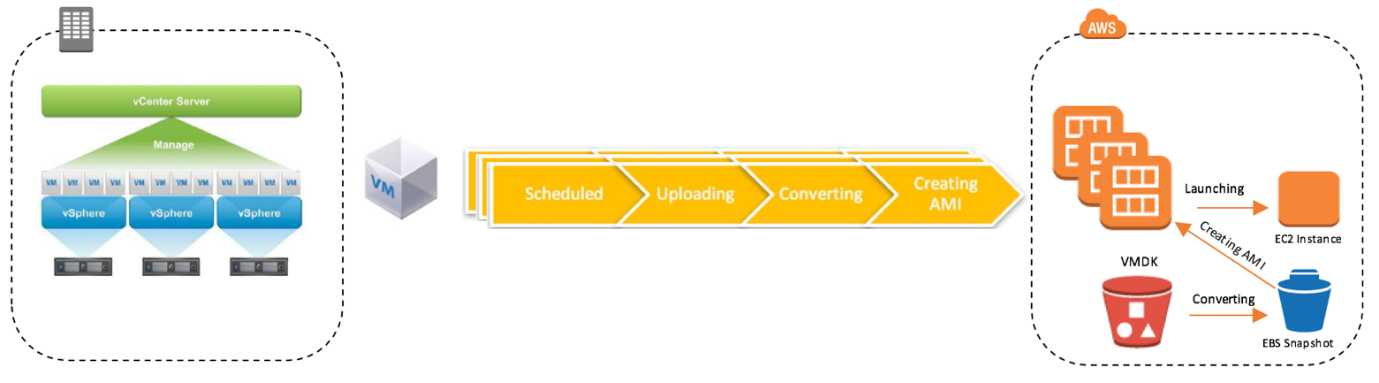
**- Orchestrate multi-server migrations.** AWS SMS orchestrates server migrations by allowing you to schedule replications and track the progress of a group of servers that constitutes an application. You can schedule initial replications, configure replication intervals, and track progress for each server using the console. When you launch a migrated application, you can apply customized configuration scripts that run during startup.

**- Test server migrations incrementally.** With support for incremental replication, AWS SMS allows fast, scalable testing of migrated servers. Because AWS SMS replicates incremental changes to your on-premises servers and transfers only the delta to the cloud, you can test small changes iteratively and save on network bandwidth.

**- Support the most widely used operating systems.** AWS SMS supports the replication of operating system images containing Windows, as well as several major Linux distributions.

**- Minimize downtime.** Incremental AWS SMS replication minimizes the business impact associated with application downtime during the final cutover.

AWS Server Migration Service is designed to simplify the end-to-end server migration process. AWS SMS currently supports the migration of on-premises virtual machines (VMs) as an agentless service using a virtual appliance. AWS SMS is an ideal solution to use when you are planning a scaled migration from VMware environments to AWS where the downtime, agentless tools, incremental replication, and testing the application before the cutover are critical considerations.



Therefore, the correct answer is: **Create a job on AWS Server Migration Service (SMS) to migrate the virtual machines to AWS. Create a replication job for each application tier to sync the changes from the on-premises environment to AWS. Launch Amazon EC2 instances based on the images created from AWS SMS. After successful testing, perform a final replication before the cutover and launch new instances based on the updated AMIs.**

The option that says: **Write an AWS CLI script that uses VM Import/Export to migrate the virtual machines. Schedule the script to run at regular intervals to synchronize the changes from the on-premises environment to AWS. Launch Amazon EC2 instances based on the images created from VM Import/Export. After successful testing, re-run the script to perform a final replication before the cutover. Launch new instances based on the updated AMIs** is incorrect. AWS VM Import/Export does not support synching incremental changes from the on-premises environment to AWS. You will need to import the VM again as a whole after you make changes to the on-premises environment. This requires a lot of time and adds more operational overhead.

The option that says: **Create a job on AWS Server Migration Service (SMS) to migrate the root volumes of the virtual machines to AWS. Import the data volumes using the AWS CLI import-snapshot command. Launch Amazon EC2 instances based on the images created from AWS SMS and attach the imported data volumes. After successful testing, perform a final replication before the cutover. Launch new instances based on the updated AMIs and attach the corresponding data volumes** is incorrect. This may be possible but creating manual snapshots of the data volumes requires more operational overhead. AWS SMS supports up to 16TB volumes so you can use it to migrate the data volumes as well.

The option that says: **Leverage both AWS Application Discovery Service and AWS Migration Hub to group the on-premises VMs as an application. Write an AWS CLI script that uses VM Import/Export to import the VMs as AMIs. Schedule the script to run at regular intervals to synchronize the changes from the on-premises environment to AWS. Launch Amazon EC2 instances based on the images created from VM Import/Export. After successful testing, perform a final virtual machine import before the cutover. Launch new instances based on the updated AMIs** is incorrect. The AWS Application Discovery Service plans migration projects by gathering information about the on-premises data center and all discovered data are stored in your AWS Migration Hub. This is similar to the other option for VM Import/Export as you will need to import the VM again as a whole after you make changes on the on-premises environment.

**References:**

<https://docs.aws.amazon.com/server-migration-service/latest/userguide/server-migration.html>

<https://aws.amazon.com/blogs/apn/aws-server-migration-service-server-migration-to-the-cloud-made-easy/>

<https://aws.amazon.com/about-aws/whats-new/2018/09/aws-server-migration-service-adds-support-for-migrating-larger-data-volumes/>

<https://docs.aws.amazon.com/server-migration-service/latest/userguide/console_workflow.html>

<https://docs.aws.amazon.com/server-migration-service/latest/userguide/application-migration.html>

**Check out this AWS Server Migration Service Cheat Sheet:**

[https://tutorialsdojo.com/aws-server-migration-service-sms/](https://tutorialsdojo.com/aws-server-migration-service-sms/?src=udemy)

Bottom of Form

Top of Form

Question 21: Skipped

The department of education just recently decided to leverage on AWS cloud infrastructure to supplement their current on-premises network. They are building a new learning portal that teaches kids basic computer science concepts and provides innovative gamified courses for teenagers where they can gain higher rankings, power-ups and badges. A Solutions Architect is instructed to build a highly available cloud infrastructure in AWS with multiple Availability Zones. The department wants to increase the application’s reliability and gain actionable insights using application logs. A Solutions Architect needs to aggregate logs, automate log analysis for errors and immediately notify the IT Operations team when errors breached a certain threshold.

Which of the following is the MOST suitable solution that the Architect should implement?

* ​

Download and install the Amazon CloudWatch agent in the on-premises servers and send the logs to Amazon CloudWatch Events. Create a metric filter in CloudWatch to turn log data into numerical metrics to identify and measure application errors. Use Amazon Athena to monitor the metric filter and immediately notify the IT Operations team for any issues.

* ​

Download and install the AWS X-Ray agent in the on-premises servers and send the logs to AWS Lambda to turn log data into numerical metrics that identify and measure application errors. Store the metrics data in Systems Manager Parameter Store. Create a CloudWatch Alarm that monitors the metric and immediately notify the IT Operations team for any issues.

* ​

Download and install the Amazon Kinesis agent in the on-premises servers and send the logs to Amazon CloudWatch Logs. Create a metric filter in CloudWatch to turn log data into numerical metrics to identify and measure application errors. Use Amazon QuickSight to monitor the metric filter in CloudWatch and immediately notify the IT Operations team for any issues.

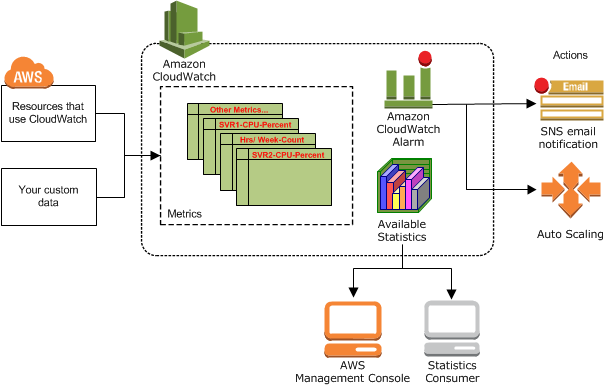
* ​

Download and install the Amazon CloudWatch agent in the on-premises servers and send the logs to Amazon CloudWatch Logs. Create a metric filter in CloudWatch to turn log data into numerical metrics to identify and measure application errors. Create a CloudWatch Alarm that monitors the metric filter and immediately notify the IT Operations team for any issues.

**(Correct)**

**Explanation**

Amazon CloudWatch monitors your Amazon Web Services (AWS) resources and the applications you run on AWS in real-time. You can use CloudWatch to collect and track metrics, which are variables you can measure for your resources and applications. The CloudWatch home page automatically displays metrics about every AWS service you use. You can additionally create custom dashboards to display metrics about your custom applications, and display custom collections of metrics that you choose.



After the CloudWatch Logs agent begins publishing log data to Amazon CloudWatch, you can begin searching and filtering the log data by creating one or more metric filters. Metric filters define the terms and patterns to look for in log data as it is sent to CloudWatch Logs. CloudWatch Logs uses these metric filters to turn log data into numerical CloudWatch metrics that you can graph or set an alarm on. You can use any type of CloudWatch statistic, including percentile statistics, when viewing these metrics or setting alarms.

You can create alarms which watch metrics and send notifications or automatically make changes to the resources you are monitoring when a threshold is breached. For example, you can monitor the CPU usage and disk reads and writes of your Amazon EC2 instances and then use this data to determine whether you should launch additional instances to handle increased load. You can also use this data to stop under-used instances to save money. With CloudWatch, you gain system-wide visibility into resource utilization, application performance, and operational health.

Hence, the correct answer is: **Download and install the Amazon CloudWatch agent in the on-premises servers and send the logs to Amazon CloudWatch Logs. Create a metric filter in CloudWatch to turn log data into numerical metrics to identify and measure application errors. Create a CloudWatch Alarm that monitors the metric filter and immediately notify the IT Operations team for any issues.**

The option that says:**Download and install the Amazon Kinesis agent in the on-premises servers and send the logs to Amazon CloudWatch Logs. Create a metric filter in CloudWatch to turn log data into numerical metrics to identify and measure application errors. Use Amazon QuickSight to monitor the metric filter in CloudWatch and immediately notify the IT Operations team for any issues** is incorrect because you have to use Amazon CloudWatch agent instead of an Amazon Kinesis agent to send the logs to Amazon CloudWatch Logs. It is also better to use CloudWatch Alarms to monitor the metric filter than to use Amazon QuickSight.

The option that says:**Download and install the AWS X-Ray agent in the on-premises servers and send the logs to AWS Lambda to turn log data into numerical metrics that identify and measure application errors. Store the metrics data in Systems Manager Parameter Store. Create a CloudWatch Alarm that monitors the metric and immediately notify the IT Operations team for any issues**is incorrect because you have to use Amazon CloudWatch agent instead of an AWS X-Ray agent to send the logs to Amazon CloudWatch Logs. AWS X-Ray is primarily used for debugging applications and not for monitoring. It is not suitable to use a Lambda function to track and process the metrics and use the Systems Manager Parameter Store to keep the metrics data. These operations can simply be done by CloudWatch.

*The option that says:***Download and install the Amazon CloudWatch agent in the on-premises servers and send the logs to Amazon CloudWatch Events. Create a metric filter in CloudWatch to turn log data into numerical metrics to identify and measure application errors. Use Amazon Athena to monitor the metric filter and immediately notify the IT Operations team for any issues** is incorrect because you have to send the logs to CloudWatch Logs and not CloudWatch Events. It is also better to use CloudWatch Alarm to monitor the metric filter and immediately notify the IT Operations team for any issues.

**References:**

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/install-CloudWatch-Agent-on-premise.html>

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

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/MonitoringPolicyExamples.html>

**Check out this Amazon CloudWatch Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudwatch/>

Bottom of Form

Top of Form

Question 22: Skipped

You are working as the Lead Systems Architect for a government bank in which you are handling a web application that retrieves and displays highly-sensitive information about the clients. The amount of traffic the site will receive is known and not expected to fluctuate. SSL will be used as part of the application's data security. Your chief information security officer (CISO) is concerned about the security of your SSL private key and she wants to ensure that the key cannot be accidentally or intentionally moved outside the corporate environment. You are also concerned that the application logs might contain some sensitive information, although you have already configured an encrypted EBS volume to store the data. In this scenario, the application logs must be stored securely and durably so that they can only be decrypted by the authorized government employees.

Which of the following is the most suitable and highly available architecture that can meet all of the requirements?

* ​

Distribute traffic to a set of web servers using an Elastic Load Balancer that performs TCP load balancing. Use an AWS CloudHSM to perform the SSL transactions and deliver your application logs to a private Amazon S3 bucket using server-side encryption.

* ​

Distribute traffic to a set of web servers using an Elastic Load Balancer that performs TCP load balancing. Use CloudHSM deployed to two Availability Zones to perform the SSL transactions and deliver your application logs to a private Amazon S3 bucket using server-side encryption.

**(Correct)**

* ​

Distribute traffic to a set of web servers using an Elastic Load Balancer. Use TCP load balancing for the load balancer and configure your web servers to retrieve the SSL private key from a private Amazon S3 bucket on boot. Use another private Amazon S3 bucket to store your web server logs using Amazon S3 server-side encryption.

* ​

Distribute traffic to a set of web servers using an Elastic Load Balancer. To secure the SSL private key, upload the key to the load balancer and configure the load balancer to offload the SSL traffic. Lastly, write your application logs to an instance store volume that has been encrypted using a randomly generated AES key.

**Explanation**

AWS CloudHSM is a cloud-based hardware security module (HSM) that enables you to easily generate and use your own encryption keys on the AWS Cloud. With CloudHSM, you can manage your own encryption keys and automate time-consuming administrative tasks for you, such as hardware provisioning, software patching, high-availability, and backups.

The option that says: ***Distribute traffic to a set of web servers using an Elastic Load Balancer that performs TCP load balancing. Use CloudHSM deployed to two Availability Zones to perform the SSL transactions and deliver your application logs to a private Amazon S3 bucket using server-side encryption*** is correct because it uses CloudHSM for performing the SSL transaction without requiring any additional way of storing or managing the SSL private key. This is the most secure way of ensuring that the key will not be moved outside of the AWS environment. Also, it uses the highly available and durable S3 service for storing the logs. Take note that this option says "*server-side encryption*" and not "*Amazon S3-Managed Encryption Keys*", which are two different things.

The option that says: ***Distribute traffic to a set of web servers using an Elastic Load Balancer. Use TCP load balancing for the load balancer and configure your web servers to retrieve the SSL private key from a private Amazon S3 bucket on boot. Use another private Amazon S3 bucket to store your web server logs using Amazon S3 server-side encryption*** is incorrect because it does not use a secure way of managing the SSL private key for the SSL transaction.

The option that says: ***Distribute traffic to a set of web servers using an Elastic Load Balancer. To secure the SSL private key, upload the key to the load balancer and configure the load balancer to offload the SSL traffic. Lastly, write your application logs to an instance store volume that has been encrypted using a randomly generated AES key*** is incorrect because the application logs are written to an ephemeral volume, which means that the data will be lost when the EC2 instance is terminated. Hence, this solution is neither durable nor secure.

The option that says: ***Distribute traffic to a set of web servers using an Elastic Load Balancer that performs TCP load balancing. Use an AWS CloudHSM to perform the SSL transactions and deliver your application logs to a private Amazon S3 bucket using server-side encryption*** is incorrect because although it is almost similar to the correct option, the architecture did not explicitly say that the CloudHSM is deployed to multiple Availability Zones, which means that this architecture is not highly available compared with the correct option.

I deliberately designed the question to have this sort of ambiguity to make it more challenging as I know how hard the actual AWS CSA Professional exam is. So in this case, the correct option does not say if it is using SSE-S3 or SSE-C. This is one of the trick part of this question. If it is using SSE-S3, then Option 2 would definitely be wrong because AWS will be managing the AES-256 key.

However, if it uses SSE-C, then the correct option would be correct because the AES-256 key would be managed by the authorized government employees only. Since this is the correct answer, it is implied that the correct option is using SSE-C, instead of SSE-S3.

Remember that in SSE-C, when you upload an object, Amazon S3 uses the encryption key you provide to apply AES-256 encryption to your data and removes the encryption key from memory. Amazon S3 does not store the encryption key you provide. Instead, they store a randomly salted HMAC value of the encryption key in order to validate future requests. The salted HMAC value cannot be used to derive the value of the encryption key or to decrypt the contents of the encrypted object. That means, if you lose the encryption key, you lose the object.

**References:**

<https://aws.amazon.com/cloudhsm/>

<https://docs.aws.amazon.com/cloudhsm/latest/userguide/ssl-offload.html>

<https://docs.aws.amazon.com/AmazonS3/latest/dev/ServerSideEncryptionCustomerKeys.html>

<https://docs.aws.amazon.com/AmazonS3/latest/dev/UsingServerSideEncryption.html>

**Check out this Amazon S3 Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-s3/>

**Learn more about AWS CloudHSM and AWS WAF in this video tutorial:**

<https://youtu.be/IMxImFoVpmI>

Bottom of Form

Top of Form

Question 23: Skipped

A tech startup is planning to launch a new global mobile marketplace using AWS Amplify and AWS Mobile Hub. To lower the latency, the backend APIs will be launched to multiple AWS regions to process the sales and financial transactions on the region closest to the users. You are instructed to design the system architecture to ensure that the transactions made in one region are automatically replicated to other regions. In the coming months ahead, it is expected that the marketplace will have millions of users across North America, South America, Europe, and Asia.

Which of the following is the most scalable, cost-effective and highly available architecture that you should implement?

* ​

In each local region, store the individual transactions to a DynamoDB table. Set up an AWS Lambda function to read recent writes from the table, and replay the data to DynamoDB tables in all other regions.

* ​

Create a Global DynamoDB table with replica tables across several AWS regions that you prefer. In each local region, store the individual transactions to a DynamoDB replica table in the same region. Any changes made in one of the replica tables will automatically be replicated across all other tables.

**(Correct)**

* ​

Create an Amazon Aurora Multi-Master database on all required regions. Store the individual transactions to the Amazon Aurora instance in the local region. Replicate the transactions table between regions using Aurora replication. In this set up, any changes made in one of the tables will be automatically replicated across all other tables.

* ​

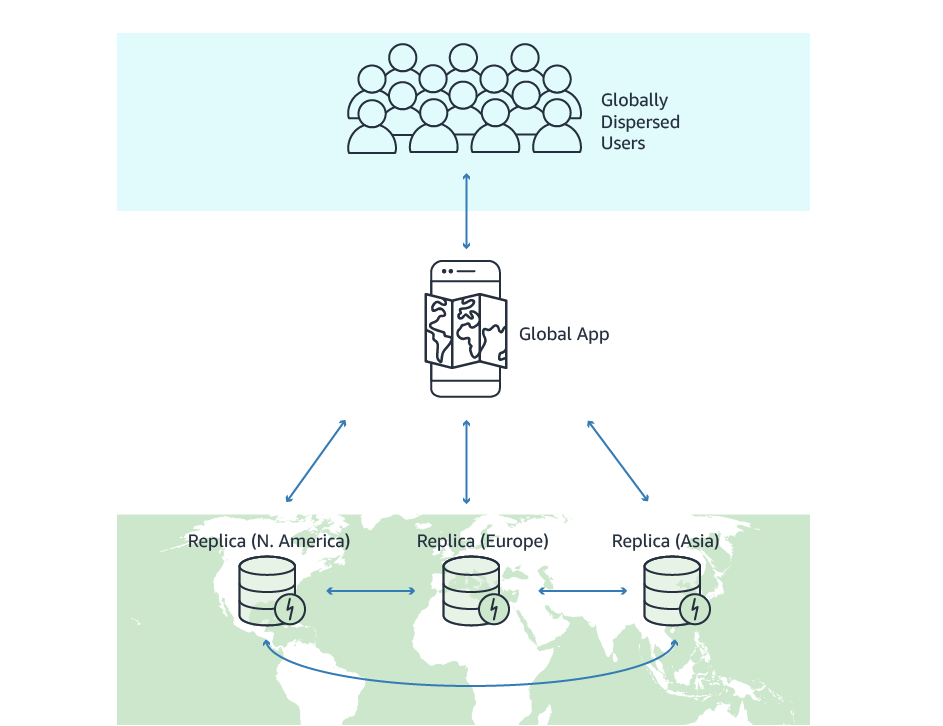
Use a combination of AWS Control Tower and Amazon Connect to launch and centrally manage multiple DynamoDB tables in various AWS Regions. In each local region, store the individual transactions to a DynamoDB replica table in the same region.

**Explanation**

Global Tables builds upon DynamoDB’s global footprint to provide you with a fully managed, multi-region, and multi-master database that provides fast, local, read and write performance for massively scaled, global applications. Global Tables replicates your Amazon DynamoDB tables automatically across your choice of AWS regions.

Global Tables eliminates the difficult work of replicating data between regions and resolving update conflicts, enabling you to focus on your application’s business logic. In addition, Global Tables enables your applications to stay highly available even in the unlikely event of isolation or degradation of an entire region.

Hence, the option that says:**Create a Global DynamoDB table with replica tables across several AWS regions that you prefer. In each local region, store the individual transactions to a DynamoDB replica table in the same region. Any changes made in one of the replica tables will automatically be replicated across all other tables** is the correct answer.



The option that says: **In each local region, store the individual transactions to a DynamoDB table. Set up an AWS Lambda function to read recent writes from the table, and replay the data to DynamoDB tables in all other regions** is incorrect because using an AWS Lambda function to replicate all data across regions is not a scalable solution. Remember that there will be millions of users who will use the mobile app around the world, and this entails a lot of replication and compute capacity for a single Lambda function. In this scenario, the best solution is to use Global DynamoDB tables with DynamoDB Stream option enabled to automatically handle the replication process.

The option that says:**Use a combination of AWS Control Tower and Amazon Connect to launch and centrally manage multiple DynamoDB tables in various AWS Regions. In each local region, store the individual transactions to a DynamoDB replica table in the same region** is incorrect because Amazon Connect is just an easy to use omnichannel cloud contact center that helps companies provide superior customer service at a lower cost, while AWS Control Tower just offers the easiest way to set up and govern a new, secure, multi-account AWS environment. You can't use these two services to set up a Global DynamoDB Table.

The option that says: **Create an Amazon Aurora Multi-Master database on all required regions. Store the individual transactions to the Amazon Aurora instance in the local region. Replicate the transactions table between regions using Aurora replication. In this setup, any changes made in one of the tables will be automatically replicated across all other tables** is incorrect because, by default, all DB instances in a multi-master cluster must be in the same AWS Region and you can't enable cross-region replicas from multi-master clusters. In addition, DynamoDB provides better global scalability for mobile applications compared to Amazon Aurora.

**References:**

<https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/globaltables_HowItWorks.html>

<https://aws.amazon.com/dynamodb/global-tables/>

**Check out this Amazon DynamoDB Cheat Sheet:**

[https://tutorialsdojo.com/amazon-dynamodb/](https://tutorialsdojo.com/amazon-dynamodb/?src=udemy)

Bottom of Form

Top of Form

Question 24: Skipped

You are the Lead Solutions Architect for an IT consulting firm which has various teams and departments that have been grouped into several organizational units (OUs) using AWS Organizations. You received a report from the security team that there was a suspected breach in your environment where a third-party AWS account was suddenly added to your organization without any prior approval. The external account has high level access privileges to the accounts that you own but luckily, no detrimental action was performed.

What should you do to properly set up a monitoring system than notifies you for any changes to your AWS accounts? (Select TWO.)

* ​

Monitor all changes to your organization using Systems Manager and use CloudWatch Events to notify you for any new activity to your account.

* ​

Set up a CloudWatch Dashboard to monitor any changes to your organizations and create an SNS topic that would send you a notification.

* ​

Create a trail in Amazon CloudTrail to capture all API calls to your AWS Organizations, including calls from the AWS Organizations console and from code calls to the AWS Organizations APIs. Use CloudWatch Events and SNS to raise events when administrator-specified actions occur in an organization and send a notification to you.

**(Correct)**

* ​

Provision an AWS-approved third-party monitoring tool from the AWS Marketplace that would send alerts if a breach was detected. Use AWS GuardDuty to analyze any possible breach and notify the administrators using AWS SNS.

* ​

Use AWS Config to monitor the compliance of your AWS Organizations. Set up an SNS Topic or CloudWatch Events that will send alerts to you for any changes.

**(Correct)**

**Explanation**

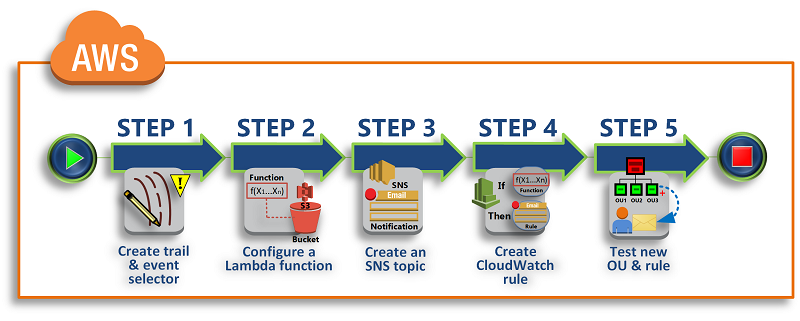
AWS Organizations can work with CloudWatch Events to raise events when administrator-specified actions occur in an organization. For example, because of the sensitivity of such actions, most administrators would want to be warned every time someone creates a new account in the organization or when an administrator of a member account attempts to leave the organization. You can configure CloudWatch Events rules that look for these actions and then send the generated events to administrator-defined targets. Targets can be an Amazon SNS topic that emails or text messages its subscribers. Combining this with Amazon CloudTrail, you can set an event to trigger whenever a matching API call is received.

Multi-account, multi-region data aggregation in AWS Config enables you to aggregate AWS Config data from multiple accounts and regions into a single account. Multi-account, multi-region data aggregation is useful for central IT administrators to monitor compliance for multiple AWS accounts in the enterprise. An aggregator is a new resource type in AWS Config that collects AWS Config data from multiple source accounts and regions

Hence, the following options are the correct answers in this scenario:

***1. Create a trail in Amazon CloudTrail to capture all API calls to your AWS Organizations, including calls from the AWS Organizations console and from code calls to the AWS Organizations APIs. Use CloudWatch Events and SNS to raise events when administrator-specified actions occur in an organization and send a notification to you.***

***2. Use AWS Config to monitor the compliance of your AWS Organizations. Set up an SNS Topic or CloudWatch Events that will send alerts to you for any changes***.



***Monitoring all changes to your organization using Systems Manager and using CloudWatch Events to notify you for any new activity to your account*** is incorrect because AWS Systems Manager is a collection of capabilities for configuring and managing your Amazon EC2 instances, on-premises servers and virtual machines, and other AWS resources at scale. This can't be used to monitor the changes to the set up of AWS Organizations.

***Setting up a CloudWatch Dashboard to monitor any changes to your organizations and creating an SNS topic that would send you a notification*** is incorrect because a CloudWatch Dashboard is primarily used to monitor your AWS resources and not the configuration of your AWS Organizations. Although you can enable sharing of all CloudWatch Events across all accounts in your organization, this can't be used to monitor if there is a new AWS account added to your AWS Organizations. Most of the time, the Amazon CloudWatch Events service is primarily used to monitor your AWS resources and the applications you run on AWS in real time.

***Provisioning an AWS-approved third-party monitoring tool from the AWS Marketplace that would send alerts if a breach was detected, then using AWS GuardDuty to analyze any possible breach and notifying the administrators using AWS SNS*** is incorrect because this option entails a lot of configuration, which is not fit for the scenario. GuardDuty might not determine similar future incidents as malicious if it was performed by an authenticated user already within the organization.

**References:**

<https://docs.aws.amazon.com/organizations/latest/userguide/orgs_monitoring.html>

<https://docs.aws.amazon.com/organizations/latest/userguide/orgs_tutorials_cwe.html>

Bottom of Form

Top of Form

Question 25: Skipped

An international foreign exchange company has a serverless forex trading application which was built using AWS SAM and is hosted on AWS Serverless Application Repository. They have millions of users worldwide who use their online portal 24/7 to trade currencies. However, they are receiving a lot of complaints that it takes a few minutes for their users to login to their portal lately, including occasional HTTP 504 errors. As the Solutions Architect, you are tasked to optimize the system and to significantly reduce the time to login to improve the customers' satisfaction.

Which of the following should you implement in order to improve the performance of the application with minimal cost? (Select TWO.)

* ​

Increase the cache hit ratio of your CloudFront distribution by configuring your origin to add a Cache-Control max-age directive to your objects, and specify the longest practical value for max-age.

* ​

Set up an origin failover by creating an origin group with two origins. Specify one as the primary origin and the other as the second origin which CloudFront automatically switches to when the primary origin returns specific HTTP status code failure responses.

**(Correct)**

* ​

Use Lambda@Edge to allow your Lambda functions to customize content that CloudFront delivers and to execute the authentication process in AWS locations closer to the users.

**(Correct)**

* ​

Deploy your application to multiple AWS regions to accommodate your users around the world. Set up a Route 53 record with latency routing policy to route incoming traffic to the region that provides the best latency to the user.

* ​

Set up multiple and geographically disperse VPCs to various AWS regions then create a transit VPC to connect all of your resources. Deploy the Lambda function in each region using AWS SAM, in order to handle the requests faster.

**Explanation**

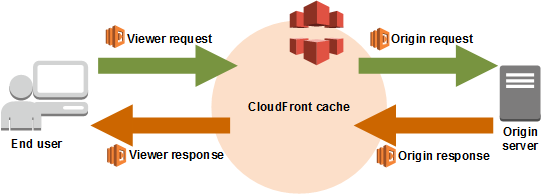
Lambda@Edge lets you run Lambda functions to customize the content that CloudFront delivers, executing the functions in AWS locations closer to the viewer. The functions run in response to CloudFront events, without provisioning or managing servers. You can use Lambda functions to change CloudFront requests and responses at the following points:

-After CloudFront receives a request from a viewer (viewer request)

-Before CloudFront forwards the request to the origin (origin request)

-After CloudFront receives the response from the origin (origin response)

-Before CloudFront forwards the response to the viewer (viewer response)



In the given scenario, you can***use Lambda@Edge to allow your Lambda functions to customize the content that CloudFront delivers and to execute the authentication process in AWS locations closer to the users.***In addition, you can***set up an origin failover by creating an origin group with two origins with one as the primary origin and the other as the second origin which CloudFront automatically switches to when the primary origin fails.*** This will alleviate the occasional HTTP 504 errors that users are experiencing.

The option that says: ***Deploy your application to multiple AWS regions to accommodate your users around the world. Set up a Route 53 record with latency routing policy to route incoming traffic to the region that provides the best latency to the user*** is incorrect because although this may resolve the performance issue, this solution entails a significant implementation cost since you have to deploy your application to multiple AWS regions. Remember that the scenario asks for a solution that will improve the performance of the application with **minimal cost**.

The option that says: ***Increase the cache hit ratio of your CloudFront distribution by configuring your origin to add a Cache-Control max-age directive to your objects, and specify the longest practical value for max-age*** is incorrect because improving the cache hit ratio for the CloudFront distribution is irrelevant in this scenario. You can improve your cache performance by increasing the proportion of your viewer requests that are served from CloudFront edge caches instead of going to your origin servers for content. However, take note that the problem in the scenario is the sluggish authentication process of your global users and not just the caching of the static objects.

**References:**

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/high_availability_origin_failover.html>

<https://docs.aws.amazon.com/lambda/latest/dg/lambda-edge.html>

**Check out these Amazon CloudFront and AWS Lambda Cheat Sheets:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudfront/>

<https://tutorialsdojo.com/aws-cheat-sheet-aws-lambda/>

Bottom of Form

Top of Form

Question 26: Skipped

Your startup is building a web app that lets users post photos of good deeds in their neighborhood with a 143-character caption/article. You decided to write the application in ReactJS, a popular javascript framework so that it would run on the broadest range of browsers, mobile phones, and tablets. Your app should provide access to Amazon DynamoDB to store the caption. The initial prototype shows that there aren't large spikes in usage.

Which option provides the most cost-effective and scalable architecture for this application?

* ​

Register the web application with a Web Identity Provider such as Google, Facebook, Amazon, or from any other popular social site. Create an IAM role for that web provider and set up permissions for the IAM role to allow GET and PUT operations in DynamoDB. Serve your web application from an NGINX server hosted on a fleet of EC2 instances, with a load balancer and auto-scaling. Add an IAM role to the EC2 instance to allow GET and PUT operations to DynamoDB tables.

* ​

Configure the ReactJS client with temporary credentials from the Security Token Service using a Token Vending Machine (TVM) to provide signed credentials to an IAM user. This will allow GET and PUT operations to DynamoDB. Serve your web application from an NGINX server hosted in a fleet of EC2 instances that are load-balanced and auto-scaled. Your EC2 instances are configured with an IAM role that allows GET and PUT operations in DynamoDB.

* ​

Register the web application with a Web Identity Provider such as Google, Facebook, Amazon, or from any other popular social sites and use the AssumeRoleWithWebIdentity API of STS to generate temporary credentials. Create an IAM role for that web provider and set up permissions for the IAM role to allow GET and PUT operations in Amazon S3 and DynamoDB. Serve your web app out of an S3 bucket enabled as a website.

**(Correct)**

* ​

Configure the ReactJS client with temporary credentials from the Security Token Service using a Token Vending Machine (TVM) on an EC2 instance. This will provide signed credentials to an IAM user allowing GET and PUT operations in the DynamoDB table and the S3 bucket. You serve your mobile application out of an S3 bucket enabled as a web site.

**Explanation**

If you don't use Amazon Cognito, then you choose to write a custom code or app that interacts with a web IdP (Login with Amazon, Facebook, Google, or any other OIDC-compatible IdP) and then call the AssumeRoleWithWebIdentity API to trade the authentication token you get from those IdPs for AWS temporary security credentials. If you have already used this approach for existing apps, you can continue to use it. You can also deploy your app in the S3 bucket.

The option that says: **Register the web application with a Web Identity Provider such as Google, Facebook, Amazon, or from any other popular social sites and use the AssumeRoleWithWebIdentity API of STS to generate temporary credentials. Create an IAM role for that web provider and set up permissions for the IAM role to allow GET and PUT operations in Amazon S3 and DynamoDB. Serve your web app out of an S3 bucket enabled as a website** is correct because it authenticates the application via a federated identity provider such as Google, Facebook, Amazon, or other social sites. It sets up proper permission for DynamoDB access and hosts the website in S3. Plus, it also uses STS and AssumeRoleWithWebIdentity API which provides a better authentication.

The option that says: **Configure the ReactJS client with temporary credentials from the Security Token Service using a Token Vending Machine (TVM) on an EC2 instance. This will provide signed credentials to an IAM user allowing GET and PUT operations in the DynamoDB table and the S3 bucket. You serve your mobile application out of an S3 bucket enabled as a web site** is incorrect because the Token Vending Machine (STS Service) is implemented on just a single EC2 instance. This poses a single point of failure which means that the architecture is not highly available and not fault-tolerant. This is not a scalable solution either as the instance can become the performance bottleneck.

The option that says: **Configure the ReactJS client with temporary credentials from the Security Token Service using a Token Vending Machine (TVM) to provide signed credentials to an IAM user. This will allow GET and PUT operations to DynamoDB. Serve your web application from an NGINX server hosted in a fleet of EC2 instances that are load-balanced and auto-scaled. Your EC2 instances are configured with an IAM role that allows GET and PUT operations in DynamoDB** is incorrect. Although this solution is highly-available and scalable, deploying EC2 instances in an auto-scaled environment is not as a cost-effective solution as the S3 website.

The option that says: **Register the web application with a Web Identity Provider such as Google, Facebook, Amazon, or from any other popular social site. Create an IAM role for that web provider and set up permissions for the IAM role to allow GET and PUT operations in DynamoDB. Serve your web application from an NGINX server hosted on a fleet of EC2 instances, with a load balancer and auto-scaling. Add an IAM role to the EC2 instance to allow GET and PUT operations to DynamoDB tables** is incorrect because it does not mention any security token service that generates temporary credentials. Furthermore, deploying EC2 instances in an auto-scaled environment, albeit scalable, is not as a cost-effective solution as the S3 website.

**Reference:**  
  
<https://docs.aws.amazon.com/AmazonS3/latest/dev/WebsiteHosting.html>

**Check out this Amazon S3 Cheat Sheet:**

[https://tutorialsdojo.com/amazon-s3/](https://tutorialsdojo.com/amazon-s3/?src=udemy)

Bottom of Form

Top of Form

Question 27: Skipped

A telecommunications company is planning to host a WordPress website on an Amazon ECS Cluster which uses the Fargate launch type. For security purposes, the database credentials should be provided to the WordPress image by using environment variables. Your manager instructed you to ensure that the credentials are secure when passed to the image and that they cannot be viewed on the cluster itself. The credentials must be kept in a dedicated storage with lifecycle management and key rotation.

Which of the following is the most suitable solution in this scenario that you can implement with the least effort?

* ​

In the ECS task definition file of the ECS Cluster, store the database credentials using Docker Secrets to centrally manage this sensitive data and securely transmit it to only those containers that need access to it. Secrets are encrypted during transit and at rest. A given secret is only accessible to those services which have been granted explicit access to it via IAM Role, and only while those service tasks are running.

* ​

In the ECS task definition file of the ECS Cluster, store the database credentials and encrypt with KMS. Store the task definition JSON file in a private S3 bucket and ensure that HTTPS is enabled on the bucket to encrypt the data in-flight. Create an IAM role to the ECS task definiton script that allows access to the specific S3 bucket and then pass the --cli-input-json parameter when calling the ECS register-task-definition. Reference the task definition JSON file in the S3 bucket which contains the database credentials.

* ​

Store the database credentials using the AWS Secrets Manager and then encrypt them using AWS KMS. Create an IAM Role for your Amazon ECS task execution role and reference it with your task definition which allows access to both KMS and AWS Secrets Manager. Within your container definition, specify secrets with the name of the environment variable to set in the container and the full ARN of the Secrets Manager secret which contains the sensitive data, to present to the container.

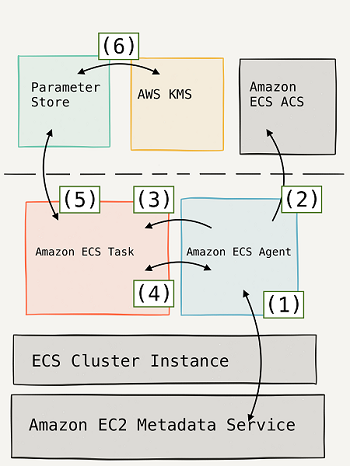
**(Correct)**

* ​

Store the database credentials using the AWS Systems Manager Parameter Store and then encrypt them using AWS KMS. Create an IAM Role for your Amazon ECS task execution role and reference it with your task definition, which allows access to both KMS and the Parameter Store. Within your container definition, specify secrets with the name of the environment variable to set in the container and the full ARN of the Systems Manager Parameter Store parameter containing the sensitive data to present to the container.

**Explanation**

**Amazon ECS** enables you to inject sensitive data into your containers by storing your sensitive data in either AWS Secrets Manager secrets or AWS Systems Manager Parameter Store parameters and then referencing them in your container definition. This feature is supported by tasks using both the EC2 and Fargate launch types.



Within your container definition, specify secrets with the name of the environment variable to set in the container and the full ARN of either the Secrets Manager secret or Systems Manager Parameter Store parameter containing the sensitive data to present to the container. The parameter that you reference can be from a different Region than the container using it, but must be from within the same account.

AWS Secrets Manager is a secrets management service that helps you protect access to your applications, services, and IT resources. This service enables you to easily rotate, manage, and retrieve database credentials, API keys, and other secrets throughout their lifecycle. Using Secrets Manager, you can secure and manage secrets used to access resources in the AWS Cloud, on third-party services, and on-premises.

If you want a single store for configuration and secrets, you can use Parameter Store. If you want a dedicated secrets store with lifecycle management, use Secrets Manager. Hence, the correct answer is the option that says: ***Store the database credentials using the AWS Secrets Manager and then encrypt them using AWS KMS. Create an IAM Role for your Amazon ECS task execution role and reference it with your task definition which allows access to both KMS and AWS Secrets Manager. Within your container definition, specify secrets with the name of the environment variable to set in the container and the full ARN of the Secrets Manager secret which contains the sensitive data, to present to the container.***

The option that says: ***Store the database credentials using the AWS Systems Manager Parameter Store and then encrypt them using AWS KMS. Create an IAM Role for your Amazon ECS task execution role and reference it with your task definition, which allows access to both KMS and the Parameter Store. Within your container definition, specify secrets with the name of the environment variable to set in the container and the full ARN of the Systems Manager Parameter Store parameter containing the sensitive data to present to the container*** is incorrect because although the use of Systems Manager Parameter Store in securing sensitive data in ECS is valid, this service doesn't provide dedicated storage with lifecycle management and key rotation, unlike Secrets Manager.

The option that says: ***In the ECS task definition file of the ECS Cluster, store the database credentials and encrypt with KMS. Store the task definition JSON file in a private S3 bucket and ensure that HTTPS is enabled on the bucket to encrypt the data in-flight. Create an IAM role to the ECS task definiton script that allows access to the specific S3 bucket and then pass the --cli-input-json parameter when calling the ECS register-task-definition. Reference the task definition JSON file in the S3 bucket which contains the database credentials*** is incorrect because although the solution may work, it is not recommended to store sensitive credentials in S3. This entails a lot of overhead and manual configuration steps which can be simplified by simply using the Secrets Manager or Systems Manager Parameter Store.

The option that says:***In the ECS task definition file of the ECS Cluster, store the database credentials using Docker Secrets to centrally manage this sensitive data and securely transmit it to only those containers that need access to it. Secrets are encrypted during transit and at rest. A given secret is only accessible to those services which have been granted explicit access to it via IAM Role, and only while those service tasks are running*** is incorrect because although you can use Docker Secrets to secure the sensitive database credentials, this feature is only applicable in Docker Swarm. In AWS, the recommended way to secure sensitive data is either through the use of Secrets Manager or Systems Manager Parameter Store.

**References:**

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/specifying-sensitive-data.html>

<https://aws.amazon.com/blogs/mt/the-right-way-to-store-secrets-using-parameter-store/>

**Check out this Amazon ECS Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-elastic-container-service-amazon-ecs/>

**Check out this AWS Secrets Manager Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-secrets-manager/>

Bottom of Form

Top of Form

Question 28: Skipped

A multinational investment bank has a hybrid cloud architecture that uses a single 1 Gbps AWS Direct Connect connection to integrate their on-premises network to AWS Cloud. The bank has a total of 10 VPCs which are all connected to their on-premises data center via the same Direct Connect connection that you manage. Based on the recent IT audit, the existing network setup has a single point of failure which needs to be addressed immediately.

Which of the following is the MOST cost-effective solution that you should implement in order to improve the connection redundancy of your hybrid network?

* ​

Establish another 1 Gbps AWS Direct Connect connection with corresponding private Virtual Interfaces (VIFs) to connect all of the 10 VPCs individually. Set up a Border Gateway Protocol (BGP) peering session for all of the VIFs.

* ​

Establish a new point-to-point Multiprotocol Label Switching (MPLS) connection to all of your 10 VPCs. Configure BGP to use this new connection with an active/passive routing.

* ​

Establish VPN tunnels from your on-premises data center to each of the 10 VPCs. Terminate each VPN tunnel connection at the virtual private gateway (VGW) of the respective VPC. Configure BGP for route management.

**(Correct)**

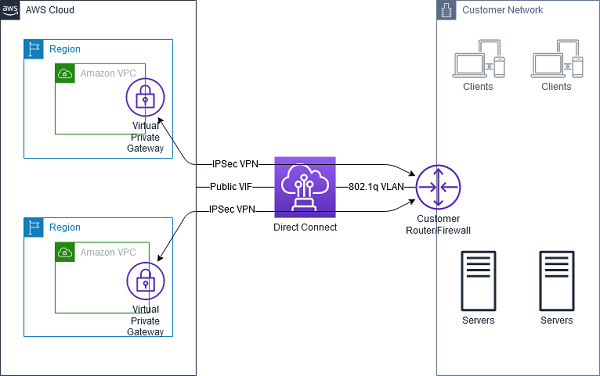
* ​

Establish another 1 Gbps AWS Direct Connect connection using a public Virtual Interface (VIF). Prepare a VPN tunnel that will terminate on the virtual private gateway (VGW) of the respective VPC using the public VIF. Handle the failover to the VPN connection through the use of BGP.

**Explanation**

With AWS Direct Connect plus VPN, you can combine one or more AWS Direct Connect dedicated network connections with the Amazon VPC VPN. This combination provides an IPsec-encrypted private connection that also reduces network costs, increases bandwidth throughput, and provides a more consistent network experience than Internet-based VPN connections.

You can use AWS Direct Connect to establish a dedicated network connection between your network and create a logical connection to public AWS resources, such as an Amazon virtual private gateway IPsec endpoint. This solution combines the AWS managed benefits of the VPN solution with low latency, increased bandwidth, more consistent benefits of the AWS Direct Connect solution, and an end-to-end, secure IPsec connection.



It costs a lot of money to establish a Direct Connect connection which you rarely use. For a more cost-effective solution, you can configure a backup VPN connection for failover with your AWS Direct Connect connection.

If you want a short-term or lower-cost solution, you might consider configuring a hardware VPN as a failover option for a Direct Connect connection. VPN connections are not designed to provide the same level of bandwidth available to most Direct Connect connections. Ensure that your use case or application can tolerate a lower bandwidth if you are configuring a VPN as a backup to a Direct Connect connection.

Hence, the correct answer is: **Establish VPN tunnels from your on-premises data center to each of the 10 VPCs. Terminate each VPN tunnel connection at the virtual private gateway (VGW) of the respective VPC. Configure BGP for route management.**

The following options are incorrect:

**- Establish another 1 Gbps AWS Direct Connect connection using a public Virtual Interface (VIF). Prepare a VPN tunnel that will terminate on the virtual private gateway (VGW) of the respective VPC using the public VIF. Handle the failover to the VPN connection through the use of BGP.**

**- Establish another 1 Gbps AWS Direct Connect connection with corresponding private Virtual Interfaces (VIFs) to connect all of the 10 VPCs individually. Set up a Border Gateway Protocol (BGP) peering session for all of the VIFs.**

Establishing yet another 1 Gbps AWS Direct Connect connection is not a cost-effective solution. It is better to establish a VPN connection instead as a backup.

The option that says: **Establishing a new point-to-point Multiprotocol Label Switching (MPLS) connection to all of your 10 VPCs and Configuring BGP to use this new connection with an active/passive routing** is incorrect because you can't directly connect to your Multiprotocol Label Switching (MPLS) to AWS. To integrate your MPLS infrastructure, you need to set up a colocation with Direct Connect by placing the CGW in the same physical facility as the Direct Connect location, which will facilitate a local cross-connect between the CGW and AWS devices.

**References:**

<https://aws.amazon.com/premiumsupport/knowledge-center/configure-vpn-backup-dx/>

<https://docs.aws.amazon.com/whitepapers/latest/aws-vpc-connectivity-options/aws-direct-connect-plus-vpn-network-to-amazon.html>

<https://aws.amazon.com/answers/networking/aws-network-connectivity-over-mpls/>

**Check out this AWS Direct Connect Cheat Sheet:**

[https://tutorialsdojo.com/aws-direct-connect/](https://tutorialsdojo.com/aws-direct-connect/?src=udemy)

Bottom of Form

Top of Form

Question 29: Skipped

An online delivery system, hosted in a fleet of EC2 instances, is deployed in multiple Availability Zones in the ap-southeast-1 region with an Application Load Balancer that evenly distributes the load. The system is using a MySQL RDS instance to store the deliveries and transactions of the system. To ensure business continuity, you are instructed to set up a disaster recovery system in which the RTO must be less than 3 hours and the RPO is 15 minutes when a system outage occurs.A system should also be implemented that can automatically discover, classify, and protect any personally identifiable information (PII) or intellectual property in your data store.

As the Solutions Architect, which disaster recovery strategy should you use to achieve the required RTO and RPO targets in the most cost-effective manner?

* ​

Schedule a database backup to AWS Storage Gateway every hour and store transaction logs to a separate S3 bucket every 5 minutes. Use AWS Shield to automatically discover, classify, and protect any personally identifiable information (PII) or intellectual property on your Storage Gateway.

* ​

Schedule 15-minute DB backups to Amazon Glacier. Store the transaction logs to an S3 bucket every 5 minutes. Use Amazon Macie to automatically discover, classify, and protect your sensitive data.

* ​

Schedule a database backup to an S3 bucket every hour and store transaction logs to a separate S3 bucket every 5 minutes. Use Amazon Macie to automatically discover, classify, and protect your sensitive data.

**(Correct)**

* ​

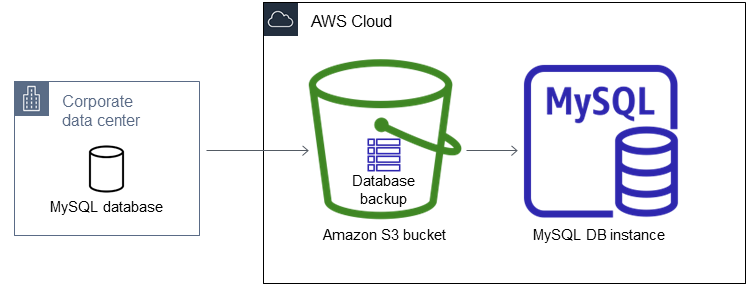
Set up asynchronous replication in the database using a Multi-AZ deployments configuration. Use AWS Shield to automatically discover, classify, and protect any personally identifiable information (PII) or intellectual property from your RDS database.

**Explanation**

Recovery time objective (RTO) is the time it takes after a disruption to restore a business process to its service level, as defined by the operational level agreement (OLA). For example, if a disaster occurs at 12:00 PM (noon) and the RTO is eight hours, the DR process should restore the business process to the acceptable service level by 8:00 PM.

Recovery point objective (RPO) is the acceptable amount of data loss measured in time. For example, if a disaster occurs at 12:00 PM (noon) and the RPO is one hour, the system should recover all data that was in the system before 11:00 AM. Data loss will span only one hour, between 11:00 AM and 12:00 PM (noon).

Amazon S3 is an ideal destination for backup data that might be needed quickly to perform a restore. Transferring data to and from Amazon S3 is typically done through the network, and is therefore accessible from any location.



Amazon Macie is a security service that uses machine learning to automatically discover, classify, and protect sensitive data in AWS. Amazon Macie recognizes sensitive data such as personally identifiable information (PII) or intellectual property, and provides you with dashboards and alerts that give visibility into how this data is being accessed or moved. The fully managed service continuously monitors data access activity for anomalies, and generates detailed alerts when it detects risk of unauthorized access or inadvertent data leaks.

Hence, the option that says:***Schedule a database backup to an S3 bucket every hour and store transaction logs to a separate S3 bucket every 5 minutes. Use Amazon Macie to automatically discover, classify, and protect your sensitive data*** is correct because by using an S3 bucket, it makes data retrievals of the backups quicker. Since the transaction logs are stored in S3 every 5 minutes, this will help to restore the application to a state that is within the required RPO of 15 minutes.

***Scheduling 15-minute DB Backups to Amazon Glacier and storing the transaction logs to an S3 bucket every 5 minutes, and using Amazon Macie to automatically discover, classify, and protect your sensitive data*** is incorrect because retrieving the database backups from Amazon Glacier archives will normally take around 3 - 5 hours using the Standard Retrievals and hence, this will not meet the RTO and RPO. Although you can use expedited retrievals, which can typically retrieve your archive within 1 – 5 minutes, this will entail additional cost and hence, not a cost-effective solution.

***Setting up asynchronous replication in the database using a Multi-AZ deployments configuration and using AWS Shield to automatically discover, classify, and protect any personally identifiable information (PII) or intellectual property from your RDS database*** is incorrect because asynchronous replication is only applicable for Read Replicas and not for Multi-AZ deployments configuration. Although this will improve the availability of the RDS database, it won't provide a better RTO or RPO, especially in the event of a regional outage since you can't export a standby instance to another region. In addition, you have to use AWS Macie to protect your sensitive data in Amazon S3 and not AWS Shield.

***Scheduling a database backup to AWS Storage Gateway every hour and storing transaction logs to a separate S3 bucket every 5 minutes and using AWS Shield to automatically discover, classify, and protect any personally identifiable information (PII) or intellectual property on your Storage Gateway***is incorrect because AWS Storage Gateway is primarily used for hybrid cloud storage service that connects your existing on-premises environments with the AWS Cloud. Although this can be a valid option, the scenario did not say that their architecture is hybrid or that they are using an on-premises data center. AWS Shield is primarily used to protect your resources from DDoS attacks, and not to automatically discover, classify, and protect any personally identifiable information (PII) or intellectual property.

**References:**

<https://www.slideshare.net/AmazonWebServices/disaster-recovery-options-with-aws>

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PIT.html>

**Check out this Amazon Macie Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-macie/>

Bottom of Form

Top of Form

Question 30: Skipped

A multinational financial company has a suite of web applications hosted in multiple VPCs in various AWS regions. As part of their security compliance, the company’s Solutions Architect has been tasked to set up a logging solution to track all of the changes made to their AWS resources in all regions, which host their enterprise accounting systems. The company is using different AWS services such as Amazon EC2 instances, Amazon S3 buckets, CloudFront web distributions, and AWS IAM. The logging solution must ensure the security, integrity, and durability of your log data in order to pass the compliance requirements. In addition, it should provide an event history of your AWS account activity, including actions taken through the AWS Management Console, AWS SDKs, command-line tools, and API calls.

In this scenario, which of the following options is the best solution to use?

* ​

Create a new AWS CloudTrail trail in a new S3 bucket using the AWS CLI and also pass the --no-include-global-service-events and --is-multi-region-trail parameter then encrypt log files using KMS encryption. Enable Multi-Factor Authentication (MFA) Delete on the S3 bucket and ensure that only authorized users can access the logs by configuring the bucket policies.

* ​

Create a new Amazon CloudWatch trail in a new S3 bucket using the AWS CLI and also pass the --include-global-service-events parameter then encrypt log files using KMS encryption. Enable Multi-Factor Authentication (MFA) Delete on the S3 bucket and ensure that only authorized users can access the logs by configuring the bucket policies.

* ​

Create a new Amazon CloudWatch trail in a new S3 bucket using the AWS CLI and also pass both the --is-multi-region-trail and --include-global-service-events parameters then encrypt log files using KMS encryption. Enable Multi-Factor Authentication (MFA) Delete on the S3 bucket and ensure that only authorized users can access the logs by configuring the bucket policies.

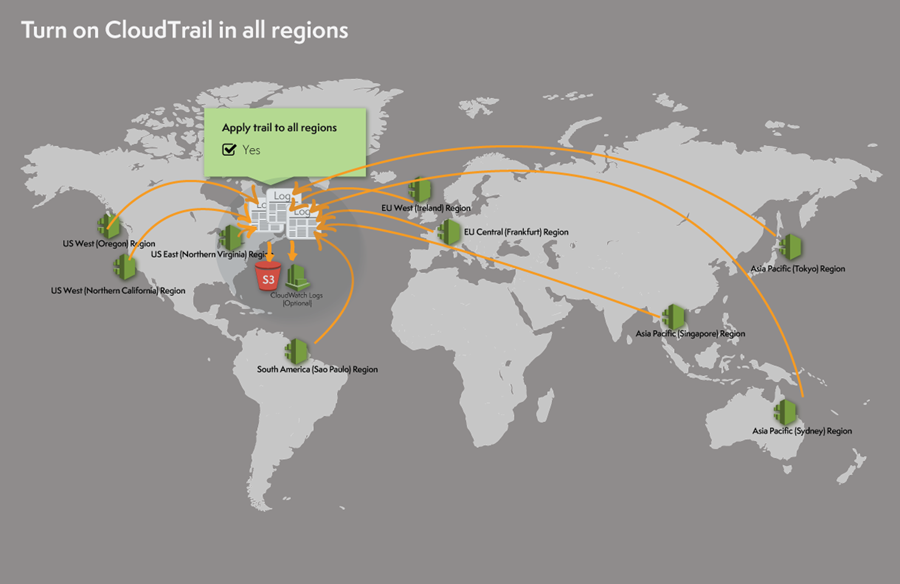
* ​

Create a new AWS CloudTrail trail in a new S3 bucket using the AWS CLI and also pass both the --is-multi-region-trail and --include-global-service-events parameters then encrypt log files using KMS encryption. Enable Multi-Factor Authentication (MFA) Delete on the S3 bucket and ensure that only authorized users can access the logs by configuring the bucket policies.

**(Correct)**

**Explanation**

The accounting firm requires a secure and durable logging solution that will track all of the activities of all AWS resources (such as EC2, S3, CloudFront, and IAM) on all regions. CloudTrail can be used for this case with multi-region trail enabled. However, CloudTrail will only cover the activities of the regional services (EC2, S3, RDS etc.) and not for global services such as IAM, CloudFront, AWS WAF, and Route 53.



The option that says: **Create a new AWS CloudTrail trail in a new S3 bucket using the AWS CLI and also pass both the --is-multi-region-trail and --include-global-service-events parameters then encrypt log files using KMS encryption. Enable Multi-Factor Authentication (MFA) Delete on the S3 bucket and ensure that only authorized users can access the logs by configuring the bucket policies** is correct because it provides security, integrity, and durability to your log data. In addition, it has the -include-global-service-events parameter enabled which will also include activity from global services such as IAM, Route 53, AWS WAF, and CloudFront.

The option that says: **Create a new Amazon CloudWatch trail in a new S3 bucket using the AWS CLI and also pass both the --is-multi-region-trail and --include-global-service-events parameters then encrypt log files using KMS encryption. Enable Multi-Factor Authentication (MFA) Delete on the S3 bucket and ensure that only authorized users can access the logs by configuring the bucket policies** is incorrect because you need to use CloudTrail instead of CloudWatch.

The option that says: **Create a new Amazon CloudWatch trail in a new S3 bucket using the AWS CLI and also pass the --include-global-service-events parameter then encrypt log files using KMS encryption. Enable Multi-Factor Authentication (MFA) Delete on the S3 bucket and ensure that only authorized users can access the logs by configuring the bucket policies** is incorrect because you need to use CloudTrail instead of CloudWatch. In addition, the --is-multi-region-trail parameter is also missing in this setup.

The option that says: **Create a new AWS CloudTrail trail in a new S3 bucket using the AWS CLI and also pass the --no-include-global-service-events and --is-multi-region-trail parameter then encrypt log files using KMS encryption. Enable Multi-Factor Authentication (MFA) Delete on the S3 bucket and ensure that only authorized users can access the logs by configuring the bucket policies** is incorrect because the --is-multi-region-trail is not enough as you also need to add the --include-global-service-events parameter to track the global service events. The --no-include-global-service-events parameter actually prevents CloudTrail from publishing events from global services such as IAM to the log files.

**References:**

<https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-concepts.html#cloudtrail-concepts-global-service-events>

<http://docs.aws.amazon.com/IAM/latest/UserGuide/cloudtrail-integration.html>

<https://docs.aws.amazon.com/awscloudtrail/latest/userguide/cloudtrail-create-and-update-a-trail-by-using-the-aws-cli.html>

**Check out this AWS CloudTrail Cheat Sheet:**

[https://tutorialsdojo.com/aws-cloudtrail/](https://tutorialsdojo.com/aws-cloudtrail/?src=udemy)

Bottom of Form

Top of Form

Question 31: Skipped

You are working as a Solutions Architect for a multinational tech company which has multiple VPCs for each of its IT departments. You are instructed to launch a new central database server which can be accessed by the other VPCs of the company using the database.tutorialsdojo.com domain name. This server should only be accessible within the associated VPCs since only internal applications will be using the database.

Which of the following should you do to meet the above requirement?

* ​

Set up a private hosted zone with a domain name of tutorialsdojo.com and specify the VPCs that you want to associate with the hosted zone. Create an A record with a value of database.tutorialsdojo.com which maps to the Elastic IP address of the EC2 instance of your database server. Modify the enableDnsHostNames attribute of your VPC to true and the enableDnsSupport attribute to false

* ​

Set up a public hosted zone with a domain name of tutorialsdojo.com and specify the VPCs that you want to associate with the hosted zone. Create a CNAME record with a value of database.tutorialsdojo.com which maps to the IP address of the EC2 instance of your database server. Modify the enableDnsHostNames attribute of your VPC to false and the enableDnsSupport attribute to false

* ​

Set up a public hosted zone with a domain name of tutorialsdojo.com and specify the VPCs that you want to associate with the hosted zone. Create an A record with a value of database.tutorialsdojo.com which maps to the IP address of the EC2 instance of your database server. Modify the enableDnsHostNames attribute of your VPC to true and the enableDnsSupport attribute to true

* ​

Set up a private hosted zone with a domain name of tutorialsdojo.com and specify the VPCs that you want to associate with the hosted zone. Create an A record with a value of database.tutorialsdojo.com which maps to the IP address of the EC2 instance of your database server. Modify the enableDnsHostNames attribute of your VPC to true and the enableDnsSupport attribute to true

**(Correct)**

**Explanation**

In AWS, a hosted zone is a container for records, and records contain information about how you want to route traffic for a specific domain, such as tutorialsdojo.com, and its subdomains (portal.tutorialsdojo.com, database.tutorialsdojo.com). A hosted zone and the corresponding domain have the same name. There are two types of hosted zones:

*Public hosted zones* contain records that specify how you want to route traffic on the internet.

*Private hosted zones* contain records that specify how you want to route traffic in an Amazon VPC

A *private hosted zone* is a container that holds information about how you want Amazon Route 53 to respond to DNS queries for a domain and its subdomains within one or more VPCs that you create with the Amazon VPC service. Your VPC has attributes that determine whether your EC2 instance receives public DNS hostnames, and whether DNS resolution through the Amazon DNS server is supported.

**enableDnsHostnames** - Indicates whether the instances launched in the VPC get public DNS hostnames. If this attribute is true, instances in the VPC get public DNS hostnames, but only if the enableDnsSupport attribute is also set to true.

**enableDnsSupport** - Indicates whether the DNS resolution is supported for the VPC. If this attribute is false, the Amazon-provided DNS server in the VPC that resolves public DNS hostnames to IP addresses is not enabled. If this attribute is true, queries to the Amazon provided DNS server at the 169.254.169.253 IP address, or the reserved IP address at the base of the VPC IPv4 network range plus two ( \*.\*.\*.2 ) will succeed.

Hence, the option that says: ***Set up a private hosted zone with a domain name of tutorialsdojo.com and specify the VPCs that you want to associate with the hosted zone. Create an A record with a value of database.tutorialsdojo.com which maps to the IP address of the EC2 instance of your database server. Modify the enableDnsHostNames attribute of your VPC to true and the enableDnsSupport attribute to true*** is the correct answer.

The options that say:

***1. Set up a public hosted zone with a domain name of tutorialsdojo.com and specify the VPCs that you want to associate with the hosted zone. Create an A record with a value of database.tutorialsdojo.com which maps to the IP address of the EC2 instance of your database server. Modify the enableDnsHostNames attribute of your VPC to true and the enableDnsSupport attribute to true***

***2. Set up a public hosted zone with a domain name of tutorialsdojo.com and specify the VPCs that you want to associate with the hosted zone. Create a CNAME record with a value of database.tutorialsdojo.com which maps to the IP address of the EC2 instance of your database server. Modify the enableDnsHostNames attribute of your VPC to false and the enableDnsSupport attribute to false***

are incorrect because you have to create a ***private*** hosted zone and not a public one, since the database server will only be accessed by the associated VPCs and not publicly over the Internet. In addition, you have to create an A record for your database server and then set both the **enableDnsHostNames** and **enableDnsSupport** attributes to true.

The option that says: ***Set up a private hosted zone with a domain name of tutorialsdojo.com and specify the VPCs that you want to associate with the hosted zone. Create an A record with a value of database.tutorialsdojo.com which maps to the Elastic IP address of the EC2 instance of your database server. Modify the enableDnsHostNames attribute of your VPC to true and the enableDnsSupport attribute to false*** is incorrect because even though it mentions the use of a private hosted zone, the configuration is incorrect since it is required to set both the **enableDnsHostNames** and **enableDnsSupport** attributes of your VPC to true. In addition, an Elastic IP address is a public IPv4 address, which is reachable from the Internet and hence, it violates the requirement that the database server should only be accessible within your associated VPCs.

**References:**

<https://docs.aws.amazon.com/vpc/latest/userguide/vpc-dns.html#vpc-dns-support>

<https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/hosted-zones-private.html>

**Check out these Amazon VPC and Route 53 Cheat Sheets:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-vpc/>

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-route-53/>

Bottom of Form

Top of Form

Question 32: Skipped

A company hosts its multi-tiered web application on a fleet of Auto Scaling EC2 instances spread across two Availability Zones. The Application Load Balancer is in the public subnets and the Amazon EC2 instances are in the private subnets. After a few weeks of operations, the users are reporting that the web application is not working properly. Upon testing, the Solutions Architect found that the website is accessible and the login is successful. However, when the “find a nearby store” function is clicked on the website, the map loads only about 50% of the time when the page is refreshed. This function involves a third-party RESTful API call to a maps provider. Amazon EC2 NAT instances are used for these outbound API calls.

Which of the following options are the MOST likely reason for this failure and the recommended solution?

* ​

This error is caused by failed NAT instance in one of the public subnets. Use NAT Gateways instead of EC2 NAT instances to ensure availability and scalability.

**(Correct)**

* ​

The error is caused by a failure in one of their availability zones in the VPC of the third-party provider. Contact the third-party provider support hotline and request for them to fix it.

* ​

This error is caused by an overloaded NAT instance in one of the subnets. Scale the EC2 NAT instances to larger-sized instances to ensure that they can handle the growing traffic.

* ​

One of the subnets in the VPC has a misconfigured Network ACL that blocks outbound traffic to the third-party provider. Update the network ACL to allow this connection and configure IAM permissions to restrict these changes in the future.

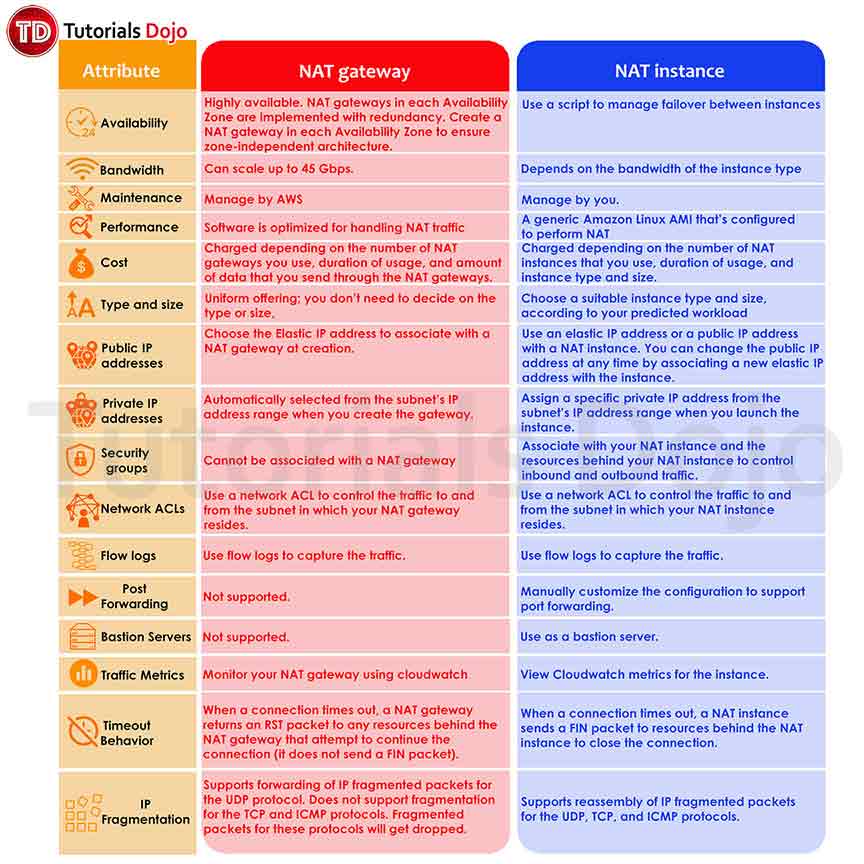
**Explanation**

You can use a NAT device to enable instances in a private subnet to connect to the Internet (for example, for software updates) or other AWS services, but prevent the Internet from initiating connections with the instances. A NAT device forwards traffic from the instances in the private subnet to the Internet or other AWS services, and then sends the response back to the instances. When traffic goes to the Internet, the source IPv4 address is replaced with the NAT device’s address, and similarly, when the response traffic goes to those instances, the NAT device translates the address back to those instances’ private IPv4 addresses.

You can either use a managed NAT device offered by AWS called a NAT gateway, or you can create your own NAT device in an EC2 instance, referred to here as a **NAT instance**. The bandwidth of NAT instances depends on the instance size – higher instances size will have high bandwidth capacity. NAT instances are managed by the customer so if the instance goes down, there could be a potential impact on the availability of your application. You have to manually check and fix the NAT instances.

AWS recommends **NAT gateways** because they provide better availability and bandwidth over NAT instances. The NAT gateway service is also a managed service that does not require your administration efforts. NAT Gateways are highly available. In each Availability Zone, they are implemented with redundancy. It is managed by AWS so you do not have to perform maintenance or monitoring if it is UP. NAT gateways automatically scale in bandwidth so you don’t have to choose instance types.

Check out the full comparison in the table below:



The correct answer is: **This error is caused by failed NAT instance in one of the public subnets. Use NAT Gateways instead of EC2 NAT instances to ensure availability and scalability.**This is very likely as we have two subnets in the scenario and NAT instances reside in only one AZ. With a failure rate of 50%, one of the NAT instances must have been down. AWS does not automatically recover the failed NAT instances. AWS recommends using NAT gateways because they provide better availability and bandwidth. Even if NAT gateway is deployed on a single AZ, AWS implements redundancy to ensure that it is always available on that AZ.

The option that says: **One of the subnets in the VPC has a misconfigured Network ACL that blocks outbound traffic to the third-party provider. Update the network ACL to allow this connection and configure IAM permissions to restrict these changes in the future** is incorrect. Network ACLs affect all the subnets associated with it. If there is a misconfigured rule, the other subnets will be affected too, which could result in a 100% failure of requests to the third-party provider.

The option that says: **The error is caused by a failure in one of their availability zones in the VPC of the third-party provider. Contact the third-party provider support hotline and request for them to fix it**is incorrect. If there is a failure on one availability zone of the third-party provider, the traffic should have stopped sending to that AZ so this failure is most likely caused by a local failure in your VPC.

The option that says: **This error is caused by an overloaded NAT instance in one of the subnets. Scale the EC2 NAT instances to larger-sized instances to ensure that they can handle the growing traffic** is incorrect. If the NAT instances are overloaded, you will notice inconsistent performance or slowdown for the third-party requests. And this failure should have been gone during off-peak hours. If the failure rate is 50% of the requests, it is most likely that one of the NAT instances is down.

**References:**

<https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-comparison.html>

<https://docs.aws.amazon.com/vpc/latest/userguide/VPC_NAT_Instance.html>

<https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html>

**Check out this Amazon VPC Cheat Sheet:**

[https://tutorialsdojo.com/amazon-vpc/](https://tutorialsdojo.com/amazon-vpc/?src=udemy)

Bottom of Form

Top of Form

Question 33: Skipped

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 Lamdba  
    functions without increasing the execution time.

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

* ​

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 Lamdba functions without increasing the execution time.

4. Configure Auto Scaling on Amazon DynamoDB to automatically adjust the capacity based on user traffic.

**(Correct)**

* ​

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 Sport  
    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. 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 Lamdba  
    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.

**Explanation**

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

You can enable API caching in Amazon API Gateway to cache your endpoint's responses. With caching, you can reduce the number of calls made to your endpoint and also improve the latency of requests to your API.

When you enable caching for a stage, API Gateway caches responses from your endpoint for a specified time-to-live (TTL) period, in seconds. API Gateway then responds to the request by looking up the endpoint response from the cache instead of making a request to your endpoint. The default TTL value for API caching is 300 seconds. The maximum TTL value is 3600 seconds. TTL=0 means caching is disabled.

With **AWS Lambda**, you pay only for what you use. You are charged based on the number of requests for your functions and the duration, the time it takes for your code to execute. Lambda counts a request each time it starts executing in response to an event notification or invoke call, including test invokes from the console.

Duration is calculated from the time your code begins executing until it returns or otherwise terminates, rounded up to the nearest 1ms\*. The price depends on the amount of memory you allocate to your function. In the AWS Lambda resource model, you choose the amount of memory you want for your function and are allocated proportional CPU power and other resources. An increase in memory size triggers an equivalent increase in CPU available to your function.

**Auto Scaling for DynamoDB** helps automate capacity management for your tables and global secondary indexes. You simply specify the desired target utilization and provide upper and lower bounds for read and write capacity. DynamoDB will then monitor throughput consumption using Amazon CloudWatch alarms and then will adjust provisioned capacity up or down as needed.

Amazon DynamoDB auto scaling uses the AWS Application Auto Scaling service to dynamically adjust provisioned throughput capacity on your behalf, in response to actual traffic patterns. This enables a table or a global secondary index to increase its provisioned read and write capacity to handle sudden increases in traffic, without throttling. When the workload decreases, Application Auto Scaling decreases the throughput so that you don't pay for unused provisioned capacity.

Therefore, the following option is correct:

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

**- Configure API caching on Amazon API Gateway to reduce the overall number of invocations to the Lambda functions.**

**- Gradually lower the timeout and memory properties of the Lamdba functions without increasing the execution time.**

**- Configure Auto Scaling on Amazon DynamoDB to automatically adjust the capacity based on user traffic.**

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 Lamdba 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 Sport 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 Lamdba 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.

**References:**

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

<https://aws.amazon.com/api-gateway/faqs/>

<https://aws.amazon.com/blogs/aws/new-auto-scaling-for-amazon-dynamodb/>

<https://aws.amazon.com/lambda/pricing/>

**Check out these Amazon API Gateway, Amazon DynamoDB, and AWS Lambda Cheat Sheets:**

[https://tutorialsdojo.com/amazon-api-gateway/](https://tutorialsdojo.com/amazon-api-gateway/?src=udemy)

[https://tutorialsdojo.com/amazon-dynamodb/](https://tutorialsdojo.com/amazon-dynamodb/?src=udemy)

[https://tutorialsdojo.com/aws-lambda/](https://tutorialsdojo.com/aws-lambda/?src=udemy)

Bottom of Form

Top of Form

Question 34: Skipped

A new online banking portal has been recently deployed to an Auto Scaling group of EC2 instances on your VPC. However, after several days, you found out that there is an unusually high amount of inbound HTTP traffic coming from a set of 15 specific IP addresses from a certain country where your company has absolutely no customers. The EC2 instances are flooded with incoming requests that the system administrators cannot even establish an SSH connection to the instances.

What should a Solutions Architect do to fix this security vulnerability in the MOST cost-effective way?

* ​

Use AWS WAF to protect your VPC against web attacks. Place the online banking portal behind a CloudFront RTMP distribution.

* ​

Use AWS Shield Advanced to protect your VPC from common, most frequently occurring network and transport layer DDoS attacks.

* ​

Set up deny rules on your inbound Network Access control list associated with the web application tier subnet to block access to the group of attacking IP addresses.

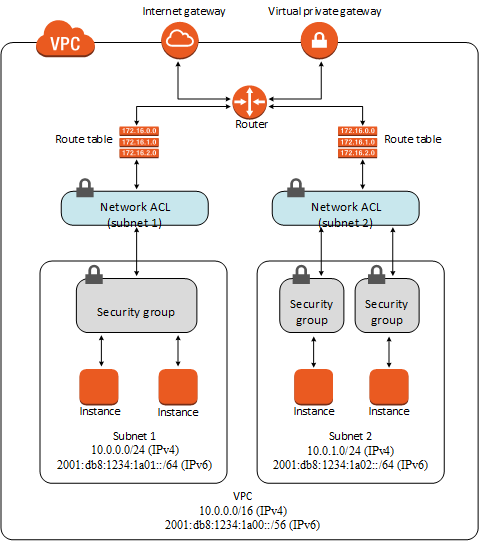
**(Correct)**

* ​

Create inbound rules in the Security Group of your EC2 instances to block the attacking IP addresses.

**Explanation**

A network access control list (ACL) is an optional layer of security for your VPC that acts as a firewall for controlling traffic in and out of one or more subnets. You might set up network ACLs with rules similar to your security groups in order to add an additional layer of security to your VPC.



***Setting up deny rules on your inbound Network Access control list associated with the web application tier subnet to block access to the group of attacking IP addresses*** is correct because you can add deny rules in Network ACL and block access to certain IP addresses.

***Using AWS WAF to protect your VPC against web attacks and placing the online banking portal behind a CloudFront RTMP distribution*** is incorrect because although you can use AWS WAF to protect your application from common web attacks, placing the online banking portal behind a CloudFront RTMP distribution is wrong because this is primarily used for real-time streaming and not for web applications. You should place it on a CloudFront Web distribution instead.

***Using AWS Shield Advanced to protect your VPC from common, most frequently occurring network and transport layer DDoS attacks*** is incorrect because although you can definitely use AWS Shield Advanced in protecting your cloud infrastructure, it is not cost-effective as compared to blocking the IP addresses using Network ACL. Take note that there are two types of AWS Shield: the Standard one which is free and the Advanced type which has an additional cost of around $3,000 per month.

***Creating inbound rules in the Security Group of your EC2 instances to block the attacking IP addresses*** is incorrect because you cannot deny port access as well as specific IP addresses using security groups. By default, all requests are denied in your security groups so you are required to explicitly allow access from a particular IP address, port or range to access your EC2 instances using your security groups.

**Reference**:

<https://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/VPC_ACLs.html>

**Check out this Amazon VPC Cheat Sheet:**

[https://tutorialsdojo.com/amazon-vpc/](https://tutorialsdojo.com/amazon-vpc/?src=udemy)

Bottom of Form

Top of Form

Question 35: Skipped

A company is using AWS Organizations to manage their multi-account and multi-region AWS infrastructure. They are currently doing large-scale automation for their key daily processes to save costs. One of these key processes is sharing specified AWS resources, which an organizational account owns, with other AWS accounts of the company using AWS RAM. There is already an existing service which was previously managed by a separate organization account moderator, who also maintained the specific configuration details.

In this scenario, what could be a simple and effective solution that would allow the service to perform its tasks on the organization accounts on the moderator's behalf?

* ​

Attach an IAM role on the service detailing all the allowed actions that it will be able to perform. Install an SSM agent in each of the worker VMs. Use AWS Systems Manager to build automation workflows that involve the daily key processes.

* ​

Use trusted access by running the enable-sharing-with-aws-organization command in the AWS RAM CLI. Mirror the configuration changes that was performed by the account that previously managed this service.

**(Correct)**

* ​

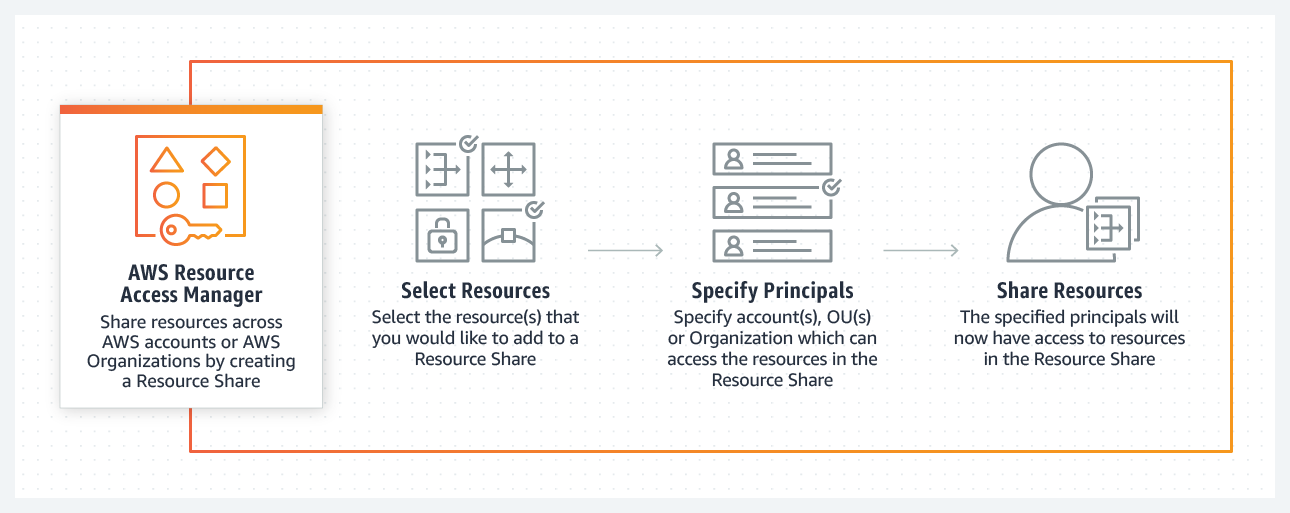
Configure a service-linked role for AWS RAM and modify the permissions policy to specify what the role can and cannot do. Lastly, modify the trust policy of the role so that other processes can utilize AWS RAM.

* ​

Enable cross-account access with AWS Organizations in the Resource Access Manager Console. Mirror the configuration changes that was performed by the account that previously managed this service.

**Explanation**

You can use ***trusted access*** to enable an AWS service that you specify, called the *trusted service*, to perform tasks in your organization and its accounts on your behalf. This involves granting permissions to the trusted service but does not otherwise affect the permissions for IAM users or roles. When you enable access, the trusted service can create an IAM role called a service-linked role in every account in your organization. That role has a permissions policy that allows the trusted service to do the tasks that are described in that service's documentation. This enables you to specify settings and configuration details that you would like the trusted service to maintain in your organization's accounts on your behalf.



AWS Resource Access Manager (AWS RAM) enables you to share specified AWS resources that you own with other AWS accounts. To enable trusted access with AWS Organizations:

From the AWS RAM CLI, use the enable-sharing-with-aws-organizations command.

Name of the IAM service-linked role that can be created in accounts when trusted access is enabled: *AWSResourceAccessManagerServiceRolePolicy*.

The option that says: ***Attach an IAM role on the service detailing all the allowed actions that it will be able to perform. Install an SSM agent in each of the worker VMs. Use AWS Systems Manager to build automation workflows that involve the daily key processes*** is incorrect because this is not the simplest way to automate interaction of AWS RAM with AWS Organizations. AWS Systems Manager is a tool that helps with automation of EC2 instances, on-premises servers, and other virtual machines. It might not support all the services being used by the key processes.

The option that says: ***Configure a service-linked role for AWS RAM and modify the permissions policy to specify what the role can and cannot do. Lastly, modify the trust policy of the role so that other processes can utilize AWS RAM*** is incorrect. This is not the simplest solution for integrating AWS RAM and AWS Organizations since using AWS Organization's trusted access will create the service-linked role for you. Also, the trust policy of a service-linked role cannot be modified. Only the linked AWS service can assume a service-linked role, which is why you cannot modify the trust policy of a service-linked role.

The option that says: ***Enable cross-account access with AWS Organizations in the Resources Access Manager Console. Mirror the configuration changes that was performed by the account that previously managed this service*** is incorrect because you should enable trusted access to AWS RAM, not a cross-account access.

**References:**

<https://docs.aws.amazon.com/organizations/latest/userguide/orgs_integrate_services.html>

<https://docs.aws.amazon.com/organizations/latest/userguide/services-that-can-integrate-ram.html>

<https://aws.amazon.com/blogs/security/introducing-an-easier-way-to-delegate-permissions-to-aws-services-service-linked-roles/>

Bottom of Form

Top of Form

Question 36: Skipped

A company wants to launch its online shopping website to give customers an easy way to purchase the products they need. The proposed setup is to host the application on an AWS Fargate cluster, utilize a Load Balancer to distribute traffic between the Fargate tasks, and use Amazon CloudFront for caching and content delivery. The company wants to ensure that the website complies with industry best practices and should be able to protect customers from common “man-in-the-middle” attacks for e-commerce websites such as DNS spoofing, HTTPS spoofing, or SSL hijacking.

Which of the following configurations will provide the MOST secure access to the website?

* ​

Register the domain name on Route 53. Use a third-party DNS provider that supports the import of the customer-managed keys for DNSSEC. Import a 2048-bit TLS/SSL certificate from a third-party certificate service to AWS Certificate Manager (ACM). Configure the Application Load Balancer with an HTTPS listener to use the imported TLS/SSL certificate. Use Server Name Identification and HTTP to HTTPS redirection on CloudFront.

* ​

Use Route 53 for domain registration. Use a third-party DNS service that supports DNSSEC for DNS requests that use the customer-managed keys. Use AWS Certificate Manager (ACM) to generate a valid 2048-bit TLS/SSL certificate for the domain name and configure the Application Load Balancer HTTPS listener to use this TLS/SSL certificate. Use Server Name Identification and HTTP to HTTPS redirection on CloudFront.

* ​

Register the domain name on Route 53 and enable DNSSEC validation for all public hosted zones to ensure that all DNS requests have not been tampered with during transit. Use AWS Certificate Manager (ACM) to generate a valid TLS/SSL certificate for the domain name. Configure the Application Load Balancer with an HTTPS listener to use the ACM TLS/SSL certificate. Use Server Name Identification and HTTP to HTTPS redirection on CloudFront.

**(Correct)**

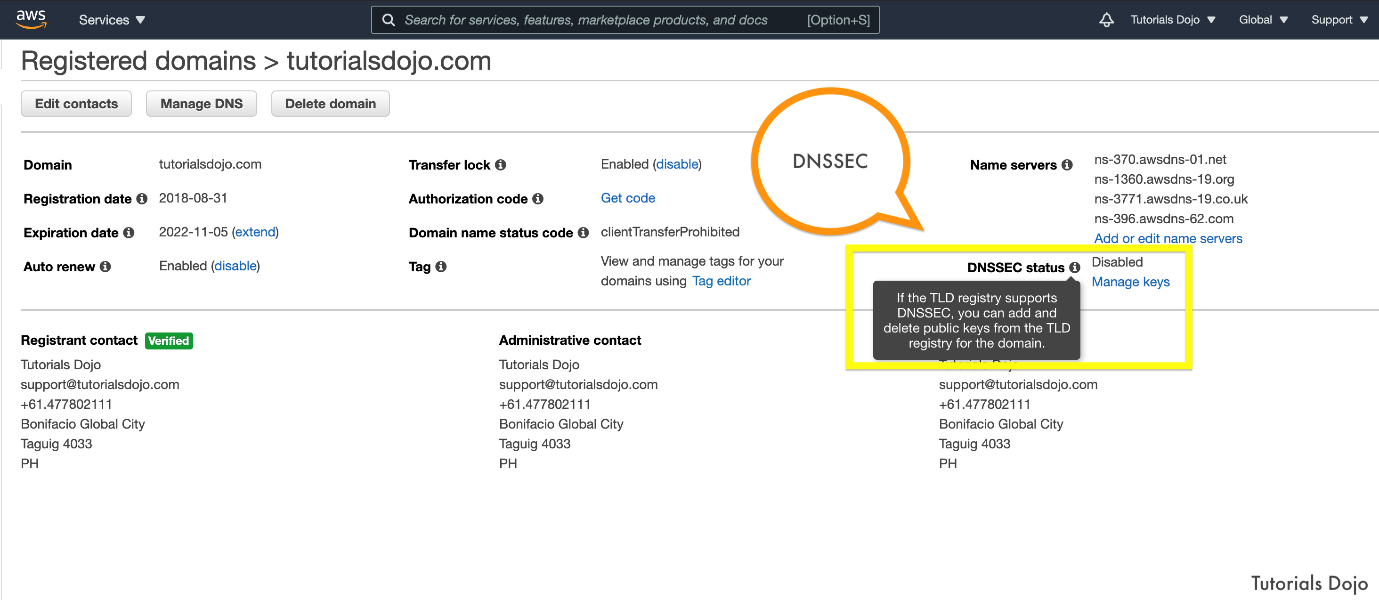
* ​

Register the domain name on Route 53. Since Route 53 only supports DNSSEC for registration, host the company DNS root servers on Amazon EC2 instances running the BIND service. Enable DNSSEC for DNS requests to ensure the replies have not been tampered with. Generate a valid certificate for the website domain name on AWS ACM and configure the Application Load Balancers HTTPS listener to use this TLS/SSL certificate. Use Server Name Identification and HTTP to HTTPS redirection on CloudFront.

**Explanation**

Amazon now allows you to enable **Domain Name System Security Extensions (DNSSEC)** signing for all existing and new public hosted zones, and enable DNSSEC validation for Amazon Route 53 Resolver. Amazon Route 53 DNSSEC provides data origin authentication and data integrity verification for DNS and can help customers meet compliance mandates, such as FedRAMP.

When you enable DNSSEC signing on a hosted zone, Route 53 cryptographically signs each record in that hosted zone. Route 53 manages the zone-signing key, and you can manage the key-signing key in AWS Key Management Service (AWS KMS). Amazon’s domain name registrar, Route 53 Domains, already supports DNSSEC, and customers can now register domains and host their DNS on Route 53 with DNSSEC signing enabled. When you enable DNSSEC validation on the Route 53 Resolver in your VPC, it ensures that DNS responses have not been tampered with in transit. This can prevent DNS Spoofing.



**AWS Certificate Manager** is a service that lets you easily provision, manage, and deploy public and private Secure Sockets Layer/Transport Layer Security (SSL/TLS) certificates for use with AWS services and your internal connected resources. SSL/TLS certificates are used to secure network communications and establish the identity of websites over the Internet as well as resources on private networks. AWS Certificate Manager removes the time-consuming manual process of purchasing, uploading, and renewing SSL/TLS certificates. Using a valid SSL Certificate for your application load balancer ensures that all requests are encrypted on transit as well as protection against SSL hijacking.

CloudFront supports Server Name Indication (SNI) for custom SSL certificates, along with the ability to take incoming HTTP requests and redirect them to secure HTTPS requests to ensure that clients are always directed to the secure version of your website.

Therefore, the correct answer is: **Register the domain name on Route 53 and enable DNSSEC validation for all public hosted zones to ensure that all DNS requests have not been tampered with during transit. Use AWS Certificate Manager (ACM) to generate a valid TLS/SSL certificate for the domain name. Configure the Application Load Balancer with an HTTPS listener to use the ACM TLS/SSL certificate. Use Server Name Identification and HTTP to HTTPS redirection on CloudFront.**

The option that says: **Register the domain name on Route 53. Use a third-party DNS provider that supports the import of the customer-managed keys for DNSSEC. Import a 2048-bit TLS/SSL certificate from a third-party certificate service to AWS Certificate Manager (ACM). Configure the Application Load Balancer with an HTTPS listener to use the imported TLS/SSL certificate. Use Server Name Identification and HTTP to HTTPS redirection on CloudFront** is incorrect. Although this is possible, you don’t have to rely on a third-party DNS provider as Route 53 supports DNSSEC signing. Also, ACM can secure a 2048-bit TLS/SSL Certificate for free so you don't have to buy certificates from other providers.

The option that says:**Use Route 53 for domain registration. Use a third-party DNS service that supports DNSSEC for DNS requests that use the customer-managed keys. Use AWS Certificate Manager (ACM) to generate a valid 2048-bit TLS/SSL certificate for the domain name and configure the Application Load Balancer HTTPS listener to use this TLS/SSL certificate. Use Server Name Identification and HTTP to HTTPS redirection on CloudFront**is incorrect. This is also possible, but you don't have to rely on a third-party DNS provider as Amazon Route 53 already supports DNSSEC signing.

The option that says:**Register the domain name on Route 53. Since Route 53 only supports DNSSEC for registration, host the company DNS root servers on Amazon EC2 instances running the BIND service. Enable DNSSEC for DNS requests to ensure the replies have not been tampered with. Generate a valid certificate for the website domain name on AWS ACM and configure the Application Load Balancers HTTPS listener to use this TLS/SSL certificate. Use Server Name Identification and HTTP to HTTPS redirection on CloudFront** is incorrect as this solution is no longer recommended. This setup was previously used as a workaround when DNSSEC signing was not supported natively yet in Amazon Route 53.

**References:**

<https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/domain-configure-dnssec.html>

<https://docs.aws.amazon.com/acm/latest/userguide/acm-services.html>

<https://aws.amazon.com/about-aws/whats-new/2020/12/announcing-amazon-route-53-support-dnssec/>

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/using-https-viewers-to-cloudfront.html>

**Check out these AWS Cheat Sheets:**

[https://tutorialsdojo.com/amazon-route-53/](https://tutorialsdojo.com/amazon-route-53/?src=udemy)

[https://tutorialsdojo.com/amazon-cloudfront/](https://tutorialsdojo.com/amazon-cloudfront/?src=udemy)

[https://tutorialsdojo.com/aws-certificate-manager/](https://tutorialsdojo.com/aws-certificate-manager/?src=udemy)

Bottom of Form

Top of Form

Question 37: Skipped

A retail company hosts its web application on an Auto Scaling group of Amazon EC2 instances deployed across multiple Availability Zones. The Auto Scaling group is configured to maintain a minimum EC2 cluster size and automatically replace unhealthy instances. The EC2 instances are behind an Application Load Balancer so that the load can be spread evenly on all instances. The application target group health check is configured with a fixed HTTP page that queries a dummy item on the database. The web application connects to a Multi-AZ Amazon RDS MySQL instance. A recent outage caused a major loss to the company's revenue. Upon investigation, it was found that the web server metrics are within the normal range but the database CPU usage is very high, causing the EC2 health checks to timeout. Failing the health checks, the Auto Scaling group continuously replaced the unhealthy instances thus causing the downtime.

Which of the following options should the Solution Architect implement to prevent this from happening again and allow the application to handle more traffic in the future? (Select TWO.)

* ​

Create an Amazon CloudWatch alarm to monitor the Amazon RDS MySQL instance if it has a high-load or in impaired status. Set the alarm action to recover the RDS instance. This will automatically reboot the database to reset the queries.

* ​

Change the target group health check to a simple HTML page instead of a page that queries the database. Create an Amazon Route 53 health check for the database dummy item web page to ensure that the application works as expected. Set up an Amazon CloudWatch alarm to send a notification to Admins when the health check fails.

**(Correct)**

* ​

Change the target group health check to use a TCP check on the EC2 instances instead of a page that queries the database. Create an Amazon Route 53 health check for the database dummy item web page to ensure that the application works as expected. Set up an Amazon CloudWatch alarm to send a notification to Admins when the health check fails.

* ​

Reduce the load on the database tier by creating an Amazon ElastiCache cluster to cache frequently requested database queries. Configure the application to use this cache when querying the RDS MySQL instance.

**(Correct)**

* ​

Reduce the load on the database tier by creating multiple read replicas for the Amazon RDS MySQL Multi-AZ cluster. Configure the web application to use the single reader endpoint of RDS for all read operations.

**Explanation**

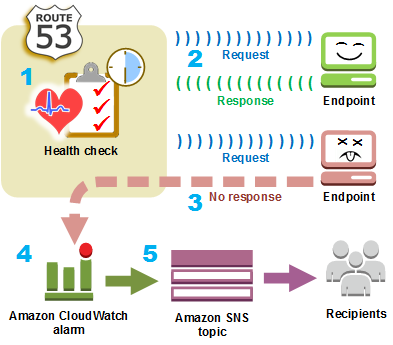
**Amazon Route 53 health checks** monitor the health and performance of your web applications, web servers, and other resources. Each health check that you create can monitor one of the following:

**The health of a specified resource, such as a web server** - You can configure a health check that monitors an endpoint that you specify either by IP address or by the domain name. At regular intervals that you specify, Route 53 submits automated requests over the Internet to your application. You can configure the health check to make requests similar to those that your users make, such as requesting a web page from a specific URL.

**The status of other health checks** - You can create a health check that monitors whether Route 53 considers other health checks healthy or unhealthy. One situation where this might be useful is when you have multiple resources that perform the same function, such as multiple web servers, and your chief concern is whether some minimum number of your resources are healthy.

**The status of an Amazon CloudWatch alarm** - You can create CloudWatch alarms that monitor the status of CloudWatch metrics, such as the number of throttled read events for an Amazon DynamoDB database or the number of Elastic Load Balancing hosts that are considered healthy.

After you create a health check, you can get the status of the health check, get notifications when the status changes, and configure DNS failover. To improve resiliency and availability, Route 53 doesn't wait for the CloudWatch alarm to go into the ALARM state. The status of a health check changes from healthy to unhealthy based on the data stream and on the criteria in the CloudWatch alarm.



Your **Application Load Balancer** periodically sends requests to its registered targets to test their status. These tests are called health checks. Each load balancer node routes requests only to the healthy targets in the enabled Availability Zones for the load balancer. Each load balancer node checks the health of each target, using the health check settings for the target groups with which the target is registered. After your target is registered, it must pass one health check to be considered healthy. After each health check is completed, the load balancer node closes the connection that was established for the health check. If a target group contains only unhealthy registered targets, the load balancer nodes route requests across its unhealthy targets.

Each health check will be executed at configured intervals to all the EC2 instances so if the health check query page involves a database query, there will be several simultaneous queries to the database. This can increase the load of your database tier if there are many EC2 instances and the health check interval period is very quick.

**Amazon ElastiCache** is a web service that makes it easy to set up, manage, and scale a distributed in-memory data store or cache environment in the cloud. It provides a high-performance, scalable, and cost-effective caching solution. At the same time, it helps remove the complexity associated with deploying and managing a distributed cache environment.

ElastiCache for Memcached has multiple features to enhance reliability for critical production deployments:

- Automatic detection and recovery from cache node failures.

- Automatic discovery of nodes within a cluster enabled for automatic discovery, so that no changes need to be made to your application when you add or remove nodes.

- Flexible Availability Zone placement of nodes and clusters.

- Integration with other AWS services such as Amazon EC2, Amazon CloudWatch, AWS CloudTrail, and Amazon SNS to provide a secure, high-performance, managed in-memory caching solution.

The option that says: **Change the target group health check to a simple HTML page instead of a page that queries the database. Create an Amazon Route 53 health check for the database dummy item web page to ensure that the application works as expected. Set up an Amazon CloudWatch alarm to send a notification to Admins when the health check fails**is correct. Changing the target group health check to a simple HTML page will reduce the queries to the database tier. The Route 53 health check can act as the “external” check on a specific page that queries the database to ensure that the application is working as expected. The Route 53 health check has an overall lower request count compared to using the target group health check.

The option that says: **Reduce the load on the database tier by creating an Amazon ElastiCache cluster to cache frequently requested database queries. Configure the application to use this cache when querying the RDS MySQL instance**is correct. Since this is a retail web application, most of the queries will be read-intensive as customers are searching for products. ElastiCache is effective at caching frequent requests which overall improves the application response time and reduces database queries.

The option that says: **Reduce the load on the database tier by creating multiple read replicas for the Amazon RDS MySQL Multi-AZ cluster. Configure the web application to use the single reader endpoint of RDS for all read operations**is incorrect. Creating read replicas is recommended to increase the read performance of an RDS cluster. However, the Amazon RDS MySQL does not have a single reader endpoint for read replicas. You must use Amazon Aurora for MySQL to support this.

The option that says: **Change the target group health check to use a TCP check on the EC2 instances instead of a page that queries the database. Create an Amazon Route 53 health check for the database dummy item web page to ensure that the application works as expected. Set up an Amazon CloudWatch alarm to send a notification to Admins when the health check fails** is incorrect. An Application Load Balancer does not support a TCP health check. ALB only supports HTTP and HTTPS target health checks.

The option that says: **Create an Amazon CloudWatch alarm to monitor the Amazon RDS MySQL instance if it has a high-load or in impaired status. Set the alarm action to recover the RDS instance. This will automatically reboot the database to reset the queries**is incorrect. Recovering the database instance results in downtime. If you have the Multi-AZ enabled, the standby database will shoulder all the load causing it to crash too. It is better to scale the database by creating read replicas or adding an ElastiCache cluster in front of it.

**References:**

<https://docs.aws.amazon.com/elasticloadbalancing/latest/application/target-group-health-checks.html>

<https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/dns-failover.html>

<https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/health-checks-types.html>

<https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/WhatIs.html>

**Check out the Amazon ElastiCache and Amazon RDS Cheat Sheets:**

[https://tutorialsdojo.com/amazon-elasticache/](https://tutorialsdojo.com/amazon-elasticache/?src=udemy)

[https://tutorialsdojo.com/amazon-relational-database-service-amazon-rds/](https://tutorialsdojo.com/amazon-relational-database-service-amazon-rds/?src=udemy)

Bottom of Form

Top of Form

Question 38: Skipped

A web application is composed of an Application Load Balancer and EC2 instances across three Availability Zones. During peak load, the web servers operate at 95% utilization. The system is set up to use Reserved Instances to handle steady state load and On-Demand Instances to handle the peak load. Your manager instructed you to review the current architecture and do the necessary changes to improve the system.

Which of the following provides the most cost-effective architecture to allow the application to recover quickly in the event that an Availability Zone is unavailable during peak load?

* ​

Launch an Auto Scaling group of Reserved instances on each AZ to handle the peak load. Retain the current set up for handling the steady state load.

* ​

Launch a Spot Fleet using a diversified allocation strategy, with Auto Scaling enabled on each AZ to handle the peak load instead of On-Demand instances. Retain the current set up for handling the steady state load.

**(Correct)**

* ​

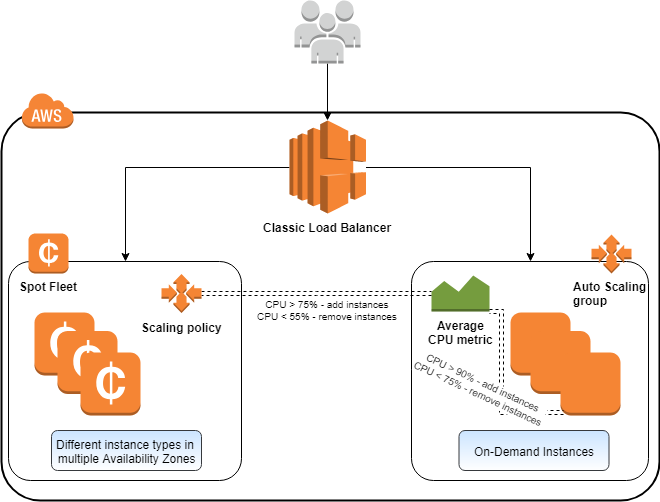
Use a combination of Spot and On-Demand instances on each AZ to handle both the steady state and peak load.

* ​

Use a combination of Reserved and On-Demand instances on each AZ to handle both the steady state and peak load.

**Explanation**

The scenario requires a cost-effective architecture to allow the application to recover quickly, hence, using an Auto Scaling group is a must to handle the peak load and improve both the availability and scalability of the application. Since the options that say: ***Use a combination of Spot and On-Demand instances on each AZ to handle both the steady state and peak load*** and ***use a combination of Reserved and On-Demand instances on each AZ to handle both the steady state and peak load*** did not mention the use of Auto Scaling Groups, these options are incorrect.



Reserved Instances cost more than Spot instances so it is more suitable to use the latter to handle the peak load. That is why ***launching an Auto Scaling group of Reserved instances on each AZ to handle the peak load and retaining the current set up for handling the steady state load*** is wrong even though it uses Auto Scaling.

Setting up a diversified allocation strategy for your Spot Fleet is a best practice to increase the chances that a spot request can be fulfilled by EC2 capacity in the event of an outage in one of the Availability Zones. You can include each AZ available to you in the launch specification. And instead of using the same subnet each time, use three unique subnets (each mapping to a different AZ).

**Reference:**

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/spot-fleet-examples.html#fleet-config5>

<https://d1.awsstatic.com/whitepapers/total-cost-of-operation-benefits-using-aws.pdf#page=11>

<https://github.com/awsdocs/amazon-ec2-user-guide/pull/56>

**Check out this AWS Billing and Cost Management Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-billing-and-cost-management/>

Bottom of Form

Top of Form

Question 39: Skipped

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.)

* ​

Build a custom OpenID Connect-compatible solution for the user authentication functionality. Use Amazon Cognito Identity Pools for authorizing access to AWS resources.

**(Correct)**

* ​

Build a custom SAML-compatible solution for user authentication. Leverage AWS Single Sign-On (SSO) for authorizing access to AWS resources.

* ​

Build a custom OpenID Connect-compatible solution in combination with AWS Single Sign-On (SSO) 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.

**(Correct)**

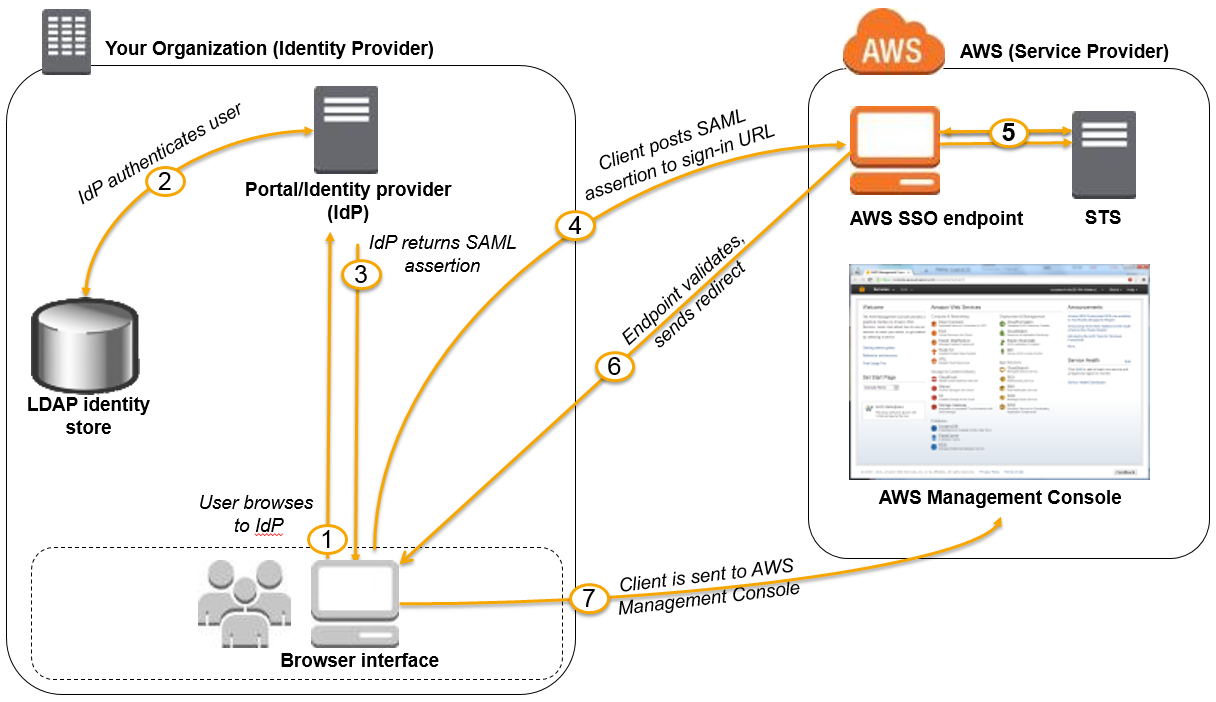
* ​

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.

**Explanation**

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 you 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:

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.

The portal verifies the user's identity in your organization.

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 browser.

The client browser is redirected to the AWS single sign-on endpoint and posts the SAML assertion.

The endpoint requests temporary security credentials on behalf of the user and creates a console sign-in URL that uses those credentials.

AWS sends the sign-in URL back to the client as a redirect.

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 user name and password, or through a third party such as Facebook, Amazon, Google, or Apple. The two main components of Amazon Cognito are user pools and identity pools. User pools are user directories that provide sign-up and sign-in options for your app users. Identity pools enable you to grant your users access to other AWS services. You can use identity pools and user pools separately or together.

**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.

The option that says: **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** is correct. The requirement is to use a custom-built solution for user authentication and this can use the company LDAP system for authentication. The SAML assertion is also needed to get authorization tokens from the IAM identity provider that will grant IAM roles to users that wish to access AWS resources.

The option that says: **Build a custom OpenID Connect-compatible solution for the user authentication functionality. Use Amazon Cognito Identity Pools for authorizing access to AWS resources**is correct. The custom OpenID Connect-compatible solution will allow users to log in from their mobile application much like a single sign-on functionality. Amazon Cognito Identity Pool will provide temporary tokens to federated users for accessing AWS resources.

The option that says: **Build a custom SAML-compatible solution for user authentication. Leverage AWS Single Sign-On (SSO) for authorizing access to AWS resources** is incorrect. The requirement is to grant federated access from the mobile application. AWS SSO supports single sign-on to business applications through web browsers only.

The option that says: **Build a custom LDAP connector using Amazon API Gateway with AWS Lambda function to 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** is incorrect. It is not recommended to store authorization tokens permanently on DynamoDB tables. These tokens should be generated upon user authentication and then temporarily saved on a DynamoDB for a fixed session length.

The option that says: **Build a custom OpenID Connect-compatible solution in combination with AWS Single Sign-On (SSO) to create authentication and authorization functionality for the application** is incorrect. AWS SSO supports only SAML 2.0–based applications so an OpenID Connect-compatible solution will not work for this scenario.

**References:**

<https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_enable-console-saml.html>

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

<https://docs.aws.amazon.com/cognito/latest/developerguide/open-id.html>

<https://aws.amazon.com/single-sign-on/faqs/>

**Check out these Amazon Cognito Cheat Sheets:**

[https://tutorialsdojo.com/amazon-cognito/](https://tutorialsdojo.com/amazon-cognito/?src=udemy)

[https://tutorialsdojo.com/amazon-cognito-user-pools-and-identity-pools-explained/](https://tutorialsdojo.com/amazon-cognito-user-pools-and-identity-pools-explained/?src=udemy)

Bottom of Form

Top of Form

Question 40: Skipped

An innovative Business Process Outsourcing (BPO) startup is planning to launch a scalable and cost-effective call center system using AWS. The system should be able to receive inbound calls from thousands of customers and generate user contact flows. Callers must have the capability to perform basic tasks such as changing their password or checking their balance without them having to speak to a call center agent. It should also have advanced deep learning functionalities such as automatic speech recognition (ASR) to achieve highly engaging user experiences and lifelike conversational interactions. A feature that allows the solution to query other business applications and send relevant data back to callers must also be implemented.

Which of the following is the MOST suitable solution that the Solutions Architect should implement?

* ​

Set up a cloud-based contact center using the AWS Ground Station service. Create a conversational chatbot using Amazon Alexa for Business with automatic speech recognition and natural language understanding to recognize the intent of the caller then integrate it with AWS Ground Station. Connect the solution to various business applications and other internal systems using AWS Lambda functions.

* ​

Set up a cloud-based contact center using the AWS Elemental MediaConnect service. Create a conversational chatbot using Amazon Polly with automatic speech recognition and natural language understanding to recognize the intent of the caller then integrate it with AWS Elemental MediaConnect. Connect the solution to various business applications and other internal systems using AWS Lambda functions.

* ​

Set up a cloud-based contact center using the Amazon Direct Connect service. Create a conversational chatbot using Amazon Rekognition with automatic speech recognition and natural language understanding to recognize the intent of the caller then integrate it with Amazon Direct Connect. Connect the solution to various business applications and other internal systems using AWS Lambda functions.

* ​

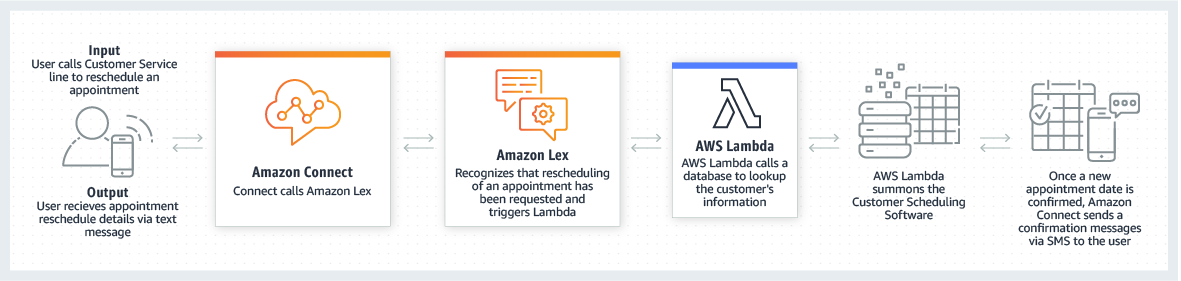
Set up a cloud-based contact center using the Amazon Connect service. Create a conversational chatbot using Amazon Lex with automatic speech recognition and natural language understanding to recognize the intent of the caller then integrate it with Amazon Connect. Connect the solution to various business applications and other internal systems using AWS Lambda functions.

**(Correct)**

**Explanation**

**Amazon Connect** provides a seamless omnichannel experience through a single unified contact center for voice and chat. Contact center agents and managers don’t have to learn multiple tools, because Amazon Connect has the same contact routing, queuing, analytics, and management tools in a single UI across voice, web chat, and mobile chat.

**Amazon Lex** is a service for building conversational interfaces into any application using voice and text. Amazon Lex provides the advanced deep learning functionalities of automatic speech recognition (ASR) for converting speech to text, and natural language understanding (NLU) to recognize the intent of the text, to enable you to build applications with highly engaging user experiences and lifelike conversational interactions. With Amazon Lex, the same deep learning technologies that power Amazon Alexa are now available to any developer, enabling you to quickly and easily build sophisticated, natural language, conversational bots ("chatbots").



Contact flows define the experience your customers have when they interact with your contact center. These are similar in concept to Interactive Voice Response (IVR). Contact flows are comprised of blocks, with each block defining a step or interaction in your contact center. For example, there are blocks to play a prompt, get input from a customer, branch based on customer input, or invoke an AWS Lambda function or and Amazon Lex bot.

By using an Amazon Lex chatbot in your Amazon Connect call center, callers can perform tasks such as changing a password, requesting a balance on an account, or scheduling an appointment, without needing to speak to an agent. These chatbots use automatic speech recognition and natural language understanding to recognize the intent of the caller. They are able to recognize human speech at an optimal (8 kHz) telephony audio sampling rate, and understand the caller’s intent without requiring the caller to speak in specific phrases. Amazon Lex uses AWS Lambda functions to query your business applications, provide information back to callers, and make updates as requested. Amazon Lex chatbots also maintain context and manage the dialogue, dynamically adjusting responses based on the conversation.

Hence, the correct answer is: ***Set up a cloud-based contact center using the Amazon Connect service. Create a conversational chatbot using Amazon Lex with automatic speech recognition and natural language understanding to recognize the intent of the caller then integrate it with Amazon Connect. Connect the solution to various business applications and other internal systems using AWS Lambda functions.***

The option that says: ***Set up a cloud-based contact center using the AWS Ground Station service. Create a conversational chatbot using Amazon Alexa for Business with automatic speech recognition and natural language understanding to recognize the intent of the caller then integrate it with AWS Ground Station. Connect the solution to various business applications and other internal systems using AWS Lambda functions***is incorrect because AWS Ground Station is a fully managed service that lets you control satellite communications, process data, and scale your operations without having to worry about building or managing your own ground station infrastructure. This service is not suitable as a cloud-based contact center. Moreover, Alexa for Business simply empowers companies to use Alexa devices as their intelligent assistant to be more productive in meeting rooms, at their desks, and even with the Alexa devices they already use at home or on the go. You should use a different machine learning service, such as Amazon Lex, to create a conversational chatbot.

The option that says: ***Set up a cloud-based contact center using the AWS Elemental MediaConnect service. Create a conversational chatbot using Amazon Polly with automatic speech recognition and natural language understanding to recognize the intent of the caller then integrate it with AWS Elemental MediaConnect. Connect the solution to various business applications and other internal systems using AWS Lambda functions***is incorrect because AWS Elemental MediaConnect is just a high-quality transport service for live video. You have to use Amazon Connect instead to set up your cloud-based contact center. And although Amazon Polly is a machine learning service, it is quite limited as it just turns text into lifelike speech that allows you to create applications that talk and build entirely new categories of speech-enabled products. A more suitable service to use here is Amazon Lex.

The option that says: ***Set up a cloud-based contact center using the Amazon Direct Connect service. Create a conversational chatbot using Amazon Rekognition with automatic speech recognition and natural language understanding to recognize the intent of the caller then integrate it with Amazon Direct Connect. Connect the solution to various business applications and other internal systems using AWS Lambda functions***is incorrect because Amazon Direct Connect is simply a cloud service solution that makes it easy to establish a dedicated network connection from your premises to AWS. The most suitable service in setting up a cloud-based contact center is Amazon Connect. Amazon Rekognition only automates your image and video analysis with machine learning. It doesn't have automatic speech recognition (ASR) for converting speech to text nor natural language understanding (NLU). You have to use Amazon Lex instead.

**References:**

<https://aws.amazon.com/connect/>

<https://aws.amazon.com/lex/>

Bottom of Form

Top of Form

Question 41: Skipped

**Last year, the big four banks in your country have collaborated to create a simple-to-use, mobile payment app that enables the users to easily transfer money and pay bills without the hassle of logging in to their online banking, entering the account details of the other party and spending time going through other security verification processes. With their mobile payment app, anyone can easily pay another person, split the bill with their friends or pay for their coffee in an instant with just a few taps in the app. The payment app is available on both Android and iOS devices, including a web portal that is deployed in AWS using OpsWorks Stacks and EC2 instances. It was a big success with over 5 millions users nationwide and has over 100 transactions every hour. After one year, a new feature that will enable the users to store their credit card information in the app is ready to be added to the existing web portal. However, due to PCI-DSS compliance, the new version of the APIs and web portal cannot be deployed to the existing application stack. As the Solutions Architect of this project, how would you deploy the new web portal for the mobile app without having any impact to your 5 million users?**

* ​

Create a new stack that contains the latest version of the web portal. Using Route 53 service, direct all the incoming traffic to the new stack at once so that all the customers get to access new features.

* ​

Deploy a new OpsWorks stack that contains a new layer with the latest web portal version. Shift traffic between existing stack and new stack, running different versions of the web portal using Blue/Green deployment strategy by using Route53. Route only a small portion of incoming production traffic to use the new application stack while maintaining the old application stack. Check the features of the new portal; once it's 100% validated, slowly increase incoming production traffic to the new stack. If there are issues on the new stack, change Route53 to revert to old stack.

**(Correct)**

* ​

Forcibly upgrade the existing application stack in Production to be PCI-DSS compliant. Once done, deploy the new version of the web portal on the existing application stack.

* ​

Deploy the new web portal using a Blue/Green deployment strategy with AWS CodeDeploy and Lambda in which the green environment represents the current web portal version serving production traffic while the blue environment is staged in running a different version of the web portal.

**Explanation**

Blue/green deployments provide near zero-downtime release and rollback capabilities. The fundamental idea behind blue/green deployment is to shift traffic between two identical environments that are running different versions of your application. The blue environment represents the current application version serving production traffic. In parallel, the green environment is staged running a different version of your application. After the green environment is ready and tested, production traffic is redirected from blue to green. If any problems are identified, you can roll back by reverting traffic back to the blue environment.

Hence, the correct answer in this scenario is the option that says: ***Deploy a new OpsWorks stack that contains a new layer with the latest web portal version. Shift traffic between existing stack and new stack, running different versions of the web portal using Blue/Green deployment strategy by using Route53. Route only a small portion of incoming production traffic to use the new application stack while maintaining the old application stack. Check the features of the new portal; once it's 100% validated, slowly increase incoming production traffic to the new stack. If there are issues on the new stack, change Route53 to revert to old stack.***

The option that says: ***Forcibly upgrade the existing application stack in Production to be PCI-DSS compliant. Once done, deploy the new version of the web portal on the existing application stack*** is incorrect because if you forcibly deploy the new web portal to the existing application stack and there is an issue on the deployment, then there would be some inevitable downtime in order for you to fix the system and revert back to the original version. It is better to use blue/green deployments instead.

The option that says:***Create a new stack that contains the latest version of the web portal. Using Route 53 service, direct all the incoming traffic to the new stack at once so that all the customers get to access new features*** is incorrect because although the current application stack can still be used if something goes wrong, the risk of service disruption is quite high when you direct all incoming traffic to the new portal. If something did go wrong, then there would be a downtime in order for you to switch back to the old stack.

The option that says: ***Deploy the new web portal using a Blue/Green deployment strategy with AWS CodeDeploy and Lambda in which the green environment represents the current web portal version serving production traffic while the blue environment is staged in running a different version of the web portal***is incorrect because although it mentioned Blue/Green deployment, the application stack doesn't mention about serverless computing at all. Hence, Lambda is irrelevant in this situation and in addition, the description of the blue and green environments are actually switched. The blue environment represents the current application while the green represents the new one.

**Reference:**

<https://d0.awsstatic.com/whitepapers/AWS_Blue_Green_Deployments.pdf>

Bottom of Form

Top of Form

Question 42: Skipped

You are working as a Senior Solutions Architect for a leading accounting firm which conducts monthly performance checks of their Windows and Linux EC2 instances. They have more than 200 On-Demand EC2 instances running in their production environment and you were instructed to ensure that each instance has a logging feature that collects various system details such as memory usage, disk space, and other metrics. The system logs will be analyzed using AWS Analytics tools and the results will be stored to an S3 bucket.

Which of the following is the most efficient way to collect and analyze logs from the instances with minimal effort?

* ​

Set up and install AWS Inspector Agent on each On-Demand EC2 instance which will collect and push data to CloudWatch Logs periodically. Set up a CloudWatch dashboard to properly analyze the log data of all instances.

* ​

Install AWS SDK on each On-Demand EC2 instance and create a custom daemon script that would collect and push data to CloudWatch Logs periodically. Enable CloudWatch detailed monitoring and use CloudWatch Logs Insights to analyze the log data of all instances.

* ​

Set up and configure a unified CloudWatch Logs agent in each On-Demand EC2 instance which will automatically collect and push data to CloudWatch Logs. Analyze the log data with CloudWatch Logs Insights.

**(Correct)**

* ​

Set up and install the AWS Systems Manager Agent (SSM Agent) on each On-Demand EC2 instance which will automatically collect and push data to CloudWatch Logs. Analyze the log data with CloudWatch Logs Insights.

**Explanation**

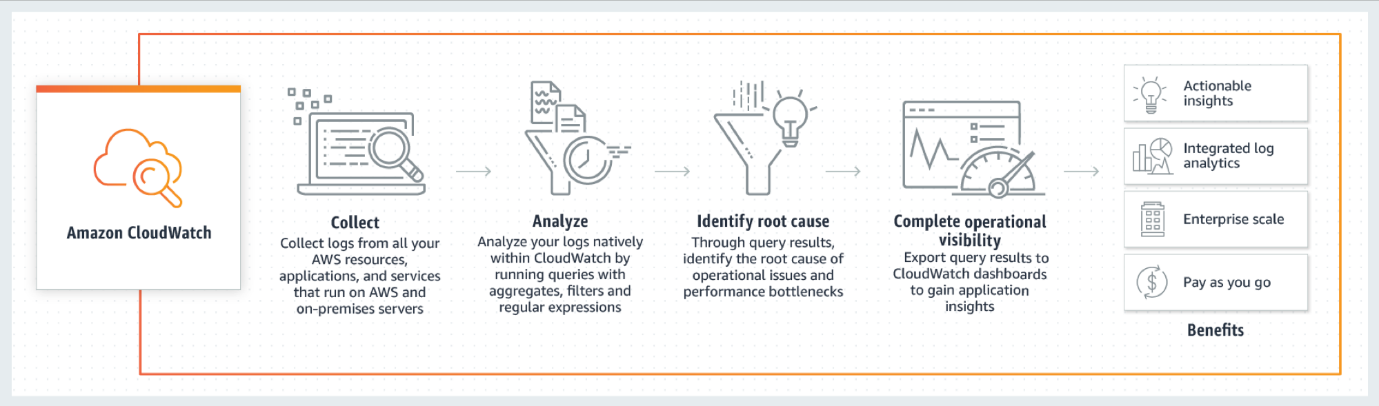
To collect logs from your Amazon EC2 instances and on-premises servers into CloudWatch Logs, AWS offers both a new unified CloudWatch agent, and an older CloudWatch Logs agent. It is recommended to use the unified CloudWatch agent which has the following advantages:

- You can collect both logs and advanced metrics with the installation and configuration of just one agent.

- The unified agent enables the collection of logs from servers running Windows Server.

- If you are using the agent to collect CloudWatch metrics, the unified agent also enables the collection of additional system metrics, for in-guest visibility.

- The unified agent provides better performance.



CloudWatch Logs Insights enables you to interactively search and analyze your log data in Amazon CloudWatch Logs. You can perform queries to help you quickly and effectively respond to operational issues. If an issue occurs, you can use CloudWatch Logs Insights to identify potential causes and validate deployed fixes.

CloudWatch Logs Insights includes a purpose-built query language with a few simple but powerful commands. CloudWatch Logs Insights provides sample queries, command descriptions, query autocompletion, and log field discovery to help you get started quickly. Sample queries are included for several types of AWS service logs. Hence, the correct answer is to ***set up and configure a unified CloudWatch Logs agent in each On-Demand EC2 instance which will automatically collect and push data to CloudWatch Logs, then analyze the log data with CloudWatch Logs Insights.***

***Installing AWS SDK on each On-Demand EC2 instance and creating a custom daemon script that would collect and push data to CloudWatch Logs periodically, then enabling CloudWatch detailed monitoring and using CloudWatch Logs Insights to analyze the log data of all instances*** is incorrect. Although this is a valid solution, this entails a lot of effort to implement as you have to allocate time to install the AWS SDK to each instance and develop a custom monitoring solution. Remember that the question is specifically looking for a solution that can be implemented with minimal effort. In addition, it is unnecessary and not cost-efficient to enable detailed monitoring in CloudWatch in order to meet the requirements of this scenario since this can be done using CloudWatch Logs.

***Setting up and installing the AWS Systems Manager Agent (SSM Agent) on each On-Demand EC2 instance which will automatically collect and push data to CloudWatch Logs, then analyzing the log data with CloudWatch Logs Insights*** is incorrect as although this is also a valid solution, it is more efficient to use CloudWatch agent than an SSM agent. Manually connecting to an instance to view log files and troubleshoot an issue with SSM Agent is time-consuming hence, for more efficient instance monitoring, you can use the CloudWatch Agent instead to send the log data to Amazon CloudWatch Logs.

***Setting up and installing AWS Inspector Agent on each On-Demand EC2 instance which will collect and push data to CloudWatch Logs periodically, then setting up a CloudWatch dashboard to properly analyze the log data of all instances*** is incorrect because AWS Inspector is simply a security assessments service which only helps you in checking for unintended network accessibility of your EC2 instances and for vulnerabilities on those EC2 instances. Furthermore, setting up an Amazon CloudWatch dashboard is not suitable since it's primarily used for scenarios where you have to monitor your resources in a single view, even those resources that are spread across different AWS Regions. It is better to use CloudWatch Logs Insights instead since it enables you to interactively search and analyze your log data.

**References:**

<https://docs.aws.amazon.com/AmazonCloudWatch/latest/logs/WhatIsCloudWatchLogs.html>

<https://docs.aws.amazon.com/systems-manager/latest/userguide/monitoring-ssm-agent.html>

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

**Check out this Amazon CloudWatch Cheat Sheet:**

[https://tutorialsdojo.com/amazon-cloudwatch/](https://tutorialsdojo.com/amazon-cloudwatch/?src=udemy)

**Additional learning material:** CloudWatch Logs Insights Customer Use Case

<https://youtu.be/RnN1o4Zdego>

**CloudWatch Agent vs SSM Agent vs Custom Daemon Scripts:**

[https://tutorialsdojo.com/cloudwatch-agent-vs-ssm-agent-vs-custom-daemon-scripts/](https://tutorialsdojo.com/cloudwatch-agent-vs-ssm-agent-vs-custom-daemon-scripts/?src=udemy)

Bottom of Form

Top of Form

Question 43: Skipped

**You are working for a San Francisco-based company which provides digital transaction management services for facilitating electronic exchanges of contracts and signed documents. Using their online system, businessmen and contractors can digitally sign contracts anywhere and anytime, removing the hassle of signing them in paper and in person. They are using AWS to host their multi-tier online portal in which the application tier is using a NGINX server hosted on an extra large EC2 instance; the database tier is using an Oracle database which is regularly backed up to an S3 bucket using a custom backup utility and lastly, its static content is kept on a 512GB stored volume in AWS Storage Gateway which is attached to the application server via the iSCSI interface. In this scenario, which AWS based disaster recovery strategy will give you the best RTO?**

* ​

1. Deploy the Oracle database on RDS.

2. Deploy the NGINX app server on an EC2 instance.

3. Restore the Recovery Manager (RMAN) Oracle backups from Amazon Glacier.

4. Generate an EBS volume of static content from the Storage Gateway and attach it to the NGINX EC2 server.

* ​

1. Deploy the Oracle database and the NGINX app server on EC2.

2. Restore the Recovery Manager (RMAN) Oracle backups from an S3 bucket.

3. Restore the static content from an AWS Storage Gateway-VTL running on Amazon EC2.

* ​

1. Deploy the Oracle database and the NGINX app server on an EC2 instance.

2. Restore the Recovery Manager (RMAN) Oracle backups from an Amazon S3 bucket.

3. Generate an EBS volume of static content from the Storage Gateway and attach it to the NGINX EC2 server.

**(Correct)**

* ​

1. Deploy the Oracle database and the NGINX app server to an EC2 instance.

2. Restore the Recovery Manager (RMAN) Oracle backups from an S3 bucket.

3. Restore the static content by attaching an AWS Storage Gateway running on Amazon EC2 as an iSCSI volume to the NGINX EC2 server.

**Explanation**

Since this uses a Volume Storage Gateway, you have to generate an EBS Volume to restore the data. Hence, the following option is correct:

***1. Deploy the Oracle database and the NGINX app server on an EC2 instance.***

***2. Restore the Recovery Manager (RMAN) Oracle backups from an Amazon S3 bucket.***

***3. Generate an EBS volume of static content from the Storage Gateway and attach it to the NGINX EC2 server.***

The following sets of options are incorrect because you cannot directly restore your data to EC2 instance:

***1. Deploy the Oracle database and the NGINX app server on EC2.***

***2. Restore the Recovery Manager (RMAN) Oracle backups from an S3 bucket.***

***3. Restore the static content from an AWS Storage Gateway-VTL running on Amazon EC2.***

***--***

***1. Deploy the Oracle database and the NGINX app server to an EC2 instance.***

***2. Restore the Recovery Manager (RMAN) Oracle backups from an S3 bucket.***

***3. Restore the static content by attaching an AWS Storage Gateway running on Amazon EC2 as an iSCSI volume to the NGINX EC2 server.***

Remember that RDS does not support RMAN backup utility, so the following option is incorrect as you cannot use RDS:

***1. Deploy the Oracle database on RDS.***

***2. Deploy the NGINX app server on an EC2 instance.***

***3. Restore the Recovery Manager (RMAN) Oracle backups from Amazon Glacier.***

***4. Generate an EBS volume of static content from the Storage Gateway and attach it to the NGINX EC2 server.***

While Amazon RDS does not use RMAN for backups, you can use the package to execute RMAN validation commands against the database, control file, SPFILE, tablespaces, or data files.

The volume gateway provides block storage to your applications using the iSCSI protocol. Data on the volumes are stored in Amazon S3. To access your iSCSI volumes in AWS, you can take EBS snapshots which can be used to create EBS volumes.

**Reference:**

<https://aws.amazon.com/storagegateway/faqs/>

**Check out this AWS Storage Gateway Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-storage-gateway/>

**Tutorials Dojo's AWS Certified Solutions Architect Professional Exam Study Guide:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-solutions-architect-professional/>

Bottom of Form

Top of Form

Question 44: Skipped

A government agency has multiple VPCs in various AWS regions across the United States that need to be linked up to an on-premises central office network in Washington, D.C. The central office requires inter-region VPC access over a private network that is dedicated to each region for enhanced security and more predictable data transfer performance. Your team is tasked to quickly build this network mesh and to minimize the management overhead to maintain these connections.

Which of the following options is the most secure, highly available, and durable solution that you should use to set up this kind of interconnectivity?

* ​

Create a link aggregation group (LAG) in the central office network to aggregate multiple connections at a single AWS Direct Connect endpoint in order to treat them as a single, managed connection. Use AWS Direct Connect Gateway to achieve inter-region VPC access to all of your AWS resources. Create a virtual private gateway in each VPC and then create a public virtual interface for each AWS Direct Connect connection to the Direct Connect Gateway.

* ​

Enable inter-region VPC peering which allows peering relationships to be established between VPCs across different AWS regions. This will ensure that the traffic will always stay on the global AWS backbone and will never traverse the public Internet.

* ​

Implement a hub-and-spoke network topology in each region that routes all traffic through a network transit center using AWS Transit Gateway. Route traffic between VPCs and the on-premise network over AWS Site-to-Site VPN.

* ​

Utilize AWS Direct Connect Gateway for inter-region VPC access. Create a virtual private gateway in each VPC, then create a private virtual interface for each AWS Direct Connect connection to the Direct Connect gateway.

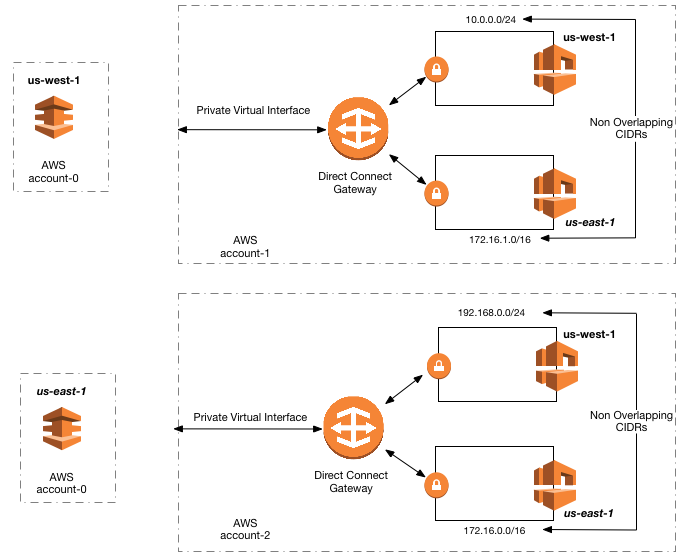
**(Correct)**

**Explanation**

AWS Direct Connect is a cloud service solution that makes it easy to establish a dedicated network connection from your premises to AWS to achieve higher privacy benefits, additional data transfer bandwidth, and more predictable data transfer performance. Using AWS Direct Connect, you can establish private connectivity between AWS and your datacenter, office, or colocation environment, which in many cases can reduce your network costs, increase bandwidth throughput, and provide a more consistent network experience than Internet-based connections.

Using industry standard 802.1q VLANs, this dedicated connection can be partitioned into multiple virtual interfaces. Virtual interfaces can be reconfigured at any time to meet your changing needs. You can use an ***AWS Direct Connect gateway***to connect your AWS Direct Connect connection over a private virtual interface to one or more VPCs in your account that are located in the same or different Regions. You associate a Direct Connect gateway with the virtual private gateway for the VPC. Then, create a private virtual interface for your AWS Direct Connect connection to the Direct Connect gateway. You can attach multiple private virtual interfaces to your Direct Connect gateway.

With Direct Connect Gateway, you no longer need to establish multiple BGP sessions for each VPC; this reduces your administrative workload as well as the load on your network devices.



The option that says: ***Create a link aggregation group (LAG) in the central office network to aggregate multiple connections at a single AWS Direct Connect endpoint in order to treat them as a single, managed connection. Use AWS Direct Connect Gateway to achieve inter-region VPC access to all of your AWS resources. Create a virtual private gateway in each VPC and then create a public virtual interface for each AWS Direct Connect connection to the Direct Connect Gateway*** is incorrect because you can only create a **private** virtual interface to a Direct Connect gateway and not a **public** virtual interface. Using a link aggregation group (LAG) is also irrelevant in this scenario because it is just a logical interface that uses the Link Aggregation Control Protocol (LACP) to aggregate multiple connections at a single AWS Direct Connect endpoint, allowing you to treat them as a single, managed connection.

The option that says:***Implement a hub-and-spoke network topology in each region that routes all traffic through a network transit center using AWS Transit Gateway. Route traffic between VPCs and the on-premise network over AWS Site-to-Site VPN*** is incorrect since the scenario requires a service that can provide a dedicated network between the VPCs and the on-premise network, as well as enhanced privacy and predictable data transfer performance. Simply using AWS Transit Gateway will not fulfill the conditions above. This option is best suited to customers who want to leverage AWS-provided, automated high availability network connectivity features and also optimize their investments in third-party product licensing such as VPN software.

The option that says: ***Enable inter-region VPC peering which allows peering relationships to be established between VPCs across different AWS regions. This will ensure that the traffic will always stay on the global AWS backbone and will never traverse the public Internet*** is incorrect because this solution would require a lot of manual setup and management overhead to successfully build a functional, error-free inter-region VPC network compared with just using a Direct Connect Gateway. Although the Inter-Region VPC Peering provides a cost-effective way to share resources between regions or replicate data for geographic redundancy, its connections are not dedicated and highly available.

**References:**

<https://aws.amazon.com/blogs/aws/new-aws-direct-connect-gateway-inter-region-vpc-access/>

<https://docs.aws.amazon.com/directconnect/latest/UserGuide/direct-connect-gateways.html>

<https://aws.amazon.com/answers/networking/aws-multiple-region-multi-vpc-connectivity/>

Bottom of Form

Top of Form

Question 45: Skipped

A stocks brokerage firm hosts its legacy application on Amazon EC2 in a private subnet of its Amazon VPC. The application is accessed by the employees from their corporate laptops through a proprietary desktop program. The company network is peered with the AWS Direct Connect (DX) connection to provide a fast and reliable connection to the private EC2 instances inside the VPC. To comply with the strict security requirements of financial institutions, the firm is required to encrypt its network traffic that flows from the employees' laptops to the resources inside the VPC.

Which of the following solution will comply with this requirement while maintaining the consistent network performance of Direct Connect?

* ​

Using the current Direct Connect connection, create a new public virtual interface and input the network prefixes that you want to advertise. Create a new site-to-site VPN connection to the VPC with the BGP protocol using the DX connection. Configure the employees’ laptops to connect to this VPN.

**(Correct)**

* ​

Using the current Direct Connect connection, create a new private virtual interface and input the network prefixes that you want to advertise. Create a new site-to-site VPN connection to the VPC with the BGP protocol using the DX connection. Configure the employees’ laptops to connect to this VPN.

* ​

Using the current Direct Connect connection, create a new private virtual interface and input the network prefixes that you want to advertise. Create a new site-to-site VPN connection to the VPC over the Internet. Configure the employees’ laptops to connect to this VPN.

* ​

Using the current Direct Connect connection, create a new public virtual interface and input the network prefixes that you want to advertise. Create a new site-to-site VPN connection to the VPC over the Internet. Configure the employees’ laptops to connect to this VPN.

**Explanation**

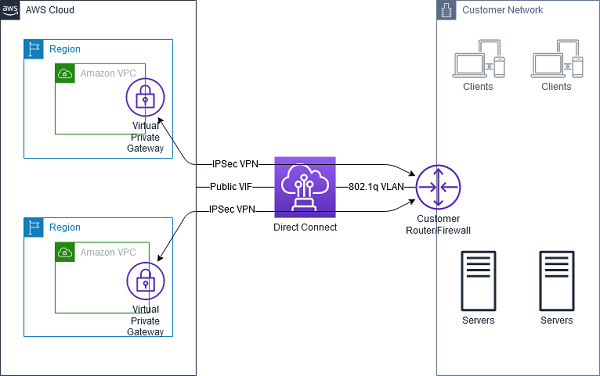
p>To connect to services such as EC2 using just Direct Connect you need to create a private virtual interface. However, if you want to encrypt the traffic flowing through Direct Connect, you will need to use the public virtual interface of DX to create a VPN connection that will allow access to AWS services such as S3, EC2, and other services.

To connect to AWS resources that are reachable by a public IP address (such as an Amazon Simple Storage Service bucket) or AWS public endpoints, use a **public virtual interface**. With a public virtual interface, you can:

- Connect to all AWS public IP addresses globally.

- Create public virtual interfaces in any DX location to receive Amazon’s global IP routes.

- Access publicly routable Amazon services in any AWS Region (except for the AWS China Region).



To connect to your resources hosted in an Amazon Virtual Private Cloud (Amazon VPC) using their private IP addresses, use a private virtual interface. With a **private virtual interface**, you can:

- Connect VPC resources (such as Amazon Elastic Compute Cloud (Amazon EC2) instances or load balancers) on your private IP address or endpoint.

- Connect a private virtual interface to a DX gateway. Then, associate the DX gateway with one or more virtual private gateways in any AWS Region (except the AWS China Region).

- Connect to multiple VPCs in any AWS Region (except the AWS China Region), because a virtual private gateway is associated with a single VPC.

If you want to establish a virtual private network (VPN) connection from your company network to an Amazon Virtual Private Cloud (Amazon VPC) over an AWS Direct Connect (DX) connection, you must use a public virtual interface for your DX connection.

Therefore, the correct answer is: **Using the current Direct Connect connection, create a new public virtual interface, and input the network prefixes that you want to advertise. Create a new site-to-site VPN connection to the VPC with the BGP protocol using the DX connection. Configure the employees’ laptops to connect to this VPN.**

The option that says: **Using the current Direct Connect connection, create a new private virtual interface, and input the network prefixes that you want to advertise. Create a new site-to-site VPN connection to the VPC with the BGP protocol using the DX connection. Configure the employees’ laptops to connect to this VPN** is incorrect because you must use a public virtual interface for your AWS Direct Connect (DX) connection and not a private one. You won't be able to establish an encrypted VPN along with your DX connection if you create a private virtual interface.

The following options are incorrect because you need to establish the VPN connection through the DX connection, and not over the Internet.

**- Using the current Direct Connect connection, create a new public virtual interface, and input the network prefixes that you want to advertise. Create a new site-to-site VPN connection to the VPC over the internet. Configure the employees’ laptops to connect to this VPN.**

**- Using the current Direct Connect connection, create a new private virtual interface, and input the network prefixes that you want to advertise. Create a new site-to-site VPN connection to the VPC over the internet. Configure the employees’ laptops to connect to this VPN.**

**References:**

<https://aws.amazon.com/premiumsupport/knowledge-center/public-private-interface-dx/>

<https://docs.aws.amazon.com/whitepapers/latest/aws-vpc-connectivity-options/aws-direct-connect-vpn.html>

<https://aws.amazon.com/premiumsupport/knowledge-center/create-vpn-direct-connect/>

**Check out this AWS Direct Connect Cheat Sheet:**

[https://tutorialsdojo.com/aws-direct-connect/](https://tutorialsdojo.com/aws-direct-connect/?src=udemy)

Bottom of Form

Top of Form

Question 46: Skipped

A company has created multiple accounts in AWS to support the rapid growth of its cloud services. The multiple accounts are used to separate their various departments such as finance, human resources, engineering, and many others. Each account is managed by a Systems Administrator which has the root access for that specific account only. There is a requirement to centrally manage policies across multiple AWS accounts by allowing or denying particular AWS services for individual accounts, or for groups of accounts.

Which is the most suitable solution that you should implement with the LEAST amount of complexity?

* ​

Provide access to externally authenticated users via Identity Federation. Set up an IAM role to specify permissions for users from each department whose identity is federated from your organization or a third-party identity provider.

* ​

Use AWS Organizations and Service Control Policies to control the list of AWS services that can be used by each member account.

**(Correct)**

* ​

Connect all departments by setting up a cross-account access to each of the AWS accounts of the company. Create and attach IAM policies to your resources based on their respective departments to control access.

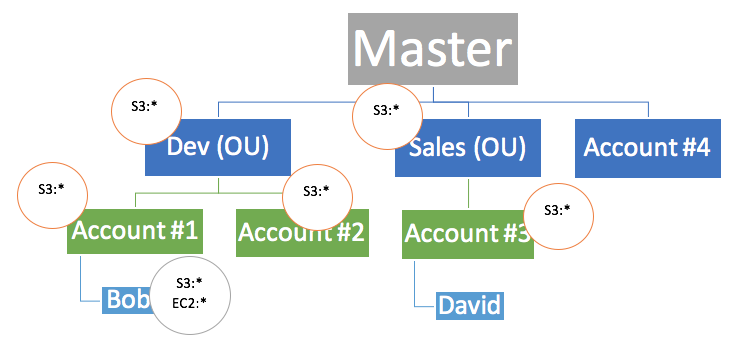
* ​

Set up AWS Organizations and Organizational Units (OU) to connect all AWS accounts of each department. Create custom IAM Policy to allow or deny the use of certain AWS services for each account.

**Explanation**

***AWS Organizations*** offers policy-based management for multiple AWS accounts. With Organizations, you can create groups of accounts, automate account creation, and apply and manage policies for those groups. Organizations enables you to centrally manage policies across multiple accounts, without requiring custom scripts and manual processes. It allows you to create ***Service Control Policies (SCPs)*** that centrally control AWS service use across multiple AWS accounts.

Remember that AWS Organizations **does not** replace associating IAM policies with users, groups, and roles within an AWS account. Hence, you still need to set up appropriate IAM policies for your root and member accounts.



IAM policies let you allow or deny access to AWS services (such as Amazon S3), individual AWS resources (such as a specific S3 bucket), or individual API actions (such as s3:CreateBucket). An IAM policy can be applied only to IAM users, groups, or roles, and it can never restrict the root identity of the AWS account.

By contrast, AWS Organizations lets you use service control policies (SCPs) to allow or deny access to particular AWS services for individual AWS accounts, or for groups of accounts within an organizational unit (OU). The specified actions from an attached SCP affect all IAM users, groups, and roles for an account, including the root account identity.

When you apply an SCP to an OU or an individual AWS account, you choose to either **enable**(whitelist), or **disable**(blacklist) the specified AWS service. Access to any service that isn’t explicitly allowed by the SCPs associated with an account, its parent OUs, or the master account is **denied**to the AWS accounts or OUs associated with the SCP. When an SCP is applied to an OU, it is inherited by all of the AWS accounts in that OU.

***Setting up AWS Organizations and Organizational Units (OU) to connect all AWS accounts of each department and creating custom IAM Policy to allow or deny the use of certain AWS services for each account*** is incorrect because although it is correct to use AWS Organizations, this option is incorrect about IAM Policy. It is the Service Control Policy (SCP) which enables you to allow or deny the use of certain AWS services for each account, and not the IAM Policy.

***Connecting all departments by setting up a cross-account access to each of the AWS accounts of the company, then creating and attaching IAM policies to your resources based on their respective departments to control access*** is incorrect because although you can set up cross-account access to each department, this entails a lot of configuration compared with using AWS Organizations and Service Control Policies (SCPs). Cross-account access would be a more suitable choice if you only have two accounts to manage, but not for multiple accounts.

***Providing access to externally authenticated users via Identity Federation and setting up an IAM role to specify permissions for users from each department whose identity is federated from your organization or a third-party identity provider*** is incorrect as this option is focused on the Identity Federation authentication set up for your AWS accounts but not the IAM policy management for multiple AWS accounts. A combination of AWS Organizations and Service Control Policies (SCPs) is a better choice compared to this option.

**Reference:**

<https://aws.amazon.com/organizations/>

<https://aws.amazon.com/premiumsupport/knowledge-center/iam-policy-service-control-policy/>

**Check out this AWS Organizations Cheat Sheet:**

[https://tutorialsdojo.com/aws-organizations/](https://tutorialsdojo.com/aws-organizations/?src=udemy)

Bottom of Form

Top of Form

Question 47: Skipped

**A global financial company is launching their new trading platform in AWS which allows people to buy and sell their bitcoin, ethereum, ripple and other cryptocurrencies as well as access to various financial reports. To meet the anti-money laundering and counter-terrorist financing (AML/CFT) measures compliance, all report files of the trading platform must not be accessible in certain countries which are listed in the Financial Action Task Force (FATF) list of non-cooperative countries or territories. You were given a task to ensure that the company complies with this requirement to avoid hefty monetary penalties. In this scenario, what is the best way to satisfy this security requirement in AWS while still delivering content to the users around the globe with lower latency?**

* ​

Use Route53 with a Geoproximity routing policy that blocks all traffic from the blacklisted countries.

* ​

Use Route53 with a Geolocation routing policy that blocks all traffic from the blacklisted countries.

* ​

Deploy the trading platform using Elastic Beanstalk and deny all incoming traffic from the IP addresses of the blacklisted countries in the Network Access Control List (ACL) of the VPC.

* ​

Create a CloudFront distribution with Geo-Restriction enabled to block all of the blacklisted countries from accessing the trading platform.

**(Correct)**

**Explanation**

You can use geo restriction - also known as geoblocking - to prevent users in specific geographic locations from accessing content that you're distributing through a CloudFront web distribution. To use geo restriction, you have two options:

1. Use the CloudFront geo restriction feature. Use this option to restrict access to all of the files that are associated with a distribution and to restrict access at the country level.

2. Use a third-party geolocation service. Use this option to restrict access to a subset of the files that are associated with a distribution or to restrict access at a finer granularity than the country level.

Hence, the correct answer for this scenario is ***creating a CloudFront distribution with Geo-Restriction enabled to block all of the blacklisted countries from accessing the trading platform*** as CloudFront can provide the users low-latency access to the files as well as block certain countries on the FTAF list.

The options that say:***Deploy the trading platform using Elastic Beanstalk and deny all incoming traffic from the IP addresses of the blacklisted countries in the Network Access Control List (ACL) of the VPC*** is incorrect because blocking all of the IP addresses of each blacklisted country in the Network Access Control List entails a lot of work and is not a recommended way to accomplish the task. Using CloudFront geo restriction feature is a better solution for this.

The options that say: ***Use Route 53 with a Geolocation routing policy that blocks all traffic from the blacklisted countries*** and ***Use Route 53 with a Geoproximity routing policy that blocks all traffic from the blacklisted countries***are incorrect because Route 53 does not provide low-latency access to users around the globe, unlike CloudFront. Geolocation routing policy is used when you want to route traffic based on the location of your users while Geoproximity routing policy is for scenarios where you want to route traffic based on the location of your resources and, optionally, shift traffic from resources on one location to resources in another.

**Reference:**

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/georestrictions.html>

**Check out this Amazon CloudFront Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudfront/>

**Latency Routing vs Geoproximity Routing vs Geolocation Routing:**

<https://tutorialsdojo.com/aws-cheat-sheet-latency-routing-vs-geoproximity-routing-vs-geolocation-routing/>

**Comparison of AWS Services Cheat Sheets:**

<https://tutorialsdojo.com/comparison-of-aws-services-for-udemy-students/>

Bottom of Form

Top of Form

Question 48: Skipped

**You are working as an AWS Developer for a mobile development company. They are currently developing new android and iOS mobile apps and are considering storing the customization data in AWS. This would provide a more uniform cross-platform experience to their users using multiple mobile devices to access the apps. The preference data for each user is estimated to be 50KB in size. Additionally, 3 million customers are expected to use the application on a regular basis, using their social login accounts for easier user authentication. How would you design a highly available, cost-effective, scalable, and secure solution to meet the above requirements?**

* ​

Launch an RDS MySQL instance in 2 availability zones to contain the user preference data. Deploy a public facing application on a server in front of the database which will manage authentication and access controls.

* ​

Setup a table in DynamoDB containing an item for each user having the necessary attributes to hold the user preferences. The mobile app will query the user preferences directly from the table. Use STS, Web Identity Federation, and DynamoDB's Fine Grained Access Control for authentication and authorization.

**(Correct)**

* ​

Have the user preference data stored in S3, and set up a DynamoDB table with an item for each user and an item attribute referencing the user's S3 object. The mobile app will retrieve the S3 URL from DynamoDB and then access the S3 object directly utilizing STS, Web identity Federation, and S3 ACLs.

* ​

Create an RDS MySQL instance with multiple read replicas in 2 availability zones to store the user preference data. The mobile application will then query the user preferences from the read replicas. Finally, utilize MySQL's user management and access privilege system to handle security and access credentials of your users.

**Explanation**

Take note that the question mentioned the use of social media logins. With web identity federation, you don't need to create custom sign-in code or manage your own user identities. Instead, users of your app can sign in using a well-known identity provider (IdP) —such as Login with Amazon, Facebook, Google, or any other OpenID Connect (OIDC)-compatible IdP, receive an authentication token, and then exchange that token for temporary security credentials in AWS that map to an IAM role with permissions to use the resources in your AWS account. Using an IdP helps you keep your AWS account secure, because you don't have to embed and distribute long-term security credentials with your application.

The option that says: ***Set up a table in DynamoDB containing an item for each user having the necessary attributes to hold the user preferences. The mobile app will query the user preferences directly from the table. Use STS, Web Identity Federation, and DynamoDB's Fine Grained Access Control for authentication and authorization*** is correct because it uses DynamoDB for scalability and cost efficiency. It uses federated access using Web Identity Provider, and uses fine grained access privileges for authenticating the access.

The option that says: ***Have the user preference data stored in S3, and set up a DynamoDB table with an item for each user and an item attribute referencing the user's S3 object. The mobile app will retrieve the S3 URL from DynamoDB and then access the S3 object directly utilizing STS, Web identity Federation, and S3 ACLs*** is incorrect because accessing the data via S3 would be slower compared to DynamoDB.  
  
The option that says:***Launch an RDS MySQL instance in 2 availability zones to contain the user preference data, then deploying a public facing application on a server in front of the database which will manage authentication and access controls*** is incorrect because RDS MySQL is not as scalable and cost-effective as DynamoDB.  
  
The option that says: ***Create an RDS MySQL instance with multiple read replicas in 2 availability zones to store the user preference data. The mobile application will then query the user preferences from the read replicas. Finally, utilize MySQL's user management and access privilege system to handle security and access credentials of your users*** is incorrect because RDS MySQL is not as scalable and cost-effective as DynamoDB, and user management and access privilege system cannot be used for controlling access.

**Reference**:

<https://aws.amazon.com/dynamodb/developer-resources>

**Check out this Amazon DynamoDB Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-dynamodb/>

Bottom of Form

Top of Form

Question 49: Skipped

A company has recently adopted a hybrid cloud architecture which requires them to migrate their databases from their on-premises data center to AWS. One of their applications requires a heterogeneous database migration in which they need to transform their on-premises Oracle database to PostgreSQL. A schema and code transformation should be done first in order to successfully migrate the data.

Which of the following options is the most suitable approach to migrate the database in AWS?

* ​

Use a combination of AWS Data Pipeline service and CodeCommit to convert the source schema and code to match that of the target PostgreSQL database in RDS. Use AWS Batch with Spot EC2 instances to cost-effectively migrate the data from the source database to the target database in a batch process.

* ​

Use the AWS Serverless Application Model (SAM) service to transform your database to PostgreSQL using AWS Lambda functions. Migrate the database to RDS using the AWS Database Migration Service (DMS).

* ​

Use the AWS Schema Conversion Tool (SCT) to convert the source schema to match that of the target database. Migrate the data using the AWS Database Migration Service (DMS) from the source database to an Amazon RDS for PostgreSQL database.

**(Correct)**

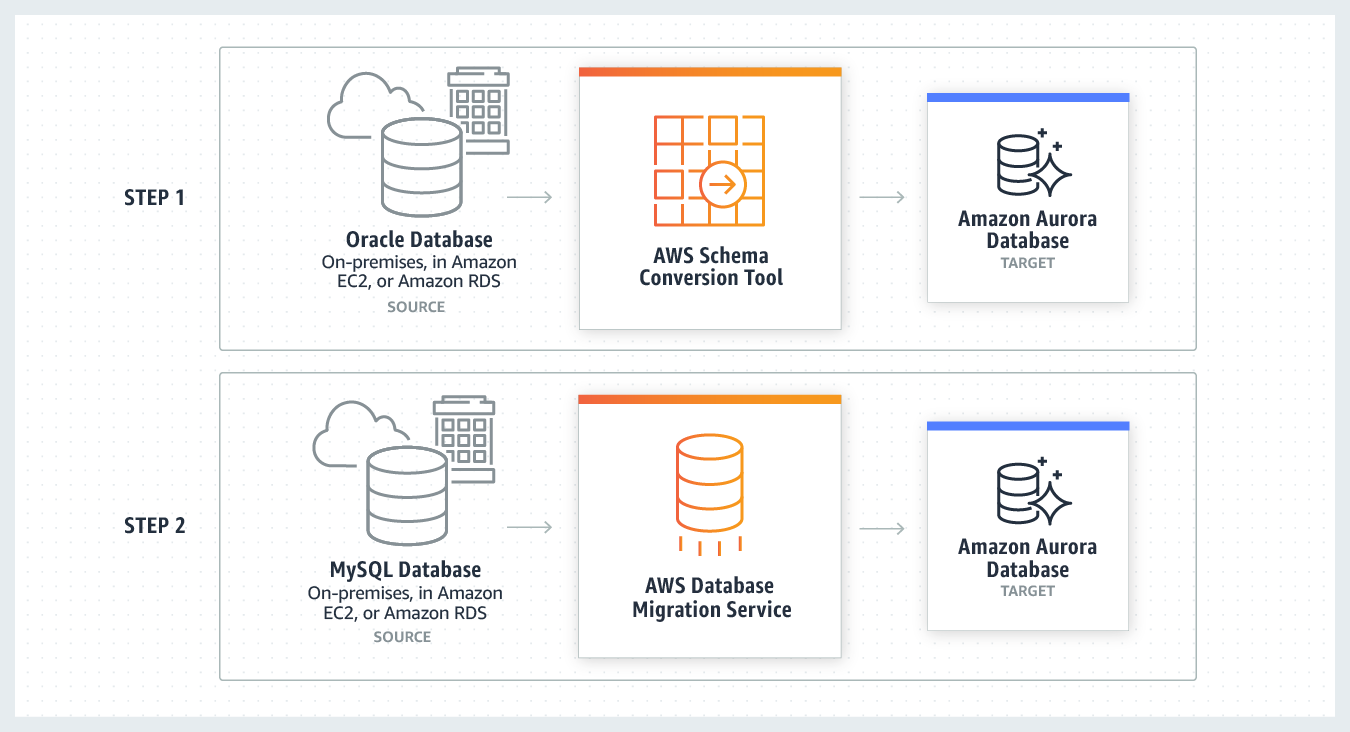
* ​

Migrate the database from your on-premises data center using the AWS Server Migration Service (SMS). Afterwards, use the AWS Database Migration Service to convert and migrate your data to Amazon RDS for PostgreSQL database.

**Explanation**

AWS Database Migration Service helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. The AWS Database Migration Service can migrate your data to and from the most widely used commercial and open-source databases.

AWS Database Migration Service can migrate your data to and from most of the widely used commercial and open source databases. It supports homogeneous migrations such as Oracle to Oracle, as well as heterogeneous migrations between different database platforms, such as Oracle to Amazon Aurora. Migrations can be from on-premises databases to Amazon RDS or Amazon EC2, databases running on EC2 to RDS, or vice versa, as well as from one RDS database to another RDS database. It can also move data between SQL, NoSQL, and text-based targets.



In heterogeneous database migrations, the source and target databases engines are different, like in the case of Oracle to Amazon Aurora, Oracle to PostgreSQL, or Microsoft SQL Server to MySQL migrations. In this case, the schema structure, data types, and database code of source and target databases can be quite different, requiring a schema and code transformation before the data migration starts. That makes heterogeneous migrations a two-step process.

First ***use the AWS Schema Conversion Tool to convert the source schema and code to match that of the target database, and then use the AWS Database Migration Service to migrate data from the source database to the target database.*** All the required data type conversions will automatically be done by the AWS Database Migration Service during the migration. The source database can be located in your own premises outside of AWS, running on an Amazon EC2 instance, or it can be an Amazon RDS database. The target can be a database in Amazon EC2 or Amazon RDS.

The option that says: ***Migrate the database from your on-premises data center using the AWS Server Migration Service (SMS). Afterwards, use the AWS Database Migration Service to convert and migrate your data to Amazon RDS for PostgreSQL database*** is incorrect because the AWS Server Migration Service (SMS) is primarily used to migrate virtual machines such as VMware vSphere and Windows Hyper-V. Although it is correct to use AWS Database Migration Service (DMS) to migrate the database, this option is still wrong because you should use the AWS Schema Conversion Tool to convert the source schema.

The option that says: ***Use a combination of AWS Data Pipeline service and CodeCommit to convert the source schema and code to match that of the target PostgreSQL database in RDS. Use AWS Batch with Spot EC2 instances to cost-effectively migrate the data from the source database to the target database in a batch process*** is incorrect because AWS Data Pipeline is primarily used to quickly and easily provision pipelines that remove the development and maintenance effort required to manage your daily data operations which lets you focus on generating insights from that data. Although you can use this to connect your data on your on-premises data center, it is not the most suitable service to use, compared with AWS DMS.

The option that says: ***Use the AWS Serverless Application Model (SAM) service to transform your database to PostgreSQL using AWS Lambda functions. Migrate the database to RDS using the AWS Database Migration Service (DMS)*** is incorrect because the Serverless Application Model (SAM) is an open-source framework that is primarily used to build serverless applications on AWS, and not for database migration.

**References:**

<https://aws.amazon.com/dms/>

<https://aws.amazon.com/cloud-migration/>

**Check out these cheat sheets on AWS Database Migration Service and other common strategies for cloud migration:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-database-migration-service/>

<https://tutorialsdojo.com/aws-cheat-sheet-aws-migration-strategies-the-6-rs/>

Bottom of Form

Top of Form

Question 50: Skipped

A company uses computer simulations for modeling weather patterns in a certain country. The simulations generate terabytes of data, which is stored in a MySQL 5.6 database that runs in an Amazon EC2 instance. A Ruby on Rails application is hosted on a separate EC2 instance to process the data. The current database size is 16 TB and is expected to grow as more complex simulations are created continuously. The facility wants to re-architect its infrastructure to be highly scalable and highly available as they need to run the application reliably 24x7.

Which of the following is the MOST cost-effective solution that can satisfy the above requirements?

* ​

Configure your application tier to run on an Auto Scaling group of smaller sized EC2 instances behind an Application Load Balancer. Purchase Reserved EC2 instances for fixed capacity and let the Auto Scaling instances run on demand. Migrate the MySQL database to Amazon Aurora. Create a read-replica on another Availability Zone of the Aurora instance for high availability.

**(Correct)**

* ​

Purchase Reserved Amazon EC2 instances for the application instance and database instances to save costs. Create an EC2 Auto Scaling group with Multi-AZ configuration behind a load balancer for the application tier. Implement a master-slave setup for your database tier. Use Logical Volume Management on the database instances to easily attach new EBS volumes and expand the file system.

* ​

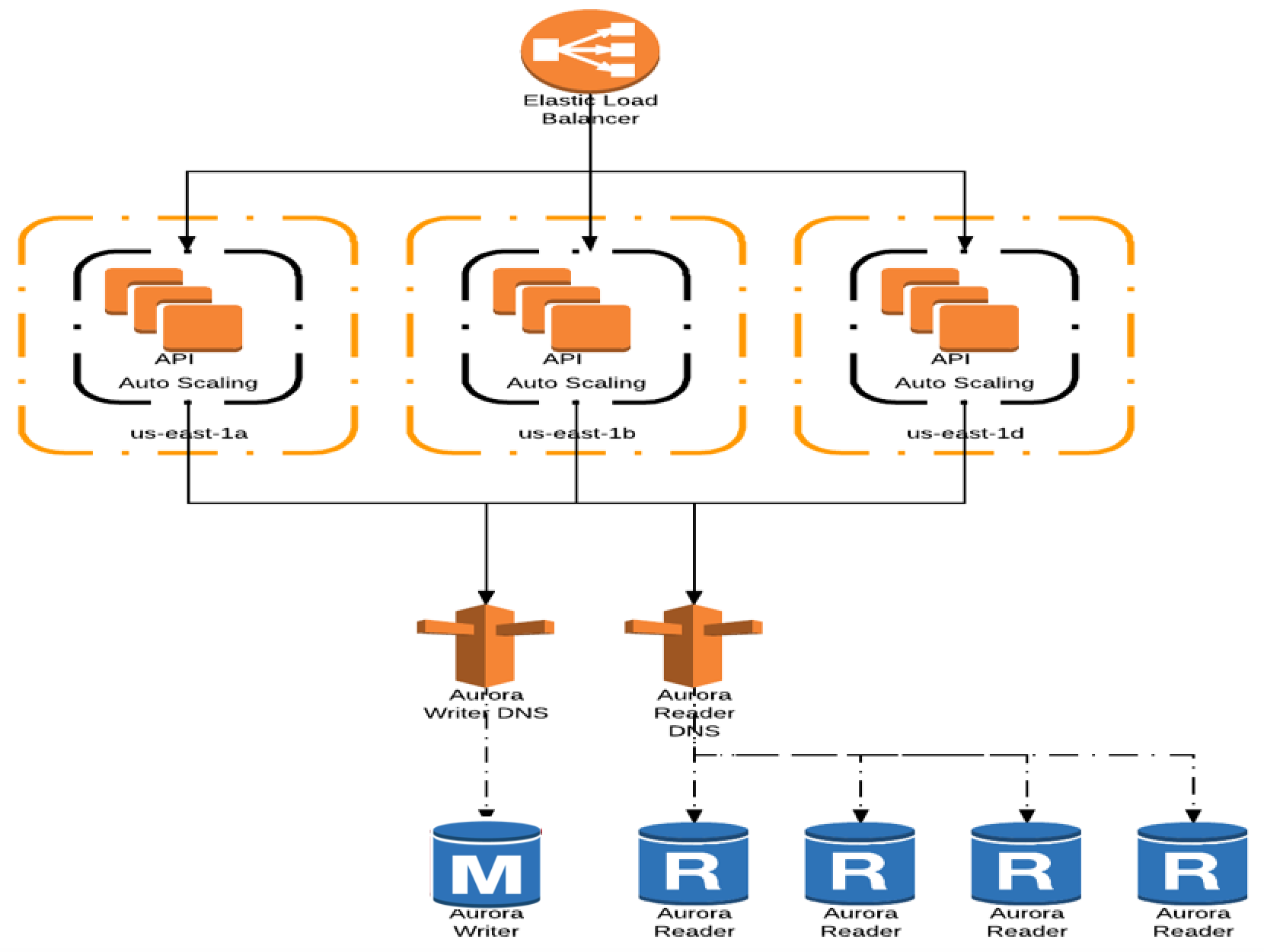
Convert the application tier to Lambda@Edge functions for high availability, scalability, and cost-effectiveness. Migrate the MySQL database to an Amazon RDS MySQL instance with Multi-AZ enabled. Enable storage autoscaling on the RDS instance.

* ​

Configure your application tier to run on an Auto Scaling group of smaller sized EC2 instances behind an Application Load Balancer. Purchase reserved EC2 instances for fixed capacity and let the Auto Scaling instances run on demand. Migrate the MySQL database to an Amazon RDS MySQL instance with Multi-AZ enabled. Adjust the RDS Storage volume manually as demand increases.

**Explanation**

You can launch and automatically scale a fleet of On-Demand Instances and Spot Instances within a single Auto Scaling group. In addition to receiving discounts for using Spot Instances, you can use Reserved Instances or a Savings Plan to receive discounted rates of the regular On-Demand Instance pricing. All of these factors combined help you to optimize your cost savings for Amazon EC2 instances, while making sure that you obtain the desired scale and performance for your application.



Amazon Aurora storage automatically scales with the data in your cluster volume. As your data grows, your cluster volume storage expands up to a maximum of 128 tebibytes (TiB). Even though an Aurora cluster volume can scale up in size to many tebibytes, you are only charged for the space that you use in the volume. The mechanism for determining billed storage space depends on the version of your Aurora cluster. Aurora stores copies of the data in a DB cluster across multiple Availability Zones in a single AWS Region. Aurora stores these copies regardless of whether the instances in the DB cluster span multiple Availability Zones. However, you still need to create a read-replica for the high availability of the Aurora DB instance. A single read-replica is enough to quickly recover the database in case the primary instance fails.

Therefore, the correct answer is:**Configure your application tier to run on an Auto Scaling group of smaller sized EC2 instances behind an Application Load Balancer. Purchase Reserved EC2 instances for fixed capacity and let the Auto Scaling instances to run on demand. Migrate the MySQL database to Amazon Aurora. Create a read-replica on another Availability Zone of the Aurora instance for high-availability.**

The option that says: **Purchase Reserved Amazon EC2 instances for the application instance and database instances to save costs. Create an EC2 Auto Scaling group with Multi-AZ configuration behind a load balancer for the application tier. Implement a master-slave setup for your database tier. Use Logical Volume Management on the database instances to easily attach new EBS volumes and expand the file system** is incorrect because running your own MySQL server on an EC2 instance entails significant management overhead. This is not scalable as you need to manually resize and configure the EC2 instance as you will eventually need more compute or storage capacity in the future. A better solution is to use Amazon Aurora instead.

The option that says: **Convert the application tier to Lambda@Edge functions for high availability, scalability, and cost-effectiveness. Migrate the MySQL database to an Amazon RDS MySQL instance with Multi-AZ enabled. Enable storage autoscaling on the RDS instance** is incorrect. Although it is true that MySQL RDS instances can support storage autoscaling for up to 64TB, using Lambda@Edge to run the application is not warranted. Lambda@Edge is just an extension of AWS Lambda that customizes the content of what Amazon CloudFront delivers. It is not stated in the scenario the application is used globally. This would be a viable option if you convert your application to Docker-based containers and run them on Amazon ECS.

The option that says: **Configure your application tier to run on an Auto Scaling group of smaller sized EC2 instances behind an Application Load Balancer. Purchase reserved EC2 instances for fixed capacity and let the Auto Scaling instances to run on demand. Migrate the MySQL database to an Amazon RDS MySQL instance with Multi-AZ enabled. Adjust the RDS Storage volume manually as demand increases** is incorrect. Although this is possible, you still need to manually adjust the disk capacity of your RDS instance. It is better to use Amazon Aurora as it automatically scales disk storage by default.

**References:**

<https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Overview.StorageReliability.html>

<https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Concepts.AuroraHighAvailability.html>

<https://aws.amazon.com/about-aws/whats-new/2019/05/amazon-rds-mysql-mariadb-postgresql-64tib-support/>

<https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_PIOPS.StorageTypes.html>

**Check out this Amazon RDS Cheat Sheet:**

[https://tutorialsdojo.com/amazon-relational-database-service-amazon-rds/](https://tutorialsdojo.com/amazon-relational-database-service-amazon-rds/?src=udemy)

Bottom of Form

Top of Form

Question 51: Skipped

A company provides big data services to enterprise clients around the globe. One of the clients has 60 TB of raw data from their on-premises Oracle data warehouse. The data is to be migrated to Amazon Redshift. However, the database receives minor updates on a daily basis while major updates are scheduled every end of the month. The migration process must be completed within approximately 30 days before the next major update on the Redshift database. The company can only allocate 50 Mbps of Internet connection for this activity to avoid impacting business operations.

Which of the following actions will satisfy the migration requirements of the company while keeping the costs low?

* ​

Create an AWS Snowball import job to request for a Snowball Edge device. Use the AWS Schema Conversion Tool (SCT) to process the on-premises data warehouse and load it to the Snowball Edge device. Install the extraction agent on a separate on-premises server and register it with AWS SCT. Once the Snowball Edge imports data to the S3 bucket, use AWS SCT to migrate the data to Amazon Redshift. Configure a local task and AWS DMS task to replicate the ongoing updates to the data warehouse. Monitor and verify that the data migration is complete.

**(Correct)**

* ​

Create an AWS Snowball Edge job using the AWS Snowball console. Export all data from the Oracle data warehouse to the Snowball Edge device. Once the Snowball device is returned to Amazon and data is imported to an S3 bucket, create an Oracle RDS instance to import the data. Create an AWS Schema Conversion Tool (SCT) project with AWS DMS task to migrate the Oracle database to Amazon Redshift. Copy the missing daily updates from Oracle in the data center to the RDS for Oracle database over the Internet. Monitor and verify if the data migration is complete before the cut-over.

* ​

Create a new Oracle Database on Amazon RDS. Configure Site-to-Site VPN connection from the on-premises data center to the Amazon VPC. Configure replication from the on-premises database to Amazon RDS. Once replication is complete, create an AWS Schema Conversion Tool (SCT) project with AWS DMS task to migrate the Oracle database to Amazon Redshift. Monitor and verify if the data migration is complete before the cut-over.

* ​

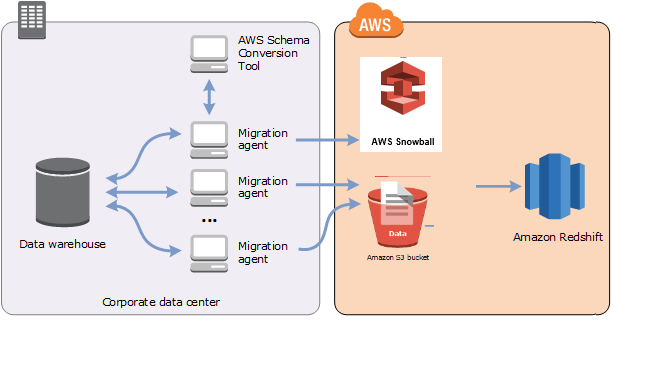
Since you have a 30-day window for migration, configure VPN connectivity between AWS and the company's data center by provisioning a 1 Gbps AWS Direct Connect connection. Launch an Oracle Real Application Clusters (RAC) database on an EC2 instance and set it up to fetch and synchronize the data from the on-premises Oracle database. Once replication is complete, create an AWS DMS task on an AWS SCT project to migrate the Oracle database to Amazon Redshift. Monitor and verify if the data migration is complete before the cut-over.

**Explanation**

You can use an **AWS SCT** agent to extract data from your on-premises data warehouse and migrate it to Amazon Redshift. The agent extracts your data and uploads the data to either Amazon S3 or, for large-scale migrations, an AWS Snowball Edge device. You can then use AWS SCT to copy the data to Amazon Redshift.

Large-scale data migrations can include many terabytes of information and can be slowed by network performance and by the sheer amount of data that has to be moved. AWS Snowball Edge is an AWS service you can use to transfer data to the cloud at faster-than-network speeds using an AWS-owned appliance. An AWS Snowball Edge device can hold up to 100 TB of data. It uses 256-bit encryption and an industry-standard Trusted Platform Module (TPM) to ensure both security and full chain-of-custody for your data. AWS SCT works with AWS Snowball Edge devices.

When you use AWS SCT and an AWS Snowball Edge device, you migrate your data in two stages. First, you use the AWS SCT to process the data locally and then move that data to the AWS Snowball Edge device. You then send the device to AWS using the AWS Snowball Edge process, and then AWS automatically loads the data into an Amazon S3 bucket. Next, when the data is available on Amazon S3, you use AWS SCT to migrate the data to Amazon Redshift. Data extraction agents can work in the background while AWS SCT is closed. You manage your extraction agents by using AWS SCT. The extraction agents act as listeners. When they receive instructions from AWS SCT, they extract data from your data warehouse.



Therefore, the correct answer is: **Create an AWS Snowball import job to request for a Snowball Edge device. Use the AWS Schema Conversion Tool (SCT) to process the on-premises data warehouse and load it to the Snowball Edge device. Install the extraction agent on a separate on-premises server and register it with AWS SCT. Once the Snowball Edge imports data to the S3 bucket, use AWS SCT to migrate the data to Amazon Redshift. Configure a local task and AWS DMS task to replicate the ongoing updates to the data warehouse. Monitor and verify that the data migration is complete.**

The option that says: **Create a new Oracle Database on Amazon RDS. Configure Site-to-Site VPN connection from the on-premises data center to the Amazon VPC. Configure replication from the on-premises database to Amazon RDS. Once replication is complete, create an AWS Schema Conversion Tool (SCT) project with AWS DMS task to migrate the Oracle database to Amazon Redshift. Monitor and verify if the data migration is complete before the cut-over** is incorrect. Replicating 60 TB worth of data over the public Internet will take several days over the 30-day migration window. It is also stated in the scenario that the company can only allocate 50 Mbps of Internet connection for the migration activity. Sending the data over the Internet could potentially affect business operations.

The option that says: **Create an AWS Snowball Edge job using the AWS Snowball console. Export all data from the Oracle data warehouse to the Snowball Edge device. Once the Snowball device is returned to Amazon and data is imported to an S3 bucket, create an Oracle RDS instance to import the data. Create an AWS Schema Conversion Tool (SCT) project with AWS DMS task to migrate the Oracle database to Amazon Redshift. Copy the missing daily updates from Oracle in the data center to the RDS for Oracle database over the internet. Monitor and verify if the data migration is complete before the cut-over** is incorrect. You need to configure the data extraction agent first on your on-premises server. In addition, you don't need the data to be imported and exported via Amazon RDS. AWS DMS can directly migrate the data to Amazon Redshift.

The option that says: **Since you have a 30-day window for migration, configure VPN connectivity between AWS and the company's data center by provisioning a 1 Gbps AWS Direct Connect connection. Install Oracle database on an EC2 instance that is configured to synchronize with the on-premises Oracle database. Once replication is complete, create an AWS DMS task on an AWS SCT project to migrate the Oracle database to Amazon Redshift. Monitor and verify if the data migration is complete before the cut-over Since you have a 30-day window for migration, configure VPN connectivity between AWS and the company's data center by provisioning a 1 Gbps AWS Direct Connect connection. Launch an Oracle Real Application Clusters (RAC) database on an EC2 instance and set it up to fetch and synchronize the data from the on-premises Oracle database. Once replication is complete, create an AWS DMS task on an AWS SCT project to migrate the Oracle database to Amazon Redshift. Monitor and verify if the data migration is complete before the cut-over** is incorrect. Although this is possible, the company wants to keep the cost low. Using a Direct Connect connection for a one-time migration is not a cost-effective solution.

**References:**

<https://aws.amazon.com/getting-started/hands-on/migrate-oracle-to-amazon-redshift/>

<https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/agents.dw.html>

<https://docs.aws.amazon.com/SchemaConversionTool/latest/userguide/agents.html>

**Tutorials Dojo's AWS Certified Solutions Architect Professional Exam Study Guide:**

[https://tutorialsdojo.com/aws-certified-solutions-architect-professional/](https://tutorialsdojo.com/aws-certified-solutions-architect-professional/?src=udemy)

Bottom of Form

Top of Form

Question 52: Skipped

A top university has launched its serverless online portal using Lambda and API Gateway in AWS that enables its students to enroll, manage their class schedule, and see their grades online. After a few weeks, the portal abruptly stopped working and lost all of its data. The university hired an external cyber security consultant and based on the investigation, the outage was due to an SQL injection vulnerability on the portal's login page in which the attacker simply injected the malicious SQL code. You also need to track historical changes to the rules and metrics associated to your firewall.

Which of the following is the most suitable and cost-effective solution to avoid another SQL Injection attack against their infrastructure in AWS?

* ​

Use AWS WAF to add a web access control list (web ACL) in front of the Lambda functions to block requests that contain malicious SQL code. Use AWS Firewall Manager, to track changes to your web access control lists (web ACLs) such as the creation and deletion of rules including the updates to the WAF rule configurations.

* ​

Create a new Application Load Balancer (ALB) and set up AWS WAF in the load balancer. Place the API Gateway behind the ALB and configure a web access control list (web ACL) in front of the ALB to block requests that contain malicious SQL code. Use AWS Firewall Manager to track changes to your web access control lists (web ACLs) such as the creation and deletion of rules including the updates to the WAF rule configurations.

* ​

Block the IP address of the attacker in the Network Access Control List of your VPC and then set up a CloudFront distribution. Set up AWS WAF to add a web access control list (web ACL) in front of the CloudFront distribution to block requests that contain malicious SQL code. Use AWS Config to track changes to your web access control lists (web ACLs) such as the creation and deletion of rules including the updates to the WAF rule configurations.

* ​

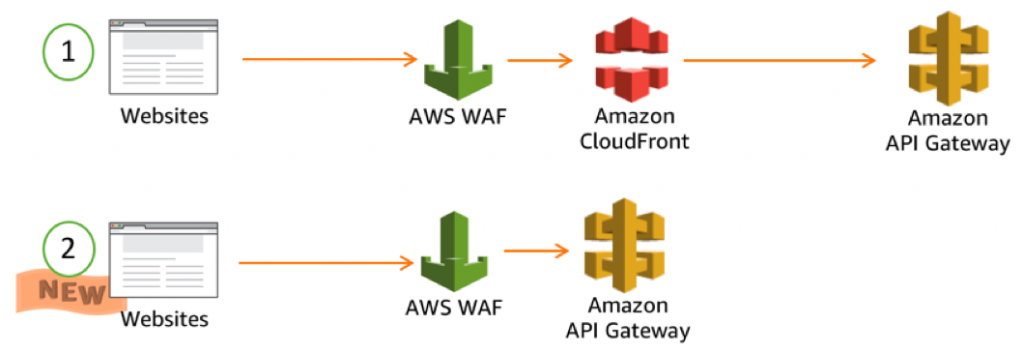
Use AWS WAF to add a web access control list (web ACL) in front of the API Gateway to block requests that contain malicious SQL code. Use AWS Config to track changes to your web access control lists (web ACLs) such as the creation and deletion of rules including the updates to the WAF rule configurations.

**(Correct)**

**Explanation**

In this scenario, the correct answer is to deploy WAF in front of the API Gateway, hence the correct answer is the option that says: ***Use AWS WAF to add a web access control list (web ACL) in front of the API Gateway to block requests that contain malicious SQL code. Use AWS Config to track changes to your web access control lists (web ACLs) such as the creation and deletion of rules including the updates to the WAF rule configurations***.

AWS WAF is a web application firewall that helps protect your web applications from common web exploits that could affect application availability, compromise security, or consume excessive resources. With AWS Config, you can track changes to WAF web access control lists (web ACLs). For example, you can record the creation and deletion of rules and rule actions, as well as updates to WAF rule configurations.



AWS WAF gives you control over which traffic to allow or block to your web applications by defining customizable web security rules. You can use AWS WAF to create custom rules that block common attack patterns, such as SQL injection or cross-site scripting, and rules that are designed for your specific application. New rules can be deployed within minutes, letting you respond quickly to changing traffic patterns. Also, AWS WAF includes a full-featured API that you can use to automate the creation, deployment, and maintenance of web security rules.

The option that says: ***Use AWS WAF to add a web access control list (web ACL) in front of the Lambda functions to block requests that contain malicious SQL code. Use AWS Firewall Manager, to track changes to your web access control lists (web ACLs) such as the creation and deletion of rules including the updates to the WAF rule configurations***is incorrect because you have to use AWS WAF in front of the API Gateway and not directly to the Lambda functions. AWS Firewall Manager is primarily used to manage your Firewall across multiple AWS accounts under your AWS Organizations and hence, it is not suitable for tracking changes to WAF web access control lists. You should use AWS Config instead.

The option that says: ***Block the IP address of the attacker in the Network Access Control List of your VPC and then set up a CloudFront distribution. Set up AWS WAF to add a web access control list (web ACL) in front of the CloudFront distribution to block requests that contain malicious SQL code. Use AWS Config to track changes to your web access control lists (web ACLs) such as the creation and deletion of rules including the updates to the WAF rule configurations*** is incorrect because even though it is valid to use AWS WAF with CloudFront, it entails an additional and unnecessary cost to launch a CloudFront distribution for this scenario. There is no requirement that the serverless online portal should be scalable and be accessible around the globe hence, a CloudFront distribution is not relevant.

The option that says: ***Create a new Application Load Balancer (ALB) and set up AWS WAF in the load balancer. Place the API Gateway behind the ALB and configure a web access control list (web ACL) in front of the ALB to block requests that contain malicious SQL code. Use AWS Firewall Manager to track changes to your web access control lists (web ACLs) such as the creation and deletion of rules including the updates to the WAF rule configurations*** is incorrect because launching a new Application Load Balancer entails additional cost and is not cost-effective. In addition, AWS Firewall manager is primarily used to manage your Firewall across multiple AWS accounts under your AWS Organizations. Using AWS Config is much more suitable for tracking changes to WAF web access control lists.

**References:**

<https://aws.amazon.com/waf/>

<https://docs.aws.amazon.com/waf/latest/developerguide/what-is-aws-waf.html>

**Check out this AWS WAF Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-waf/>

**Tutorials Dojo's AWS Certified Solutions Architect Professional Exam Study Guide:**

[https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-solutions-architect-professional/](https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-solutions-architect-professional/?src=udemy)

Bottom of Form

Top of Form

Question 53: Skipped

A company processes several petabytes of images submitted by users on their photo hosting site every month. Each month, the images are processed in its on-premises data center by a High-Performance Computing (HPC) cluster with a capacity of 5,000 cores and 10 petabytes of data. Processing a month’s worth of images by thousands of jobs running in parallel takes about a week and the processed images are stored on a network file server, which also backups the data to a disaster recovery site.

The current data center is nearing its capacity so the users are forced to spread the jobs within the course of the month. This is not ideal for the requirement of the jobs, so the Solutions Architect was tasked to design a scalable solution that can exceed the current capacity with the least amount of management overhead while maintaining the current level of durability.

Which of the following solutions will meet the company's requirements while being cost-effective?

* ​

Utilize AWS Batch with Managed Compute Environments to create a fleet using Spot Instances. Store the raw data on an Amazon S3 bucket. Create jobs on AWS Batch Job Queues that will pull objects from the Amazon S3 bucket and temporarily store them to the EC2 EBS volumes for processing. Send the processed images back to another Amazon S3 bucket.

**(Correct)**

* ​

Create an Amazon SQS queue and submit the list of jobs to be processed. Create an Auto Scaling Group of Amazon EC2 Spot Instances that will process the jobs from the SQS queue. Share the raw data across all the instances using Amazon EFS. Store the processed images in an Amazon S3 bucket for long term storage.

* ​

Using a combination of On-demand and Reserved Instances as Task Nodes, create an EMR cluster that will use Spark to pull the raw data from an Amazon S3 bucket. List the jobs that need to be processed by the EMR cluster on a DynamoDB table. Store the processed images on a separate Amazon S3 bucket.

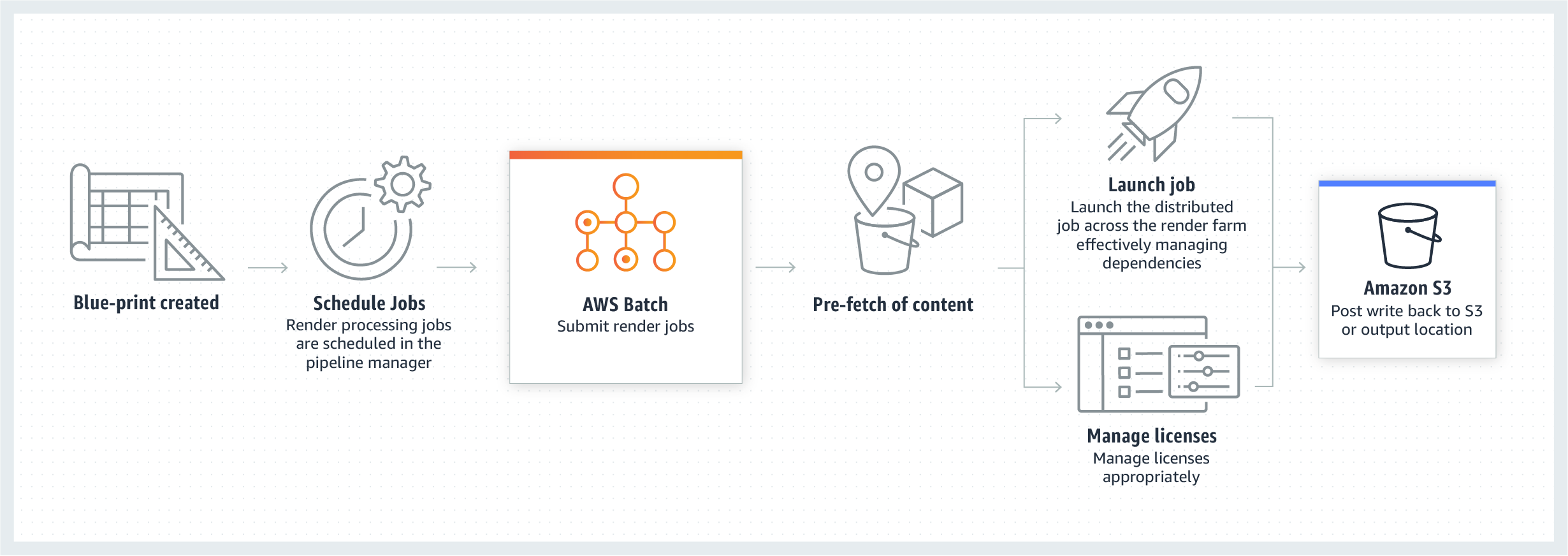
* ​

Package the executable file for the job on a Docker image hosted on Amazon Elastic Container Service. Use the Docker image to run Amazon ECS tasks with a fleet of Spot Instances in an Auto Scaling group. Store the raw data temporarily on Amazon EBS SC1 volumes and then send the images to an Amazon S3 bucket after processing.

**Explanation**

**AWS Batch** enables developers, scientists, and engineers to easily and efficiently run hundreds of thousands of batch computing jobs on AWS. AWS Batch dynamically provisions the optimal quantity and type of compute resources (e.g., CPU or memory optimized instances) based on the volume and specific resource requirements of the batch jobs submitted. With AWS Batch, there is no need to install and manage batch computing software or server clusters that you use to run your jobs, allowing you to focus on analyzing results and solving problems.

There is no additional charge for AWS Batch. You only pay for the AWS resources (e.g. EC2 instances or Fargate jobs) you create to store and run your batch jobs. From the AWS Batch use cases page, we can see an example similar to this scenario wherein Digital Media and Entertainment companies require highly scalable batch computing resources to enable accelerated and automated processing of data as well as the compilation and processing of files, graphics, and visual effects for high-resolution video content. Use AWS Batch to accelerate content creation, dynamically scale media packaging, and automate asynchronous media supply chain workflows.



In AWS Batch, job queues are mapped to one or more compute environments. Compute environments contain the Amazon ECS container instances that are used to run containerized batch jobs. A specific compute environment can also be mapped to one or many job queues. Within a job queue, the associated compute environments each have an order that's used by the scheduler to determine where jobs that are ready to be run should run.

Therefore, the correct answer is:**Utilize AWS Batch with Managed Compute Environments to create a fleet using Spot Instances. Store the raw data on an Amazon S3 bucket. Create jobs on AWS Batch Job Queues that will pull objects from the Amazon S3 bucket and temporarily store them to the EC2 EBS volumes for processing. Send the processed images back to another Amazon S3 bucket.**

The option that says: **Package the executable file for the job on a Docker image hosted on Amazon Elastic Container Service. Use the Docker image to run Amazon ECS tasks with a fleet of Spot Instances in an Auto Scaling group. Store the raw data temporarily on Amazon EBS SC1 volumes and then send the images to an Amazon S3 bucket after processing** is incorrect. Although this is possible, managing the ECS cluster adds management overhead and since you can’t quickly increase/decrease SC1 EBS volumes, creating a large volume to handle petabytes of data is not economical.

The option that says: **Using a combination of On-demand and Reserved Instances as Task Nodes, create an EMR cluster that will use Apache Spark to pull the raw data from an Amazon S3 bucket. List the jobs that need to be processed by the EMR cluster on a DynamoDB table. Store the processed images on a separate Amazon S3 bucket** is incorrect as managing the EMR cluster and Apache Spark adds significant management overhead for this solution. There is also an additional cost for the EC2 instances that are constantly running even if there are only a few jobs that need to be run.

The option that says: **Create an Amazon SQS queue and submit the list of jobs to be processed. Create an Auto Scaling Group of Amazon EC2 Spot Instances that will process the jobs from the SQS queue. Share the raw data across all the instances using Amazon EFS. Store the processed images in an Amazon S3 bucket for long term storage** is incorrect as Amazon EFS is more expensive than storing the raw data on S3 buckets. This is also not efficient as listing the jobs on SQS Queue can cause some to be processed twice, depending on the state of your Spot instances.

**References:**

<https://docs.aws.amazon.com/batch/latest/userguide/compute_environments.html>

<https://aws.amazon.com/batch/use-cases/>

<https://aws.amazon.com/blogs/compute/building-high-throughput-genomic-batch-workflows-on-aws-batch-layer-part-3-of-4/>

**Check out the AWS Batch Cheat Sheet:**

[https://tutorialsdojo.com/aws-batch/](https://tutorialsdojo.com/aws-batch/?src=udemy)

Bottom of Form

Top of Form

Question 54: Skipped

A company has production, development, and test environments in its software development department, and each environment contains tens to hundreds of EC2 instances, along with other AWS services. Recently, Ubuntu released a series of security patches for a critical flaw that was detected in their OS. Although this is an urgent matter, there is no guarantee yet that these patches will be bug-free and production-ready hence, the company must immediately patch all of its affected Amazon EC2 instances in all the environments, except for the production environment. The EC2 instances in the production environment will only be patched after it has been verified that the patches work effectively. Each environment also has different baseline patch requirements that needed to be satisfied.

Using the AWS Systems Manager service, how should you perform this task with the least amount of effort?

* ​

Tag each instance based on its environment and OS. Create various shell scripts for each environment that specifies which patch will serve as its baseline. Using AWS Systems Manager Run Command, place the EC2 instances into Target Groups and execute the script corresponding to each Target Group.

* ​

Tag each instance based on its environment and OS. Create a patch baseline in AWS Systems Manager Patch Manager for each environment. Categorize EC2 instances based on their tags using Patch Groups and apply the patches specified in the corresponding patch baseline to each Patch Group.

**(Correct)**

* ​

Tag each instance based on its OS. Create a patch baseline in AWS Systems Manager Patch Manager for each environment. Categorize EC2 instances based on their tags using Patch Groups and then apply the patches specified in the corresponding patch baseline to each Patch Group. Afterward, verify that the patches have been installed correctly using Patch Compliance. Record the changes to patch and association compliance statuses using AWS Config.

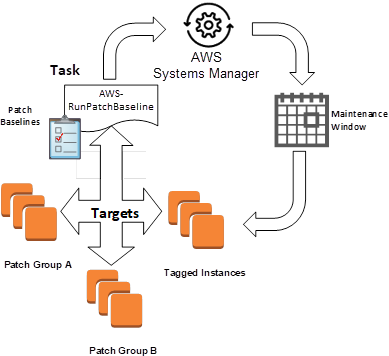
* ​

Schedule a maintenance period in AWS Systems Manager Maintenance Windows for each environment, where the period is after business hours so as not to affect daily operations. During the maintenance period, Systems Manager will execute a cron job that will install the required patches for each EC2 instance in each environment. After that, verify in Systems Manager Managed Instances that your environments are fully patched and compliant.

**Explanation**

**AWS Systems Manager Patch Manager** automates the process of patching managed instances with security-related updates. For Linux-based instances, you can also install patches for non-security updates. You can patch fleets of Amazon EC2 instances or your on-premises servers and virtual machines (VMs) by operating system type.

Patch Manager uses **patch baselines**, which include rules for auto-approving patches within days of their release, as well as a list of approved and rejected patches. You can install patches on a regular basis by scheduling patching to run as a Systems Manager Maintenance Window task. You can also install patches individually or to large groups of instances by using Amazon EC2 tags. For each auto-approval rule that you create, you can specify an auto-approval delay. This delay is the number of days of wait after the patch was released, before the patch is automatically approved for patching.



A **patch group** is an optional means of organizing instances for patching. For example, you can create patch groups for different operating systems (Linux or Windows), different environments (Development, Test, and Production), or different server functions (web servers, file servers, databases). Patch groups can help you avoid deploying patches to the wrong set of instances. They can also help you avoid deploying patches before they have been adequately tested. You create a patch group by using Amazon EC2 tags. Unlike other tagging scenarios across Systems Manager, a patch group must be defined with the tag key: **Patch Group**. After you create a patch group and tag instances, you can register the patch group with a patch baseline. By registering the patch group with a patch baseline, you ensure that the correct patches are installed during the patching execution.

Hence, the correct answer is: **Tag each instance based on its environment and OS. Create a patch baseline in AWS Systems Manager Patch Manager for each environment. Categorize EC2 instances based on their tags using Patch Groups and apply the patches specified in the corresponding patch baseline to each Patch Group.**

The option that says: **Tag each instance based on its environment and OS. Create various shell scripts for each environment that specifies which patch will serve as its baseline. Using AWS Systems Manager Run Command, place the EC2 instances into Target Groups and execute the script corresponding to each Target Group**is incorrect as this option takes more effort to perform because you are using Systems Manager Run Command instead of Patch Manager. The Run Command service enables you to automate common administrative tasks and perform ad hoc configuration changes at scale, however, it takes a lot of effort to implement this solution. You can use Patch Manager instead to perform the task required by the scenario since you need to perform this task with the least amount of effort.

The option that says: **Tag each instance based on its OS. Create a patch baseline in AWS Systems Manager Patch Manager for each environment. Categorize EC2 instances based on their tags using Patch Groups and then apply the patches specified in the corresponding patch baseline to each Patch Group. Afterward, verify that the patches have been installed correctly using Patch Compliance. Record the changes to patch and association compliance statuses using AWS Config** is incorrect because you should be tagging instances based on the environment and its OS type in which they belong and not just its OS type. This is because the type of patches that will be applied varies between the different environments. With this option, the Ubuntu EC2 instances in all of your environments, including in production, will automatically be patched.

The option that says: **Schedule a maintenance period in AWS Systems Manager Maintenance Windows for each environment, where the period is after business hours so as not to affect daily operations. During the maintenance period, Systems Manager will execute a cron job that will install the required patches for each EC2 instance in each environment. After that, verify in Systems Manager Managed Instances that your environments are fully patched and compliant** is incorrect because this is not the simplest way to address the issue using AWS Systems Manager. The AWS Systems Manager Maintenance Windows feature lets you define a schedule for when to perform potentially disruptive actions on your instances such as patching an operating system, updating drivers, or installing software or patches. Each Maintenance Window has a schedule, a maximum duration, a set of registered targets (the instances that are acted upon), and a set of registered tasks. Although this solution may work, it entails a lot of configuration and effort to implement.

**References:**

<https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-patch.html>

<https://aws.amazon.com/blogs/mt/patching-your-windows-ec2-instances-using-aws-systems-manager-patch-manager/>

**Check out this AWS Systems Manager Cheat Sheet:**

[https://tutorialsdojo.com/aws-systems-manager/](https://tutorialsdojo.com/aws-systems-manager/?src=udemy)

Bottom of Form

Top of Form

Question 55: Skipped

A company is migrating an interactive car registration web system hosted on its on-premises network to AWS Cloud. The current architecture of the system consists of a single NGINX web server and a MySQL database running on a Fedora server, which both reside in their on-premises data center. For the new cloud architecture, a load balancer must be used to evenly distribute the incoming traffic to the application servers. Route 53 must be used for both domain registration and domain management.

In this scenario, what would be the most efficient way to transfer the web application to AWS?

* ​

1. Use the AWS Application Discovery Service to migrate the NGINX web server.

2. Configure Auto Scaling to launch two web servers in two Availability Zones.

3. Launch a Multi-AZ MySQL Amazon Relational Database Service (RDS) instance in one Availability Zone only.

4. Import the data into Amazon RDS from the latest MySQL backup.

5. Use Amazon Route 53 to create a private hosted zone and point a non-alias Arecord to the ELB.

* ​

1. Use the AWS Server Migration Service (SMS) to create an EC2 AMI of the NGINX web server.

2. Configure auto-scaling to launch in two Availability Zones.

3. Launch a multi-AZ MySQL Amazon RDS instance in one availability zone only.

4. Import the data into Amazon RDS from the latest MySQL backup.

5. Create an ELB to front your web servers.

6. Use Amazon Route 53 and create an A record pointing to the elastic load balancer.

* ​

1. Launch two NGINX EC2 instances in two Availability Zones.

2. Copy the web files from the on-premises web server to each Amazon EC2 web server, using Amazon S3 as the repository.

3. Migrate the database using the AWS Database Migration Service.

4. Create an ELB to front your web servers.

5. Use Route 53 and create an alias A recordpointing to the ELB.

**(Correct)**

* ​

1. Export web files to an Amazon S3 bucket in one Availability Zone using AWS Migration Hub.

2. Run the website directly out of Amazon S3.

3. Migrate the database using the AWS Database Migration Service and AWS Schema Conversion Tool (AWS SCT).

4. Use Route 53 and create an alias record pointing to the ELB.

**Explanation**

This is a trick question that contains a lot of information to confuse you, especially if you don't know the fundamental concepts in AWS. All options seem to be correct except for their last steps in setting up Route 53.

To route domain traffic to an ELB load balancer, use Amazon Route 53 to create an alias record that points to your load balancer. An alias record is a Route 53 extension to DNS. It's similar to a CNAME record, but you can create an alias record both for the root domain, such as example.com, and for subdomains, such as www.example.com. (You can create CNAME records only for subdomains). For EC2 instances, always use a Type A Record without an Alias. For ELB, Cloudfront and S3, always use a Type A Record with an Alias and finally, for RDS, always use the CNAME Record with no Alias.

Hence, the following option is the correct answer:

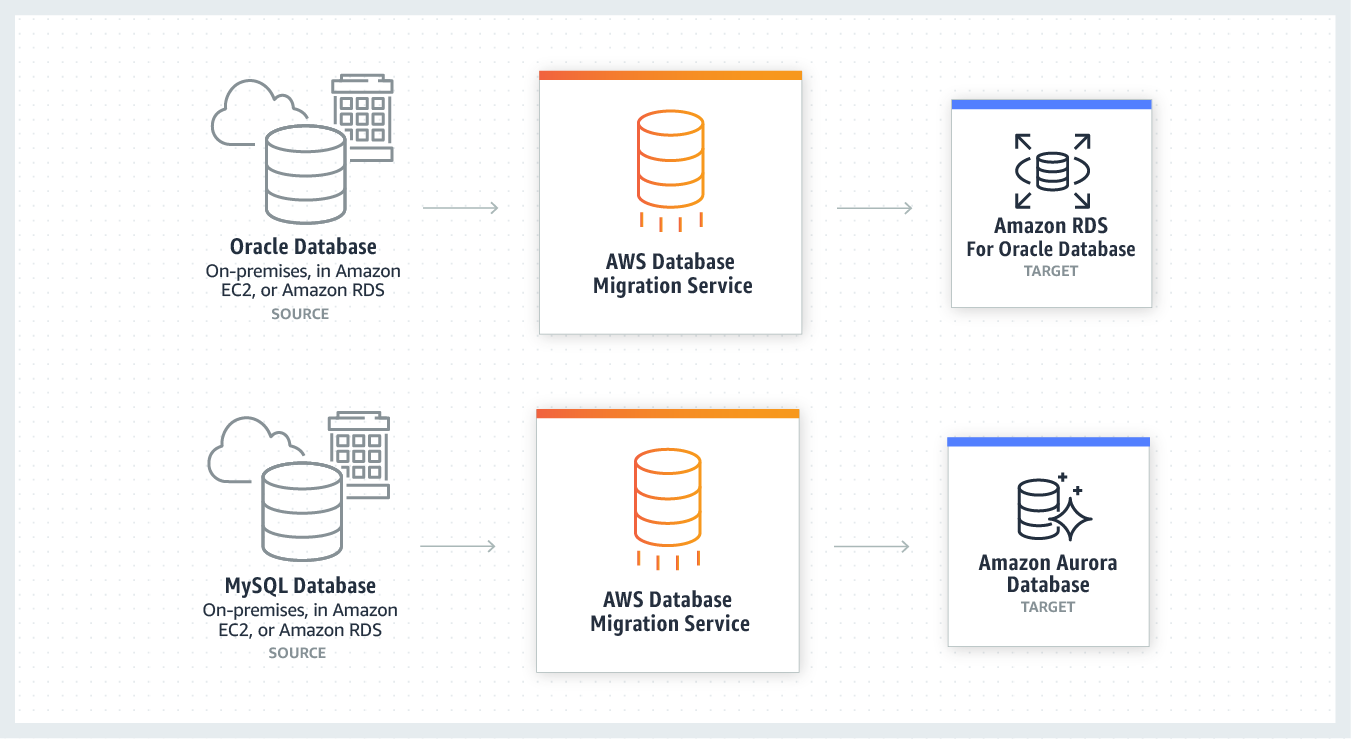
**1. Launch two NGINX EC2 instances in two Availability Zones.**

**2. Copy the web files from the on-premises web server to each Amazon EC2 web server, using Amazon S3 as the repository.**

**3. Migrate the database using the AWS Database Migration Service.**

**4. Create an ELB to front your web servers.**

**5. Use Route 53 and create an alias A record pointing to the ELB.**



**AWS Database Migration Service** helps you migrate databases to AWS quickly and securely. The source database remains fully operational during the migration, minimizing downtime to applications that rely on the database. The AWS Database Migration Service can migrate your data to and from the most widely used commercial and open-source databases.

The following sets of options are incorrect because they are just using an A record without an Alias:

**1. Use the AWS Server Migration Service (SMS) to create an EC2 AMI of the NGINX web server.**

**2. Configure auto-scaling to launch in two Availability Zones.**

**3. Launch a multi-AZ MySQL Amazon RDS instance in one availability zone only.**

**4. Import the data into Amazon RDS from the latest MySQL backup.**

**5. Create an ELB to front your web servers.**

**6. Use Amazon Route 53 and create an A record pointing to the elastic load balancer.**

*--*

**1. Use the AWS Application Discovery Service to migrate the NGINX web server.**

**2. Configure Auto Scaling to launch two web servers in two Availability Zones.**

**3. Launch a Multi-AZ MySQL Amazon Relational Database Service (RDS) instance in one Availability Zone only.**

**4. Import the data into Amazon RDS from the latest MySQL backup.**

**5. Use Amazon Route 53 to create a private hosted zone and point a non-alias A record to the ELB.**

Take note as well that the AWS Server Migration Service (SMS) is primarily used to migrate virtual machines only, which can be from VMware vSphere and Windows Hyper-V to your AWS cloud. In addition, the AWS Application Discovery Service simply helps you to plan migration projects by gathering information about your on-premises data centers but this service is not a suitable migration service.

The following option is also incorrect because the web system that is being migrated is a non-static (dynamic) website, which cannot be hosted in S3:

**1. Export web files to an Amazon S3 bucket in one Availability Zone using AWS Migration Hub.**

**2. Run the website directly out of Amazon S3.**

**3. Migrate the database using the AWS Database Migration Service and AWS Schema Conversion Tool (AWS SCT).**

**4. Use Route 53 and create an alias record pointing to the ELB.**

**References:**

<https://aws.amazon.com/cloud-migration/>

<https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/routing-to-elb-load-balancer.html>

**Check out this Amazon Route 53 Cheat Sheet:**

[https://tutorialsdojo.com/amazon-route-53/](https://tutorialsdojo.com/amazon-route-53/?src=udemy)

**Check out this AWS Database Migration Service Cheat Sheet:**

[https://tutorialsdojo.com/aws-database-migration-service/](https://tutorialsdojo.com/aws-database-migration-service/?src=udemy)

Bottom of Form

Top of Form

Question 56: Skipped

A leading financial company is planning to launch its MERN (MongoDB, Express, React, Node.js) application with an Amazon RDS MariaDB database to serve its clients worldwide. The application will run on both on-premises servers as well as Reserved EC2 instances. To comply with the company's strict security policy, the database credentials must be encrypted both at rest and in transit. These credentials will be used by the application servers to connect to the database. The Solutions Architect is tasked to manage all of the aspects of the application architecture and production deployment.

How should the Architect automate the deployment process of the application in the MOST secure manner?

* ​

Upload the database credentials with a Secure String data type in AWS Systems Manager Parameter Store. Set up a new IAM role with an attached policy that enables access and decryption of the database credentials then associate this roleto all on-premises servers and EC2 instances. Deploy the application packages to the EC2 instances and on-premises servers using AWS CodeDeploy.

**(Correct)**

* ​

Upload the database credentials with a Secure String data type in AWS Systems Manager Parameter Store. Set up a new IAM role that enables access and decryption of the database credentials then associate this roleto all on-premises servers and EC2 instances. Use Elastic Beanstalk to host and manage the application on both on-premises servers and EC2 instances. Deploy the succeeding application revisions to AWS and on-premises servers using Elastic Beanstalk.

* ​

Upload the database credentials with a Secure String data type in AWS Systems Manager Parameter Store. Set up a new IAM policy that enables access and decryption of the database credentials then attach this IAM policy to the instance profile for CodeDeploy-managed EC2 instances. Associate the same policy as well to the on-premises instances. Using AWS CodeDeploy, launch the application packages to the Amazon EC2 instances and on-premises servers.

* ​

Upload the database credentials with key rotation in AWS Secrets Manager. Set up a new IAM role that enables access and decryption of the database credentials then associate this role to all on-premises servers and EC2 instances. Use Elastic Beanstalk to host and manage the application on both on-premises servers and EC2 instances. Deploy the succeeding application revisions to AWS and on-premises servers using Elastic Beanstalk.

**Explanation**

**AWS Systems Manager Parameter Store** provides secure, hierarchical storage for configuration data management and secrets management. You can store data such as passwords, database strings, and license codes as parameter values. You can store values as plain text or encrypted data. You can then reference values by using the unique name that you specified when you created the parameter. Highly scalable, available, and durable, Parameter Store is backed by the AWS Cloud.

Servers and virtual machines (VMs) in a hybrid environment require an IAM role to communicate with the Systems Manager service. The role grants AssumeRole trust to the Systems Manager service. You only need to create the service role for a hybrid environment once for each AWS account.

Users in your company or organization who will use Systems Manager on your hybrid machines must be granted permission in IAM to call the SSM API.



Hence, the correct answer is: **Upload the database credentials with a Secure String data type in AWS Systems Manager Parameter Store. Set up a new IAM role with an attached policy that enables access and decryption of the database credentials then associate this role to all on-premises servers and EC2 instances. Deploy the application packages to the EC2 instances and on-premises servers using AWS CodeDeploy*.***

The option that says: **Upload the database credentials with a Secure String data type in AWS Systems Manager Parameter Store. Set up a new IAM role that enables access and decryption of the database credentials then associate this role to all on-premises servers and EC2 instances. Use Elastic Beanstalk to host and manage the application on both on-premises servers and EC2 instances. Deploy the succeeding application revisions to AWS and on-premises servers using Elastic Beanstalk**is incorrect because you can't deploy an application to your on-premises servers using Elastic Beanstalk. This is only applicable to your Amazon EC2 instances.

The option that says: **Upload the database credentials with a Secure String data type in AWS Systems Manager Parameter Store. Set up a new IAM policy that enables access and decryption of the database credentials then attach this IAM policy to the instance profile for CodeDeploy-managed EC2 instances. Associate the same policy as well to the on-premises instances. Using AWS CodeDeploy, launch the application packages to the Amazon EC2 instances and on-premises servers** is incorrect because you have to use an IAM Role and not an IAM Policy to grant access to AWS Systems Manager Parameter Store.

The option that says: **Upload the database credentials with key rotation in AWS Secrets Manager. Set up a new IAM role that enables access and decryption of the database credentials then associate this role to all on-premises servers and EC2 instances. Use Elastic Beanstalk to host and manage the application on both on-premises servers and EC2 instances. Deploy the succeeding application revisions to AWS and on-premises servers using Elastic Beanstalk** is incorrect because although you can store the database credentials to AWS Secrets Manager, you still can't deploy an application to your on-premises servers using Elastic Beanstalk.

**References:**

<https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-parameter-store.html>

<https://docs.aws.amazon.com/systems-manager/latest/userguide/sysman-service-role.html>

<https://docs.aws.amazon.com/systems-manager/latest/userguide/systems-manager-managedinstances.html>

**Check out this AWS Systems Manager Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-systems-manager/>

Bottom of Form

Top of Form

Question 57: Skipped

You are working as a Solutions Architect for a leading media company. There is a requirement to copy information to or from the shared resource in the other AWS account. You have to provide the other account access to several AWS resources such as S3, KMS and Amazon ES, in the form of a list of AWS account ID numbers. In addition, the user in the other account should still work in the trusted account and there is no need to give up his or her user permissions in place of the role permissions. You must also set up a solution that continuously assess, audit, and monitor the policy configurations.

Which of the following is the MOST suitable type of policy that you should use in this scenario?

* ​

Set up a service-linked role with a service control policy. Use AWS Systems Manager rules to periodically audit changes to the IAM policy and monitor the compliance of the configuration.

* ​

Set up a service-linked role with an identity-based policy. Use AWS Systems Manager rules to periodically audit changes to the IAM policy and monitor the compliance of the configuration.

* ​

Set up cross-account access with a user-based policy configuration. Use AWS Config rules to periodically audit changes to the IAM policy and monitor the compliance of the configuration.

* ​

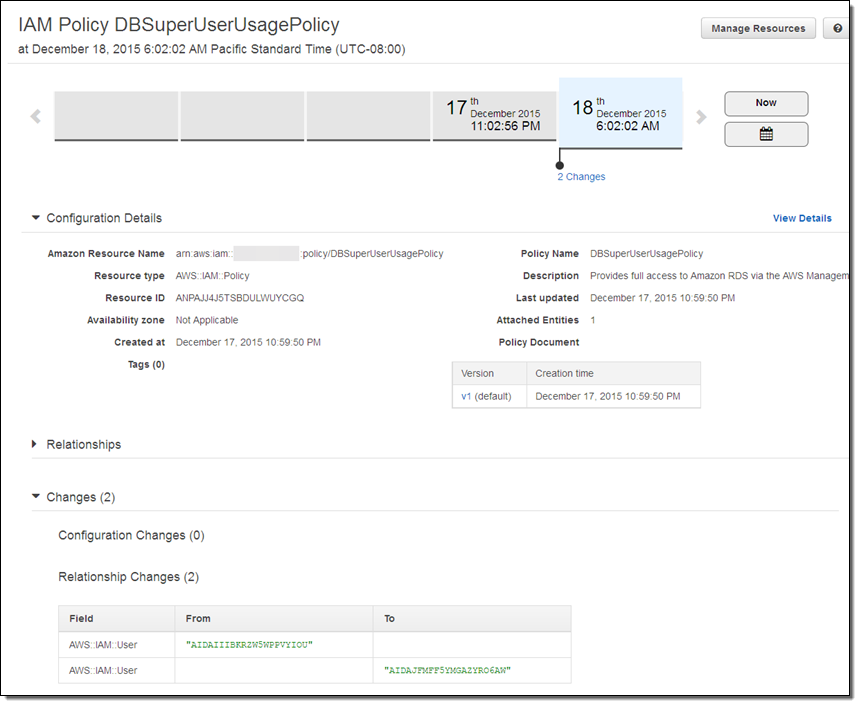
Set up cross-account access with a resource-based Policy. Use AWS Config rules to periodically audit changes to the IAM policy and monitor the compliance of the configuration.

**(Correct)**

**Explanation**

For some AWS services, you can grant cross-account access to your resources. To do this, you attach a policy directly to the resource that you want to share, instead of using a role as a proxy. The resource that you want to share must support resource-based policies. Unlike a user-based policy, a resource-based policy specifies who (in the form of a list of AWS account ID numbers) can access that resource.

Cross-account access with a resource-based policy has some advantages over a role. With a resource that is accessed through a resource-based policy, the user still works in the trusted account and does not have to give up his or her user permissions in place of the role permissions. In other words, the user continues to have access to resources in the trusted account at the same time as he or she has access to the resource in the trusting account. This is useful for tasks such as copying information to or from the shared resource in the other account.



AWS Config is a service that enables you to assess, audit, and evaluate the configurations of your AWS resources. Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations.

Hence, the option that says: **Set up cross-account access with a resource-based Policy. Use AWS Config rules to periodically audit changes to the IAM policy and monitor the compliance of the configuration** is correct.

The option that says: **Set up cross-account access with a user-based policy. configuration. Use AWS Config rules to periodically audit changes to the IAM policy and monitor the compliance of the configuration** is incorrect because a user-based policy maps the access to a certain IAM user and not to a certain AWS resource.

The option that says: **Set up a service-linked role with an identity-based policy. Use AWS Systems Manager rules to periodically audit changes to the IAM policy and monitor the compliance of the configuration** is incorrect because a service-linked role is just a unique type of IAM role that is linked directly to an AWS service. In addition, it is the AWS Config service, and not the AWS Systems Manager, that enables you to assess, audit, and evaluate the configurations of your AWS resources.

The option that says: **Set up a service-linked role with a service control policy. Use AWS Systems Manager rules to periodically audit changes to the IAM policy and monitor the compliance of the configuration** is incorrect because a service control policy is primarily used in AWS Organizations and not for cross-account access. Service-linked roles are predefined by the service and include all the permissions that the service requires to call other AWS services on your behalf. This is not suitable for providing access to your resources to other AWS accounts, unlike cross-account access. You should also use AWS Config, and not AWS Systems Manager, to periodically audit changes to the IAM policy.

**References:**

<https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_compare-resource-policies.html>

<https://aws.amazon.com/config/>

**Check out this AWS Config Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-config/>

Bottom of Form

Top of Form

Question 58: Skipped

**An online immigration system is currently hosted on one large EC2 instance with EBS volumes to store all of the applicants' data. The registration system accepts the information from the user including documents and photos and then performs automated verification and processing to check if the applicant is eligible for immigration. The immigration system becomes unavailable at times when there is a surge of applicants using the system. The existing architecture needs improvement as it takes a long time for the system to complete the processing and the attached EBS volumes are not enough to store the ever-growing data being uploaded by the users. Which of the following is the best option to achieve high availability and a more scalable data storage?**

* ​

Use a SNS to distribute the tasks to a group of EC2 instances. Use Auto Scaling to dynamically increase or decrease the group of EC2 instances depending on the length of the SQS queue.

* ​

Upgrade your architecture to use an S3 bucket with cross-region replication (CRR) enabled, as the storage service. Set up an SQS queue to distribute the tasks to a group of EC2 instances with Auto Scaling to dynamically increase or decrease the group of EC2 instances depending on the length of the SQS queue. Use CloudFormation to replicate your architecture to another region.

**(Correct)**

* ​

Use EBS with Provisioned IOPS to store files, SNS to distribute tasks to a group of EC2 instances working in parallel, and Auto Scaling to dynamically size the group of EC2 instances depending on the number of SNS notifications. Use CloudFormation to replicate your architecture to another region.

* ​

Upgrade to EBS with Provisioned IOPS as your main storage service and change your architecture to use an SQS queue to distribute the tasks to a group of EC2 instances. Use Auto Scaling to dynamically increase or decrease the group of EC2 instances depending on the length of the SQS queue.

**Explanation**

In this scenario, you need to overhaul the existing immigration service to upgrade its storage and computing capacity. Since EBS Volumes can only provide limited storage capacity and are not scalable, you should use S3 instead. The system goes down at times when there is a surge of requests which indicates that the existing large EC2 instance could not handle the requests any longer. In this case, you should implement a highly-available architecture and a queueing system with SQSa and Auto Scaling.

The option that says: ***Upgrade your architecture to use an S3 bucket with cross-region replication (CRR) enabled, as the storage service. Set up an SQS queue to distribute the tasks to a group of EC2 instances with Auto Scaling to dynamically increase or decrease the group of EC2 instances depending on the length of the SQS queue. Use CloudFormation to replicate your architecture to another region*** is correct because it provides high availability and a scalable data storage with S3. Auto-scaling of EC2 instances reduces the overall processing time and SQS helps in distributing the tasks to a group of EC2 instances.

The option that says: ***Use EBS with Provisioned IOPS to store files, SNS to distribute tasks to a group of EC2 instances working in parallel, and Auto Scaling to dynamically size the group of EC2 instances depending on the number of SNS notifications. Use CloudFormation to replicate your architecture to another region***is incorrect because neither EBS nor SNS are valid choices in this scenario. Using SQS is more suitable in distributing the tasks to an Auto Scaling group of EC2 instances and not SNS.

The option that says: ***Use a SNS to distribute the tasks to a group of EC2 instances. Use Auto Scaling to dynamically increase or decrease the group of EC2 instances depending on the length of the SQS queue. Use a SNS to distribute the tasks to a group of EC2 instances. Use Auto Scaling to dynamically increase or decrease the group of EC2 instances depending on the length of the SQS queue*** is incorrect because SNS is not a valid choice in this scenario.

The option that says: ***Upgrade to EBS with Provisioned IOPS as your main storage service and change your architecture to use an SQS queue to distribute the tasks to a group of EC2 instances. Use Auto Scaling to dynamically increase or decrease the group of EC2 instances depending on the length of the SQS queue*** is incorrect because EBS is not a scalable storage solution. You should use S3 instead.

The option that says: ***Upgrade your architecture to use EC2 instance store as a cost-effective storage solution. Set up an SQS queue to distribute the tasks to a group of EC2 instances with Auto Scaling to dynamically increase or decrease the group of EC2 instances depending on the length of the SQS queue***is incorrect because an EC2 instance store is an ephemeral storage and should never be your choice in storing user data. You should use S3 instead.

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

<https://aws.amazon.com/s3/>

**Check out this Amazon S3 Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-s3/>

Bottom of Form

Top of Form

Question 59: Skipped

A serverless application is using a Lambda function which fetches data from a public REST API as part of its processing. There is a new requirement to configure the function to store the results to a database hosted in a virtual private cloud (VPC) in your account. You have provided the additional VPC-specific configuration information which includes the subnet IDs and security group IDs. However, your function had stopped working and could not complete the processing after your change.

Which of the following should you do to fix this issue? (Select TWO.)

* ​

Ensure that the associated security group of the Lambda function allows outbound connections.

**(Correct)**

* ​

Submit a limit increase request for concurrent executions of your Lambda function.

* ​

Configure Lambda to forward payloads that were not processed to a dead-letter queue (DLQ) using SQS.

* ​

Manually set up elastic network interfaces (ENIs) to enable your function to connect securely to other resources within your private VPC.

* ​

Add a NAT gateway to your VPC.

**(Correct)**

**Explanation**

By default, Lambda runs your functions in a secure VPC with access to AWS services and the Internet. The VPC is owned by Lambda and does not connect to your account's default VPC. When you connect a function to a VPC in your account, it does not have access to the Internet unless your VPC provides access.

AWS Lambda uses the VPC information you provide to set up ENIs that allow your Lambda function to access VPC resources. Each ENI is assigned a private IP address from the IP address range within the subnets you specify, but is not assigned any public IP addresses.

Therefore, if your Lambda function requires Internet access (for example, to access AWS services that don't have VPC endpoints ), you can configure a NAT instance inside your VPC or you can use the Amazon VPC NAT gateway. For more information, see NAT Gateways in the Amazon VPC User Guide. You cannot use an Internet gateway attached to your VPC, since that requires the ENI to have public IP addresses.

If your Lambda function needs Internet access, do not attach it to a public subnet or to a private subnet without Internet access. Instead, attach it only to private subnets with Internet access through a NAT instance or an Amazon VPC NAT gateway. You should also ensure that the associated security group of the Lambda function allows outbound connections. Hence, the correct answers in this scenario are ***ensuring that the associated security group of the Lambda function allows outbound connections****and****adding a NAT gateway to your VPC.***

***Manually setting up elastic network interfaces (ENIs) to enable your function to connect securely to other resources within your private VPC***is incorrect because AWS Lambda uses the VPC information you provide to automatically set up ENIs that allow your Lambda function to access VPC resources. You don't need to do this step in order for your Lambda function to be integrated with your VPC.

***Submitting a limit increase request for concurrent executions of your Lambda function***is incorrect because submitting a limit increase request for concurrent executions will not solve this problem since the issue here is that the function cannot connect to public REST APIs over the Internet anymore.

***Configuring Lambda to forward payloads that were not processed to a dead-letter queue (DLQ) using SQS***is incorrect because configuring your Lambda function to forward payloads that were not processed to a dead-letter queue (DLQ) will only improve the error handling of your serverless application. The issue here is the Internet connectivity of your Lambda function and not its error handling hence, this option will not solve the problem in this scenario.

**References:**

<https://docs.aws.amazon.com/lambda/latest/dg/vpc.html>

<https://aws.amazon.com/premiumsupport/knowledge-center/internet-access-lambda-function/>

**Check out this AWS Lambda Cheat Sheet:**

[https://tutorialsdojo.com/aws-lambda/](https://tutorialsdojo.com/aws-lambda/?src=udemy)

**AWS Knowledge Center Videos: How do I give internet access to my Lambda function in a VPC?**

<https://www.youtube.com/watch?v=JcRKdEP94jM&feature=youtu.be>

Bottom of Form

Top of Form

Question 60: Skipped

A graphics design startup is using multiple Amazon S3 buckets to store high-resolution media files for their various digital artworks. After securing a partnership deal with a leading media company, the two parties shall be sharing digital resources with one another as part of the contract. The media company frequently performs multiple object retrievals from the S3 buckets every day, which increased the startup's data transfer costs.

As the Solutions Architect, what should you do to help the startup lower their operational costs?

* ​

Create a new billing account for the social media company by using AWS Organizations. Apply SCPs on the organization to ensure that each account has access only to its own resources and each other's S3 buckets.

* ​

Advise the media company to create their own S3 bucket. Then run the aws s3 sync s3://sourcebucket s3://destinationbucket command to copy the objects from their S3 bucket to the other party's S3 bucket. In this way, future retrievals can be made on the media company's S3 bucket instead.

* ​

Provide cross-account access for the media company, which has permissions to access contents in the S3 bucket. Cross-account retrieval of S3 objects is charged to the account that made the request.

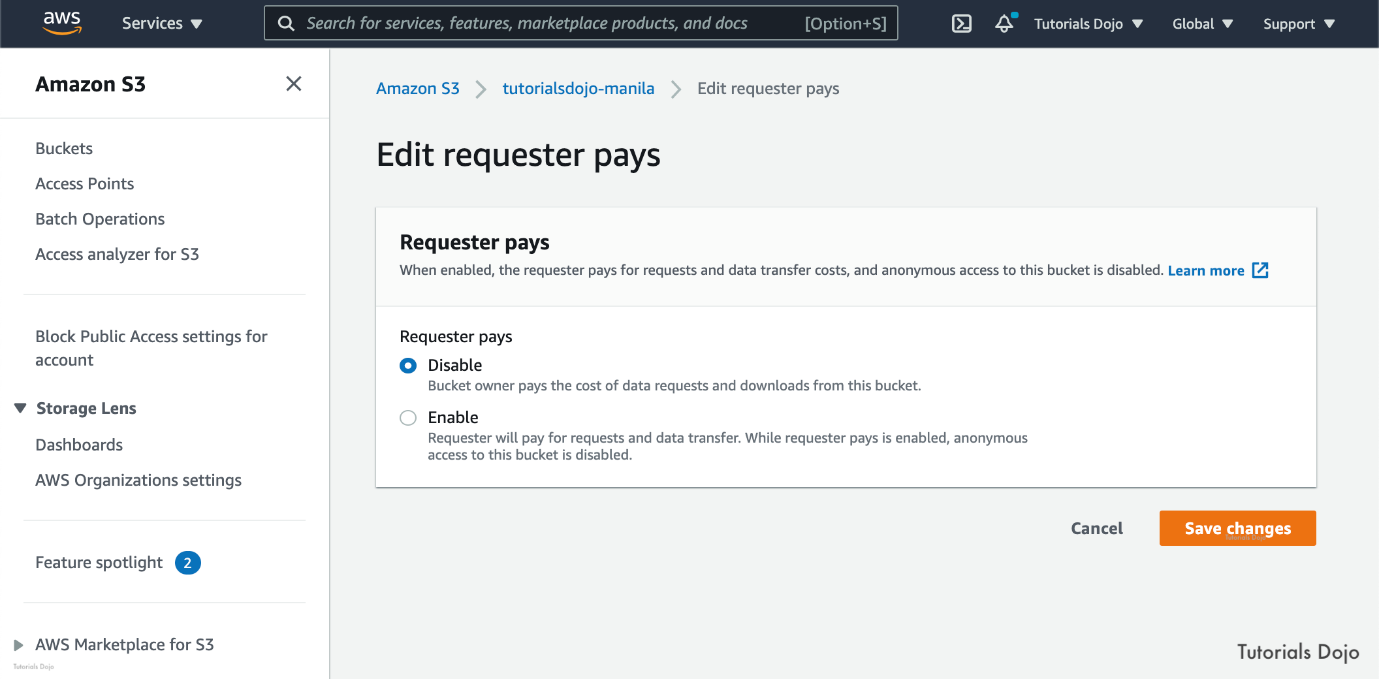
* ​

Enable the Requester Pays feature in all of the startup's S3 buckets to make the media company pay the cost of the data transfer from the buckets.

**(Correct)**

**Explanation**

In general, bucket owners pay for all Amazon S3 storage and data transfer costs associated with their bucket. A bucket owner, however, can configure a bucket to be a **Requester Pays** bucket. With Requester Pays buckets, the requester instead of the bucket owner pays the cost of the request and the data download from the bucket. The bucket owner always pays the cost of storing data.



You must authenticate all requests involving Requester Pays buckets. The request authentication enables Amazon S3 to identify and charge the requester for their use of the Requester Pays bucket. After you configure a bucket to be a Requester Pays bucket, requesters must include x-amz-request-payer in their requests either in the header, for POST, GET and HEAD requests, or as a parameter in a REST request to show that they understand that they will be charged for the request and the data download.

Hence, the correct answer is to **enable the Requester Pays feature in all of the startup's S3 buckets to make the media company pay the cost of the data transfer from the buckets.**

The option that says: **Advise the media company to create their own S3 bucket. Then run the aws s3 sync s3://sourcebucket s3://destinationbucket command to copy the objects from their S3 bucket to the other party's S3 bucket. In this way, future retrievals can be made on the media company's S3 bucket instead** is incorrect because sharing all the assets of the startup to the media entails a lot of costs considering that you will be charged for the data transfer charges made during the sync process.

**Creating a new billing account for the social media company by using AWS Organizations, then applying SCPs on the organization to ensure that each account has access only to its own resources and each other's S3 buckets** is incorrect because AWS Organizations does not create a separate billing account for every account under it. Instead, what AWS Organizations has is consolidated billing. You can use the consolidated billing feature in AWS Organizations to consolidate billing and payment for multiple AWS accounts. Every organization in AWS Organizations has a master account that pays the charges of all the member accounts.

The option that says: **Provide cross-account access for the media company, which has permissions to access contents in the S3 bucket. Cross-account retrieval of S3 objects is charged to the account that made the request** is incorrect because cross-account access does not shoulder the charges that are made during S3 object requests. Unless Requester Pays is enabled on the bucket, the bucket owner is still the one that is charged.

**Reference:**

<https://docs.aws.amazon.com/AmazonS3/latest/dev/RequesterPaysBuckets.html>

**Check out this Amazon S3 Cheat Sheet:**

[https://tutorialsdojo.com/amazon-s3/](https://tutorialsdojo.com/amazon-s3/?src=udemy)

Bottom of Form

Top of Form

Question 61: Skipped

A company that provides cloud-based accounting and management solutions to various clients recently identified a series of infrastructure layer attacks to their VPC such as SYN floods, UDP floods, and other DDoS attacks. As the Solutions Architect, you are required to fortify the security of their current cloud infrastructure to protect the data of your clients and ensure that you will get an immediate notification for any ongoing DDoS attacks via CloudWatch. You also have to set up a system that will continuously monitor and record your AWS resource configurations and will allow you to automate the evaluation of recorded configurations against the desired configurations of your company.

Which of the following services should you use to meet the above requirement? (Select TWO.)

* ​

AWS Firewall Manager

* ​

AWS Systems Manager

* ​

AWS Config

**(Correct)**

* ​

AWS Shield Advanced

**(Correct)**

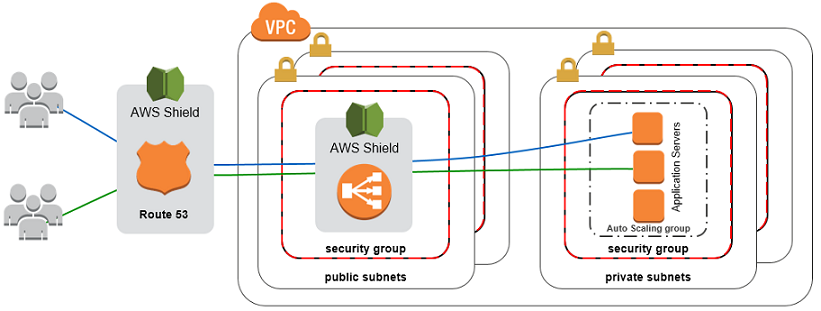
* ​

AWS WAF

**Explanation**

In this scenario, it is best to use ***AWS Shield Advanced***and ***AWS Config*** to protect your cloud infrastructure against DDoS attacks, as well as AWS Config which continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations. Hence, ***AWS Shield Advanced*** and ***AWS Config*** are the correct answers.

For higher levels of protection against attacks targeting your applications running on Amazon Elastic Compute Cloud (EC2), Elastic Load Balancing(ELB), Amazon CloudFront, and Amazon Route 53 resources, you can subscribe to AWS Shield Advanced. In addition to the network and transport layer protections that come with Standard, AWS Shield Advanced provides additional detection and mitigation against large and sophisticated DDoS attacks, near real-time visibility into attacks, and integration with AWS WAF, a web application firewall. AWS Shield Advanced also gives you 24x7 access to the AWS DDoS Response Team (DRT) and protection against DDoS related spikes in your Amazon Elastic Compute Cloud (EC2), Elastic Load Balancing(ELB), Amazon CloudFront, and Amazon Route 53 charges.



AWS Config is a service that enables you to assess, audit, and evaluate the configurations of your AWS resources. Config continuously monitors and records your AWS resource configurations and allows you to automate the evaluation of recorded configurations against desired configurations. With Config, you can review changes in configurations and relationships between AWS resources, dive into detailed resource configuration histories, and determine your overall compliance against the configurations specified in your internal guidelines. This enables you to simplify compliance auditing, security analysis, change management, and operational troubleshooting.

***AWS Firewall Manager*** is incorrect because it is mainly used to simplify your AWS WAF administration and maintenance tasks across multiple accounts and resources. It does not protect your VPC against DDoS attacks.

***AWS WAF*** is incorrect because even though it can help you block common attack patterns to your VPC such as SQL injection or cross-site scripting, this is still not enough to withstand DDoS attacks. It is better to use AWS Shield in this scenario.

***AWS Systems Manager*** is incorrect because it is just a management service that helps you automatically collect software inventory, apply OS patches, and create system images, but this does not protect your VPC against DDoS attacks.

**References:**

<https://docs.aws.amazon.com/waf/latest/developerguide/what-is-aws-waf.html>

<https://aws.amazon.com/config/>

<https://aws.amazon.com/shield/>

**Check out these AWS Shield and AWS Config Cheat Sheets:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-shield/>

<https://tutorialsdojo.com/aws-cheat-sheet-aws-config/>

Bottom of Form

Top of Form

Question 62: Skipped

**A well-funded startup company is building a mobile app that showcases their latest fashion accessories and gadgets. The marketing manager hired a famous model with millions of Instagram followers to promote their new products and hence, it is expected that the app will be a huge hit once it is launched in the market. It must have the ability to automatically scale to handle millions of views of its static contents and to allow users to store their own photos of themselves wearing fashionable accessories with a maximum of 100-character caption. In this scenario, which of the following would fulfill this requirement?**

* ​

1. Use Cognito to handle the user authentication and management.

2. Use an RDS database to store user data.

3. Create an S3 bucket to store all of the user photos and other static files.

4. Distribute the static contents using CloudFront to improve scalability.

* ​

1. Use Cognito to handle the user authentication and management.

2. Launch a DynamoDB table to store user data.

3. Create an S3 bucket to store all of the user photos and other static files.

4. Distribute the static contents using CloudFront to improve scalability.

**(Correct)**

* ​

1. Configure an on-premises Active Directory (AD) server utilizing SAML 2.0 to manage the application users inside of the on-premises AD server.

2. Develop a custom code that authenticates against the LDAP server.

3. Use DynamoDB as the main database of the app and S3 as the scalable object storage.

4. Grant an IAM role assigned to the STS token to allow the end-user to access the required data in the DynamoDB table.

5. Distribute the static contents using CloudFront to improve scalability.

* ​

1. Set up a SAML 2.0-based Federation that let the users sign into the app using a third party identity provider such as Amazon, Google or Facebook.

2. Set up an RDS database and an S3 bucket to store the photos.

3. Use the AssumeRoleWithWebIdentity API call to assume the IAM role containing the proper permissions to communicate with the RDS database.

4. Distribute the static contents using S3 to improve scalability.

**Explanation**

In this scenario, the top priority is scalability to meet the upcoming surge of users of your mobile app. Since all of these services can provide the scalability that your mobile app needs, the best option that you can choose is the following:

***1. Use Cognito to handle the user authentication and management.***

***2. Launch a DynamoDB table to store user data.***

***3. Create an S3 bucket to store all of the user photos and other static files.***

***4. Distribute the static contents using CloudFront to improve scalability.***

Amazon Cognito scales to millions of users and supports sign-in with social identity providers, such as Facebook, Google, and Amazon, and enterprise identity providers via SAML 2.0. S3 provides a scalable object storage solution. CloudFront is a Content Delivery Network that helps your system to handle the millions of user views and finally, DynamoDB is a scalable database that can handle millions of records.

The following option is incorrect because a SAML 2.0-based federation doesn't use social media logins and it is better to use CloudFront to distribute static contents compared to an S3 bucket:

***1. Set up a SAML 2.0-based Federation that let the users sign into the app using a third party identity provider such as Amazon, Google or Facebook.***

***2. Set up an RDS database and an S3 bucket to store the photos.***

***3. Use the AssumeRoleWithWebIdentity API call to assume the IAM role containing the proper permissions to communicate with the RDS database.***

***4. Distribute the static contents using S3 to improve scalability.***

The following option is incorrect because using an on-premise Active Directory server is not a suitable solution for this scenario:

***1. Configure an on-premises Active Directory (AD) server utilizing SAML 2.0 to manage the application users inside of the on-premises AD server.***

***2. Develop a custom code that authenticates against the LDAP server.***

***3. Use DynamoDB as the main database of the app and S3 as the scalable object storage.***

***4. Grant an IAM role assigned to the STS token to allow the end-user to access the required data in the DynamoDB table.***

***5. Distribute the static contents using CloudFront to improve scalability.***

Remember that this is a public mobile app and hence, it is better to set up a Web Identity Federation instead

The following option is incorrect because RDS is not as scalable than DynamoDB:

***1. Use Cognito to handle the user authentication and management.***

***2. Use an RDS database to store user data.***

***3. Create an S3 bucket to store all of the user photos and other static files.***

***4. Distribute the static contents using CloudFront to improve scalability.***

Based on the type of data you will be storing, a relational database like RDS is not suitable to store simple data sets such as a 100-character caption which does not need multiple tables or relational data models.

**Reference:**

<https://aws.amazon.com/cognito/>

Bottom of Form

Top of Form

Question 63: Skipped

A logistics company plans to host its web application on AWS to allow customers to track their shipping worldwide. The web application will have a multi-tier setup – Amazon EC2 instances for running the web and application layer, Amazon S3 bucket for hosting the static content, and a NoSQL database. The company plans to provision the resources in the us-east-1 region. The company also wants to have a second site hosted on us-west-1 region for disaster recovery. The second site must have the same copy of data from the primary site and the failover should be as quick as possible when the primary region becomes unavailable. Failing back to the primary region should be done automatically once it becomes available again.

Which of the following solutions should the Solutions Architect implement to meet the company requirements?

* ​

Create the same resources of Auto Scaling group of EC2 instances for web and application tiers on both regions using AWS CloudFormation StackSets. Enable Amazon S3 cross-Region on the S3 bucket to asynchronously replicate the contents to the secondary region. Create Amazon Route 53 DNS zone entries with a failover routing policy and set the us-west-1 region as the secondary site. For the database tier, create a DynamoDB global table spanning both regions.

**(Correct)**

* ​

Provision the same Auto Scaling group of EC2 instances for web and application tiers in both regions using AWS Service Catalog. Enable Amazon S3 cross-Region on the S3 bucket to asynchronously replicate the contents to the secondary region. Ensure that Amazon Route 53 health check is enabled on the primary region and update the public DNS zone entry with the secondary region in case of an outage. For the database tier, create an Amazon RDS for MySQL and enable cross-region replication to create a read-replica on the secondary region.

* ​

Create the same resources of Auto Scaling group of EC2 instances for web and application tiers on both regions using AWS CloudFormation StackSets. Enable Amazon S3 cross-Region on the S3 bucket to asynchronously replicate the contents to the secondary region. Create an Amazon CloudFront distribution. Set the S3 bucket as the origin for static files and multi-origins for the web and application tiers. For the database tier, create an Amazon DynamoDB table in each region and regularly backup to an Amazon S3 bucket.

* ​

Create the same resources of Auto Scaling group of EC2 instances for web and application tiers on both regions using AWS CloudFormation StackSets. Enable Amazon S3 cross-Region on the S3 bucket to asynchronously replicate the contents to the secondary region. Create Amazon Route 53 DNS zone entries with a failover routing policy and set the us-west-1 region as the secondary site. For the database tier, create an Amazon Aurora global database spanning the two regions.

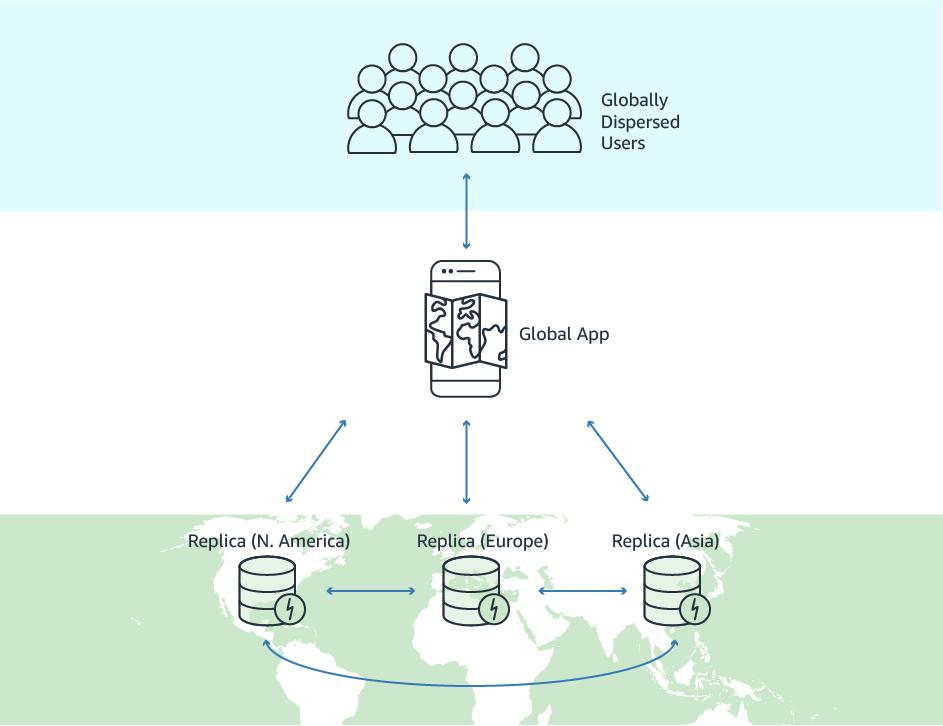
**Explanation**

**AWS CloudFormation** helps AWS customers implement an Infrastructure as Code model. Instead of setting up their environments and applications by hand, they build a template and use it to create all of the necessary resources, collectively known as a CloudFormation stack. This model removes opportunities for manual error, increases efficiency, and ensures consistent configurations over time.

With **Amazon CloudFront StackSets** you can define an AWS resource configuration in a CloudFormation template and then roll it out across multiple AWS accounts and/or Regions with a couple of clicks. You can use this to set up a baseline level of AWS functionality that addresses the cross-account and cross-region scenarios. Once you have set this up, you can easily expand coverage to additional accounts and regions.

Amazon S3 Replication enables automatic, asynchronous copying of objects across Amazon S3 buckets. Buckets that are configured for object replication can be owned by the same AWS account or by different accounts. The objects may be replicated to a single destination bucket or multiple destination buckets. Destination buckets can be in different AWS Regions or within the same Region as the source bucket. **Amazon S3 Cross-Region Replication (CRR)** is used to copy objects across Amazon S3 buckets in different AWS Regions. CRR is helpful if you want to meet compliance requirements such as the need to have a copy of your data on another location.

**Amazon DynamoDB global tables** provide you with a fully managed, multi-region and multi-active database that delivers fast, local, read and write performance for massively scaled, global applications. Global tables replicate your DynamoDB tables automatically across your choice of AWS Regions. Global tables eliminate the difficult work of replicating data between Regions and resolving update conflicts, enabling you to focus on your application's business logic. In addition, global tables enable your applications to stay highly available even in the unlikely event of isolation or degradation of an entire Region.



Therefore, the correct answer is: **Create the same resources of Auto Scaling group of EC2 instances for web and application tiers on both regions using AWS CloudFormation StackSets. Enable Amazon S3 cross-Region on the S3 bucket to asynchronously replicate the contents to the secondary region. Create Amazon Route 53 DNS zone entries with a failover routing policy and set the us-west-1 region as the secondary site. For the database tier, create a DynamoDB global table spanning both regions.**

The option that says: **Create the same resources of Auto Scaling group of EC2 instances for web and application tiers on both regions using AWS CloudFormation StackSets. Enable Amazon S3 cross-Region on the S3 bucket to asynchronously replicate the contents to the secondary region. Create Amazon Route 53 DNS zone entries with a failover routing policy and set the us-west-1 region as the secondary site. For the database tier, create an Amazon Aurora global database spanning the two regions** is incorrect. The application is designed for a NoSQL database so a DynamoDB global table is recommended for this, not an Amazon Aurora global database.

The option that says: **Provision the same Auto Scaling group of EC2 instances for web and application tiers in both regions using AWS Service Catalog. Enable Amazon S3 cross-Region on the S3 bucket to asynchronously replicate the contents to the secondary region. Ensure that Amazon Route 53 health check is enabled on the primary region and update the public DNS zone entry with the secondary region in case of an outage. For the database tier, create an Amazon RDS for MySQL and enable cross-region replication to create a read-replica on the secondary region** is incorrect. You don’t have to manually update the public DNS zone entry with the secondary region, you just have to configure the failover routing policy in Route 53 to automatically failover to the secondary site and vice versa. MySQL is not recommended as the application is designed for a NoSQL database.

The option that says: **Create the same resources of Auto Scaling group of EC2 instances for web and application tiers on both regions using AWS CloudFormation StackSets. Enable Amazon S3 cross-Region on the S3 bucket to asynchronously replicate the contents to the secondary region. Create an Amazon CloudFront distribution. Set the S3 bucket as the origin for static files and multi-origins for the web and application tiers. For the database tier, create an Amazon DynamoDB table in each region and regularly backup to an Amazon S3 bucket** is incorrect. You can’t reliably sync DynamoDB tables on two regions with just backups from S3. There will be a delay on the backup and restore. You should use DynamoDB global tables instead.

**References:**

<https://aws.amazon.com/blogs/aws/use-cloudformation-stacksets-to-provision-resources-across-multiple-aws-accounts-and-regions/>

<https://docs.aws.amazon.com/AmazonS3/latest/dev/replication.html>

<https://aws.amazon.com/dynamodb/global-tables/>

**Check out these Amazon S3, Amazon DynamoDB, and CloudFormation StackSet Cheat Sheets:**

[https://tutorialsdojo.com/amazon-dynamodb/](https://tutorialsdojo.com/amazon-dynamodb/?src=udemy)

[https://tutorialsdojo.com/amazon-s3/](https://tutorialsdojo.com/amazon-s3/?src=udemy)

[https://tutorialsdojo.com/aws-cloudformation-stacksets-and-nested-stacks/](https://tutorialsdojo.com/aws-cloudformation-stacksets-and-nested-stacks/?src=udemy)

Bottom of Form

Top of Form

Question 64: Skipped

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 the ECS tasks then use IAM roles for tasks to access other resources.

* ​

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 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 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.

**(Correct)**

**Explanation**

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.

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.

Hence, the correct answer is: ***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.***

The option that says: ***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***is incorrect because you won't be able to attach security groups to your ECS tasks using this network mode type. This will only use the Docker's built-in virtual network which runs inside each container instance. You have to use the awsvpc network mode instead to allow you to use security groups and network monitoring tools at a more granular level within your tasks. Moreover, if you are using the awsvpc network mode, you should attach the security group to the ECS task and not to the EC2 instance.

The option that says: ***Use the bridge 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*** is incorrect because, in order for you to use security groups and network monitoring tools at a more granular level within your ECS tasks, you have to use the awsvpc network mode instead.

The option that says: ***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*** is incorrect because although it uses the correct network mode, you have to use an IAM Role instead. It is a security risk to pass the IAM credentials into the container as it could be potentially exposed.

**References:**

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task-networking.html>

<https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definition_parameters.html#network_mode>

**Check out this Amazon ECS Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-elastic-container-service-amazon-ecs/>

Bottom of Form

Top of Form

Question 65: Skipped

A leading media company has a hybrid architecture where its on-premises data center is connected to AWS via a Direct Connect connection. They also have a repository of over 50-TB digital videos and media files. These files are stored on their on-premises tape library and are used by their Media Asset Management (MAM) system. Due to the sheer size of their data, they want to implement an automated catalog system that will enable them to search their files using facial recognition. A catalog will store the faces of the people who are present in these videos including a still image of each person. Eventually, the media company would like to migrate these media files to AWS including the MAM video contents.

Which of the following options provides a solution which uses the LEAST amount of ongoing management overhead and will cause MINIMAL disruption to the existing system?

* ​

Use Amazon Kinesis Video Streams to set up a video ingestion stream and with Amazon Rekognition, build a collection of faces. Stream the media files from the MAM solution into Kinesis Video Streams and configure the Amazon Rekognition to process the streamed files. Launch a stream consumer to retrieve the required metadata, and push the metadata into the MAM solution. Finally, configure the stream to store the files in an S3 bucket.

* ​

Set up a tape gateway appliance on-premises and connect it to your AWS Storage Gateway. Configure the MAM solution to fetch the media files from the current archive and push them into the tape gateway in the AWS Cloud. Using Amazon Rekognition, build a collection from the catalog of faces. Utilize a Lambda function which invokes the Rekognition Javascript SDK to have Amazon Rekognition process the video directly from the tape gateway in real-time, retrieve the required metadata, and push the metadata into the MAM solution.

* ​

Integrate the file system of your local data center to AWS Storage Gateway by setting up a file gateway appliance on-premises. Utilize the MAM solution to extract the media files from the current data store and send them into the file gateway. Build a collection using Amazon Rekognition by populating a catalog of faces from the processed media files. Use an AWS Lambda function to invoke Amazon Rekognition Javascript SDK to have it fetch the media file from the S3 bucket which is backing the file gateway, retrieve the needed metadata, and finally, persist the information into the MAM solution.

**(Correct)**

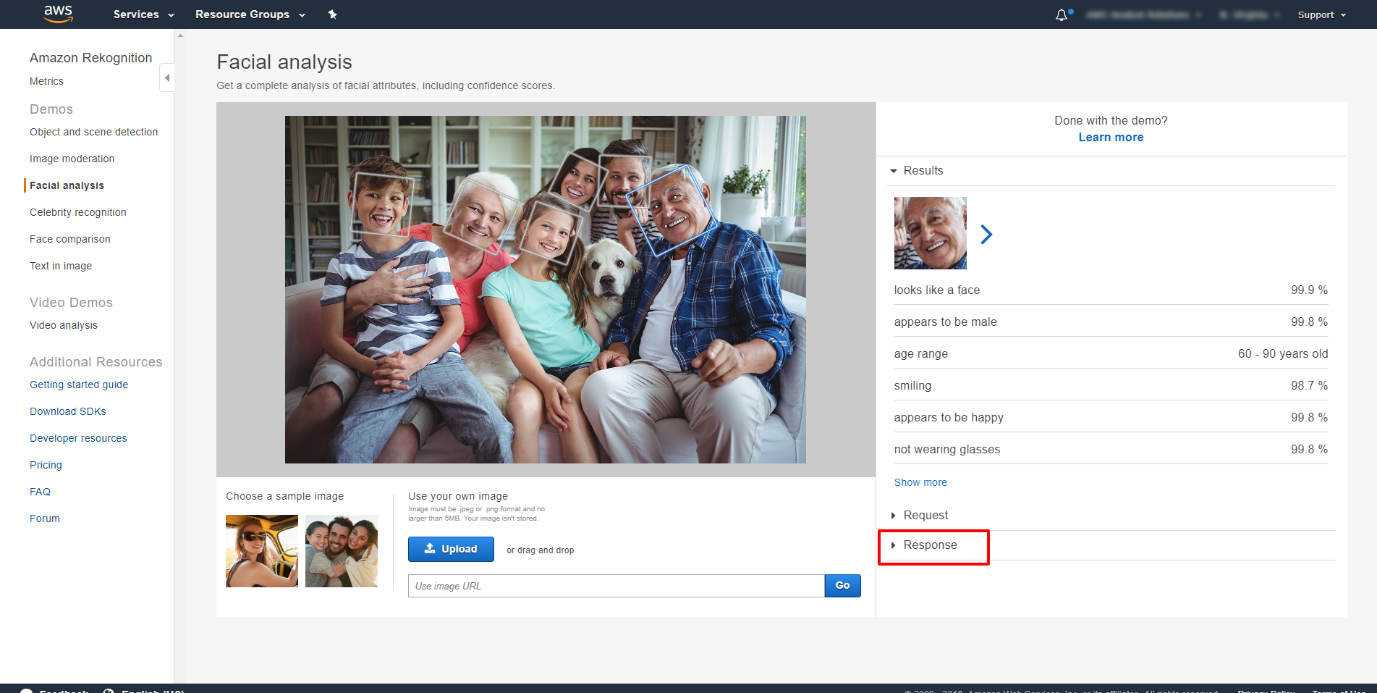
* ​

Migrate all of the media files from the on-premises library into an EBS volume mounted on a large EC2 instance. Install an open-source facial recognition tool in the instance like OpenFace or OpenCV. Process the media files to retrieve the metadata and push this information into the MAM solution. Lastly, copy the media files to an S3 bucket.

**Explanation**

**Amazon Rekognition** can store information about detected faces in server-side containers known as collections. You can use the facial information that's stored in a collection to search for known faces in images, stored videos, and streaming videos. Amazon Rekognition supports the [IndexFaces](https://docs.aws.amazon.com/rekognition/latest/dg/API_IndexFaces.html) operation. You can use this operation to detect faces in an image and persist information about facial features that are detected into a collection. This is an example of a *storage-based* API operation because the service persists information on the server.

To store facial information, you must first create ([CreateCollection](https://docs.aws.amazon.com/rekognition/latest/dg/API_CreateCollection.html)) a face collection in one of the AWS Regions in your account. You specify this face collection when you call the IndexFaces operation. After you create a face collection and store facial feature information for all faces, you can search the collection for face matches. To search for faces in an image, call [SearchFacesByImage](https://docs.aws.amazon.com/rekognition/latest/dg/API_SearchFacesByImage.html). To search for faces in a stored video, call [StartFaceSearch](https://docs.aws.amazon.com/rekognition/latest/dg/API_StartFaceSearch.html). To search for faces in a streaming video, call [CreateStreamProcessor](https://docs.aws.amazon.com/rekognition/latest/dg/API_CreateStreamProcessor.html).



AWS Storage Gateway offers file-based, volume-based, and tape-based storage solutions. With a tape gateway, you can cost-effectively and durably archive backup data in GLACIER or DEEP\_ARCHIVE. A tape gateway provides a virtual tape infrastructure that scales seamlessly with your business needs and eliminates the operational burden of provisioning, scaling, and maintaining a physical tape infrastructure.

You can run AWS Storage Gateway either on-premises as a VM appliance, as a hardware appliance, or in AWS as an Amazon Elastic Compute Cloud (Amazon EC2) instance. You deploy your gateway on an EC2 instance to provision iSCSI storage volumes in AWS. You can use gateways hosted on EC2 instances for disaster recovery, data mirroring, and providing storage for applications hosted on Amazon EC2.

Hence, the correct answer is: ***Integrate the file system of your local data center to AWS Storage Gateway by setting up a file gateway appliance on-premises. Utilize the MAM solution to extract the media files from the current data store and send them into the file gateway. Build a collection using Amazon Rekognition by populating a catalog of faces from the processed media files. Use an AWS Lambda function to invoke Amazon Rekognition Javascript SDK to have it fetch the media file from the S3 bucket which is backing the file gateway, retrieve the needed metadata, and finally, persist the information into the MAM solution.***

The option that says: ***Migrate all of the media files from the on-premises library into an EBS volume mounted on a large EC2 instance. Install an open-source facial recognition tool in the instance like OpenFace or OpenCV. Process the media files to retrieve the metadata and push this information into the MAM solution. Lastly, copy the media files to an S3 bucket*** is incorrect because it entails a lot of ongoing management overhead instead of just using Amazon Rekognition. Moreover, it is more suitable to use the AWS Storage Gateway service rather than an EBS Volume.

The option that says: ***Set up a tape gateway appliance on-premises and connect it to your AWS Storage Gateway. Configure the MAM solution to fetch the media files from the current archive and push them into the tape gateway in the AWS Cloud. Using Amazon Rekognition, build a collection from the catalog of faces. Utilize a Lambda function which invokes the Rekognition Javascript SDK to have Amazon Rekognition process the video directly from the tape gateway in real-time, retrieve the required metadata, and push the metadata into the MAM solution*** is incorrect because although this is using the right combination of AWS Storage Gateway and Amazon Rekognition, take note that you can't directly fetch the media files from your tape gateway in real-time since this is backed up using Glacier. Although the on-premises data center is using a tape gateway, you can still set up a solution to use a file gateway in order to properly process the videos using Amazon Rekognition. Keep in mind that the tape gateway in AWS Storage Gateway service is primarily used as an archive solution.

The option that says: ***Use Amazon Kinesis Video Streams to set up a video ingestion stream and with Amazon Rekognition, build a collection of faces. Stream the media files from the MAM solution into Kinesis Video Streams and configure the Amazon Rekognition to process the streamed files. Launch a stream consumer to retrieve the required metadata, and push the metadata into the MAM solution. Finally, configure the stream to store the files in an S3 bucket***is incorrectbecause you won't be able to connect your tape gateway directly to your Kinesis Video Streams service. You need to use the AWS Storage Gateway first.

**References:**

<https://docs.aws.amazon.com/rekognition/latest/dg/collections.html>

<https://aws.amazon.com/storagegateway/file/>

**Check out this Amazon Rekognition Cheat Sheet:**

[**https://tutorialsdojo.com/aws-cheat-sheet-amazon-rekognition/**](https://tutorialsdojo.com/aws-cheat-sheet-amazon-rekognition/)

**Tutorials Dojo's AWS Certified Solutions Architect Professional Exam Study Guide:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-solutions-architect-professional/>

Bottom of Form

Top of Form

Question 66: Skipped

**A print media company has a popular web application hosted on their on-premises network which allows anyone around the globe to search its back catalog and retrieve individual newspaper pages on their web portal. They have scanned the old newspapers into PNG image format and used Optical Character Recognition (OCR) software to automatically convert images to a text file. The license of their OCR software will expire soon and the news organization decided to move to AWS and produce a scalable, durable, and highly available architecture. Which is the best option to achieve this requirement?**

* ​

Use S3 Intelligent-Tiering storage class to store and serve the scanned files. Migrate the on-premises web application as well as the Optical Character Recognition (OCR) software to an Auto Scaling group of Spot EC2 Instances across multiple Availability Zones with an Application Load Balancer to balance the incoming load. Use Amazon Rekognition to detect and recognize text from the scanned old newspapers.

* ​

Create a new S3 bucket to store and serve the scanned image files using a CloudFront web distribution. Launch a new Elastic Beanstalk environment to host the website across multiple Availability Zones and set up a CloudSearch for query processing, which the website can use. Use Amazon Rekognition to detect and recognize text from the scanned old newspapers.

**(Correct)**

* ​

Create a new CloudFormation template which has EBS-backed EC2 instances with an Application Load Balancer in front. Install and run an NGINX web server and an open source search application. Store the images to EBS volumes with Amazon Data Lifecycle Manager configured, and which automatically attach new volumes to the EC2 instances as required.

* ​

Store the images in an S3 bucket and prepare a separate bucket to host the static website. Utilize S3 Select for searching the images stored in S3. Set up a lifecycle policy to move the images to Glacier after 3 months and if needed, use Glacier Select to query the archives.

**Explanation**

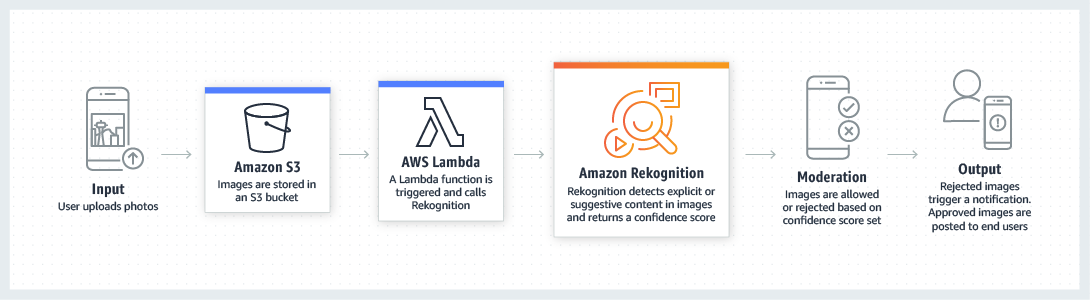
**Amazon CloudSearch** is a managed service in the AWS Cloud that makes it simple and cost-effective to set up, manage, and scale a search solution for your website or application.

With Amazon CloudSearch, you can quickly add rich search capabilities to your website or application. You don't need to become a search expert or worry about hardware provisioning, setup, and maintenance. With a few clicks in the AWS Management Console, you can create a search domain and upload the data that you want to make searchable, and Amazon CloudSearch will automatically provision the required resources and deploy a highly tuned search index.

You can easily change your search parameters, fine-tune search relevance, and apply new settings at any time. As your volume of data and traffic fluctuates, Amazon CloudSearch seamlessly scales to meet your needs.

Hence, the option that says: **Create a new S3 bucket to store and serve the scanned image files using a CloudFront web distribution. Launch a new Elastic Beanstalk environment to host the website across multiple Availability Zones and set up a CloudSearch for query processing, which the website can use. Use Amazon Rekognition to detect and recognize text from the scanned old newspapers** is correct because it satisfies the requirement given in the scenario i.e. it uses S3 to store the images, instead of the commercial product which will be decommissioned soon. More importantly, it uses CloudSearch for query processing, and in addition, it uses Multi-AZ implementation which provides high availability. It is also correct to use Amazon Rekognition to detect and recognize text from the scanned old newspapers.

Amazon Rekognition makes it easy to add image and video analysis to your applications. You just provide an image or video to the Rekognition API, and the service can identify the objects, people, text, scenes, and activities, as well as detect any inappropriate content. Amazon Rekognition also provides highly accurate facial analysis and facial recognition on images and video that you provide. You can detect, analyze, and compare faces for a wide variety of user verification, people counting, and public safety use cases.



The option that says:**Create a new CloudFormation template which has EBS-backed EC2 instances with an Application Load Balancer in front. Install and run an NGINX web server and an open source search application. Store the images to EBS volumes with Amazon Data Lifecycle Manager configured, and which automatically attach new volumes to the EC2 instances as required** is incorrect because an EBS volume is not a scalable nor a durable solution compared with S3. In addition, it is not as cost-effective compared to S3 since it entails maintenance overhead unlike the fully managed storage service provided by S3.

The option that says: **Store the images in an S3 bucket and prepare a separate bucket to host the static website. Utilize S3 Select for searching the images stored in S3. Set up a lifecycle policy to move the images to Glacier after 3 months and if needed, use Glacier Select to query the archives** is incorrect because although using S3 Select is a feasible option, it is not as scalable compared to CloudSearch. Amazon S3 Select can only retrieve a subset of data using SQL statements. Storing your data to Amazon Glacier will also affect the retrieval time of your data.

The option that says: **Use S3 Intelligent-Tiering storage class to store and serve the scanned files. Migrate the on-premises web application as well as the Optical Character Recognition (OCR) software to an Auto Scaling group of Spot EC2 Instances across multiple Availability Zones with an Application Load Balancer to balance the incoming load. Use Amazon Rekognition to detect and recognize text from the scanned old newspapers** is incorrect because even though it properly uses S3 for durable and scalable storage, it still uses the Optical Character Recognition (OCR) software which will be decommissioned soon. It is better to use CloudSearch instead.

**References**:

<https://aws.amazon.com/cloudsearch/>

<https://aws.amazon.com/rekognition/>

**Check out these Amazon CloudSearch and Amazon Rekognition Cheat Sheets:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudsearch/>

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-rekognition/>

Bottom of Form

Top of Form

Question 67: Skipped

A multinational consumer goods corporation structured their AWS accounts to use AWS Organizations, which consolidates payment of their multiple AWS accounts for their various Business Units (BU’s) namely Beauty products, Baby products, Health products, and Home Care products unit. One of their Solutions Architects for the Baby products business unit has purchased 10 Reserved Instances for their new Supply Chain application which will go live 3 months from now. However, they do not want their Reserved Instance (RI) discounts to be shared by the other business units.

Which of the following options is the most suitable solution for this scenario?

* ​

Since the Baby product business unit is part of an AWS Organization, the Reserved Instances will always be shared across other member accounts. There is no way to disable this setting.

* ​

Remove the AWS account of the Baby products business unit out of the AWS Organization.

* ​

Set the Reserved Instance (RI) sharing to private on the AWS account of the Baby products business unit.

* ​

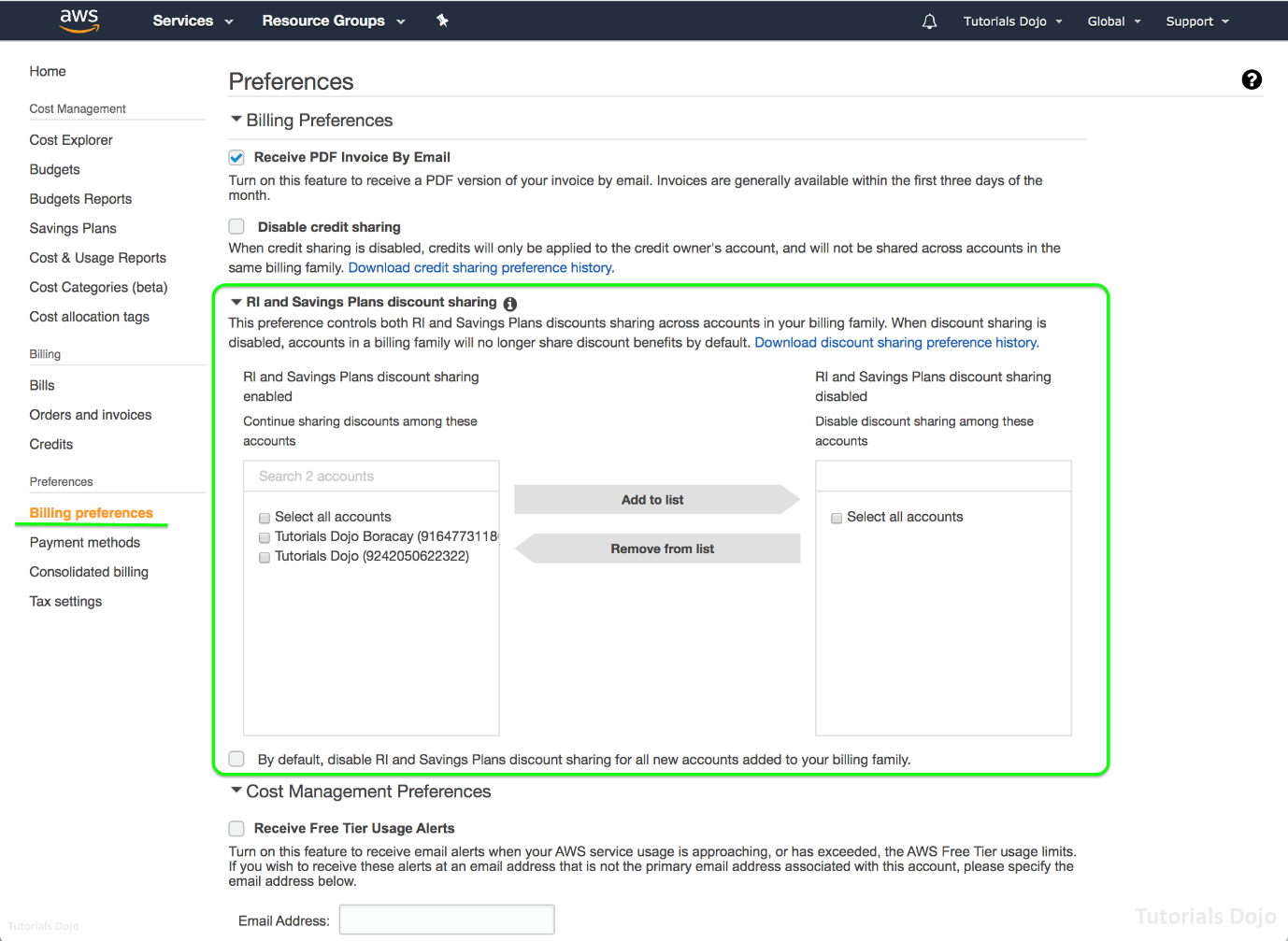
Turn off the Reserved Instance (RI) sharing on the master account for all of the member accounts in the Baby products business unit.

**(Correct)**

**Explanation**

For billing purposes, the consolidated billing feature of AWS Organizations treats all the accounts in the organization as one account. This means that all accounts in the organization can receive the hourly cost-benefit of Reserved Instances that are purchased by any other account. In the payer account, you can turn off Reserved Instance discount sharing on the Preferences page on the Billing and Cost Management console.

The master account of an organization can turn off Reserved Instance (RI) sharing for member accounts in that organization. This means that Reserved Instances are not shared between that member account and other member accounts. You can change this preference multiple times. Each estimated bill is computed using the last set of preferences. However, take note that turning off Reserved Instance sharing can result in a higher monthly bill.



Hence, the correct answer is: **Turn off the Reserved Instance (RI) sharing on the master account for all of the member accounts in the Baby products business unit**.

**Setting the Reserved Instance (RI) sharing to private on the AWS account of the Baby products business unit** is incorrect because there is no "private" option in the RI and Savings Plan discount sharing settings in the Billing Management Console. By default, the member account doesn't have the capability to turn off RI sharing on their account.

**Removing the AWS account of the Baby products business unit out of the AWS Organization** is incorrect because removing the Baby products business unit account from the AWS Organization is not the optimal solution to prevent the other account from sharing its RI discounts. You can simply turn off the Reserved Instance discount sharing in the payer account.

The option that says: **Since the Baby product business unit is part of an AWS Organization, the Reserved Instances will always be shared across other member accounts. There is no way to disable this setting is** incorrect because this statement is false. There is certainly a way to disable the current setting by simply turning off RI sharing.

**References:**

<https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/ri-turn-off.html>

<https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/ri-turn-off-process.html>

**Check out this AWS Billing and Cost Management Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-billing-and-cost-management/>

Bottom of Form

Top of Form

Question 68: Skipped

A company has just launched a new central employee registry application that contains all of the public employee registration information of each staff of the company. The application has a microservices architecture running in Docker in a single AWS Region. The management teams from other departments who have their servers located in different VPCs need to connect to the central repository application to continue their work. The Solutions Architect must ensure that the traffic to the application does not traverse the public Internet. The IT Security team must also be notified of any denied requests and be able to view the corresponding source IP.

How will the Architect implement the architecture of the new application given these circumstances?

* ​

Use AWS Direct Connect to create a dedicated connection between the central VPC and each of the teams' VPCs. Enable the VPC Flow Logs on each VPC to capture rejected traffic requests, including the source IPs, that will be delivered to a CloudWatch Logs group. Set up an Amazon CloudWatch Logs subscription that streams the log data to the IT Security account.

* ​

Set up a Transit VPC by using third-party marketplace VPN appliances running on an On-Demand Amazon EC2 instance that dynamically routes the VPN connections to the virtual private gateways (VGWs) attached to each VPC. Set up an AWS Config rule on each VPC to capture rejected traffic requests, including the source IPs, that will be delivered to an Amazon CloudWatch Logs group. Set up a CloudWatch Logs subscription that streams the log data to the IT Security account.

* ​

Set up an IPSec Tunnel between the central VPC and each of the teams' VPCs. Create VPC Flow Logs on each VPC to capture rejected traffic requests, including the source IPs, that will be delivered to an Amazon CloudWatch Logs group. Create a CloudWatch Logs subscription that streams the log data to the IT Security account.

* ​

Link each of the teams' VPCs to the central VPC using VPC Peering. Create VPC Flow Logs on each VPC to capture rejected traffic requests, including the source IPs, that will be delivered to an Amazon CloudWatch Logs group. Set up a CloudWatch Logs subscription that streams the log data to the IT Security account.

**(Correct)**

**Explanation**

A **VPC peering connection** is a networking connection between two VPCs that enables you to route traffic between them privately. Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own VPCs, with a VPC in another AWS account, or with a VPC in a different AWS Region.

VPC Flow Logs is a feature that enables you to capture information about the IP traffic going to and from network interfaces in your VPC. Flow log data can be published to Amazon CloudWatch Logs or Amazon S3. After you've created a flow log, you can retrieve and view its data in the chosen destination.

Flow logs can help you with a number of tasks, such as:

- Diagnosing overly restrictive security group rules

- Monitoring the traffic that is reaching your instance

- Determining the direction of the traffic to and from the network interfaces

Flow log data is collected outside of the path of your network traffic, and therefore does not affect network throughput or latency. You can create or delete flow logs without any risk of impact to network performance.

Hence, the correct answer is: **Link each of the teams' VPCs to the central VPC using VPC Peering. Create VPC Flow Logs on each VPC to capture rejected traffic requests, including the source IPs, that will be delivered to an Amazon CloudWatch Logs group. Set up a CloudWatch Logs subscription that streams the log data to the IT Security account.**

The option that says: **Set up an IPSec Tunnel between the central VPC and each of the teams' VPCs. Create VPC Flow Logs on each VPC to capture rejected traffic requests, including the source IPs, that will be delivered to an Amazon CloudWatch Logs group. Create a CloudWatch Logs subscription that streams the log data to the IT Security account** is incorrect because it is mentiond in the scenario that the traffic to the application must not traverse the public Internet. Since an IPSec tunnel uses the Internet to transfer data from your VPC to a specified destination, this solution is definitely incorrect.

The option that says: **Use AWS Direct Connect to create a dedicated connection between the central VPC and each of the teams' VPCs. Enable the VPC Flow Logs on each VPC to capture rejected traffic requests, including the source IPs, that will be delivered to a CloudWatch Logs group. Set up an Amazon CloudWatch Logs subscription that streams the log data to the IT Security account** is incorrect because you cannot set up Direct Connect between different VPCs. AWS Direct Connect is primarily used to set up a dedicated connection between your on-premises data center and your Amazon VPC.

The option that says: **Set up a Transit VPC by using third-party marketplace VPN appliances running on an On-Demand Amazon EC2 instance that dynamically routes the VPN connections to the virtual private gateways (VGWs) attached to each VPC. Set up an AWS Config rule on each VPC to capture rejected traffic requests, including the source IPs, that will be delivered to an Amazon CloudWatch Logs group. Set up a CloudWatch Logs subscription that streams the log data to the IT Security account** is incorrect because an AWS Config rule is not capable of capturing source IP of the incoming requests. A VPN appliance is using the public Internet to transfer data. Thus, it violates the requirement of ensuring that the data is securely within the AWS network.

**References**:

<http://docs.aws.amazon.com/AmazonVPC/latest/UserGuide/vpc-peering.html>

<https://docs.aws.amazon.com/vpc/latest/userguide/flow-logs.html>

**Check out these Amazon VPC and VPC Peering Cheat Sheets:**

[https://tutorialsdojo.com/amazon-vpc/](https://tutorialsdojo.com/amazon-vpc/?src=udemy)

[https://tutorialsdojo.com/vpc-peering/](https://tutorialsdojo.com/vpc-peering/?src=udemy)

Bottom of Form

Top of Form

Question 69: Skipped

A fintech startup has developed a cloud-based payment processing system which accepts credit card payments as well as cryptocurrencies such as Bitcoin, Ripple and the likes. The system is deployed in AWS which uses EC2, DynamoDB, S3, and CloudFront to process the payments. Since they are accepting credit card information from the users, they are required to be compliant with the Payment Card Industry Data Security Standard (PCI DSS). On the recent 3rd-party audit, it was found that the credit card numbers are not properly encrypted and hence, their system failed the PCI DSS compliance test. You were hired by the fintech startup to solve this issue so they can release the product in the market as soon as possible. In addition, you also have to improve performance by increasing the proportion of your viewer requests that are served from CloudFront edge caches instead of going to your origin servers for content.

In this scenario, what is the best option to protect and encrypt the sensitive credit card information of the users and to improve the cache hit ratio of your CloudFront distribution?

* ​

Add a custom SSL in the CloudFront distribution. Configure your origin to add User-Agent and Host headers to your objects to increase your cache hit ratio.

* ​

Configure the CloudFront distribution to use Signed URLs. Configure your origin to add a Cache-Control max-age directive to your objects, and specify the longest practical value for max-age to increase your cache hit ratio.

* ​

Configure the CloudFront distribution to enforce secure end-to-end connections to origin servers by using HTTPS and field-level encryption. Configure your origin to add a Cache-Control max-age directive to your objects, and specify the longest practical value for max-age to increase your cache hit ratio.

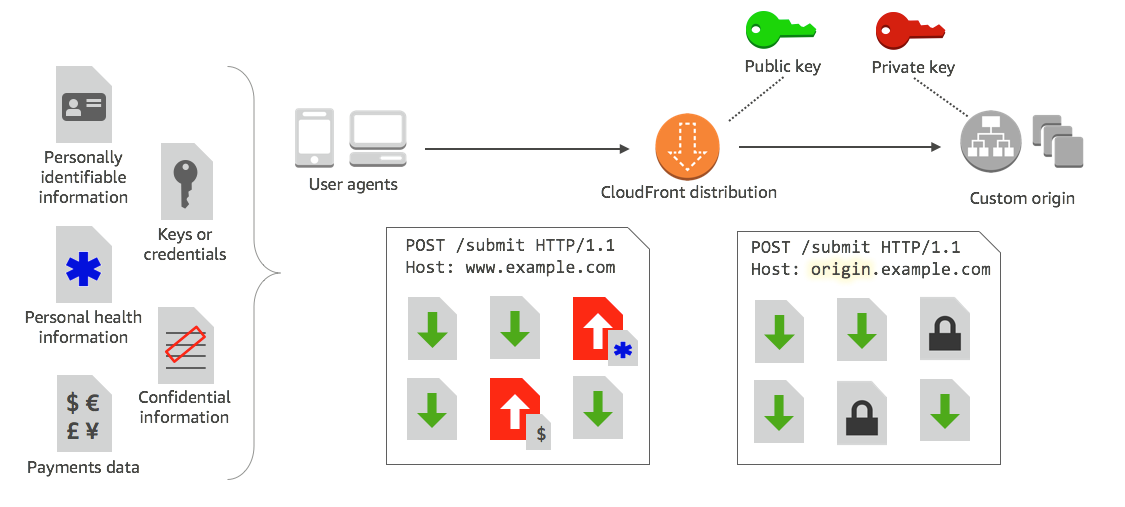
**(Correct)**

* ​

Create an origin access identity (OAI) and add it to the CloudFront distribution. Configure your origin to add User-Agent and Host headers to your objects to increase your cache hit ratio.

**Explanation**

Field-level encryption adds an additional layer of security along with HTTPS that lets you protect specific data throughout system processing so that only certain applications can see it. Field-level encryption allows you to securely upload user-submitted sensitive information to your web servers. The sensitive information provided by your clients is encrypted at the edge closer to the user and remains encrypted throughout your entire application stack, ensuring that only applications that need the data—and have the credentials to decrypt it—are able to do so.



To use field-level encryption, you configure your CloudFront distribution to specify the set of fields in POST requests that you want to be encrypted, and the public key to use to encrypt them. You can encrypt up to 10 data fields in a request. Hence, the correct answer for this scenario is the option that says: ***Configure the CloudFront distribution to enforce secure end-to-end connections to origin servers by using HTTPS and field-level encryption. Configure your origin to add a Cache-Control max-age directive to your objects, and specify the longest practical value for max-age to increase your cache hit ratio.***

You can improve performance by increasing the proportion of your viewer requests that are served from CloudFront edge caches instead of going to your origin servers for content; that is, by improving the cache hit ratio for your distribution. To increase your cache hit ratio, you can configure your origin to add a Cache-Control max-age directive to your objects, and specify the longest practical value for max-age. The shorter the cache duration, the more frequently CloudFront forwards another request to your origin to determine whether the object has changed and, if so, to get the latest version.

The option that says: ***Add a custom SSL in the CloudFront distribution. Configure your origin to add User-Agent and Host headers to your objects to increase your cache hit ratio*** is incorrect as although it provides secure end-to-end connections to origin servers, it is better to add field-level encryption to protect the credit card information.

The option that says: ***Configure the CloudFront distribution to use Signed URLs. Configure your origin to add a* Cache-Control max-age directive to your objects, and specify the longest practical value for max-age to increase your cache hit ratio** is incorrect because a Signed URL provides a way to distribute private content but it doesn't encrypt the sensitive credit card information.

The option that says: **Create an Origin Access Identity (OAI) and add it to the CloudFront distribution. Configure your origin to add *User-Agent*and *Host* headers to your objects to increase your cache hit ratio** is incorrect because OAI is mainly used to restrict access to objects in S3 bucket, but not provide encryption to specific fields.

**References:**

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/field-level-encryption.html#field-level-encryption-setting-up>

<https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/cache-hit-ratio.html>

**Check out this Amazon CloudFront Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-cloudfront/>

**Tutorials Dojo's AWS Certified Solutions Architect Professional Exam Study Guide:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-certified-solutions-architect-professional/>

Bottom of Form

Top of Form

Question 70: Skipped

You are working as a Solutions Architect for a leading IT consultancy company which has offices in San Francisco, Frankfurt, Tokyo, and Manila. They are using AWS Organizations to easily manage the multiple AWS accounts being used by their regional offices and subsidiaries. A new AWS account was recently added to a specific organizational unit (OU) which is responsible for the overall systems administration. The administrator noticed that the account is using a root-created AWS ECS Cluster with an attached service-linked role. For regulatory purposes, you created a custom SCP that would deny the new account from performing certain actions in relation to using ECS. However, after applying the policy, the new account could still perform the actions that it was supposed to be restricted from doing.

What could be the most likely reason for this problem?

* ​

The default SCP grants all permissions attached to every root, OU, and account. To apply stricter permissions, this policy is required to be modified.

* ​

There is an SCP attached to a higher-level OU that permits the actions of the service-linked role. This permission would therefore be inherited by the current OU, and override the SCP placed by the administrator.

* ​

The ECS service is being run outside the jurisdiction of the organization. SCPs affect only the principals that are managed by accounts that are part of the organization.

* ​

SCPs do not affect any service-linked role. Service-linked roles enable other AWS services to integrate with AWS Organizations and can't be restricted by SCPs.

**(Correct)**

**Explanation**

Users and roles must still be granted permissions using IAM permission policies attached to them or to groups. The SCPs filter the permissions granted by such policies, and the user can't perform any actions that the applicable SCPs don't allow. Actions allowed by the SCPs can be used if they are granted to the user or role by one or more IAM permission policies.

When you attach SCPs to the root, OUs, or directly to accounts, all policies that affect a given account are evaluated together using the same rules that govern IAM permission policies:

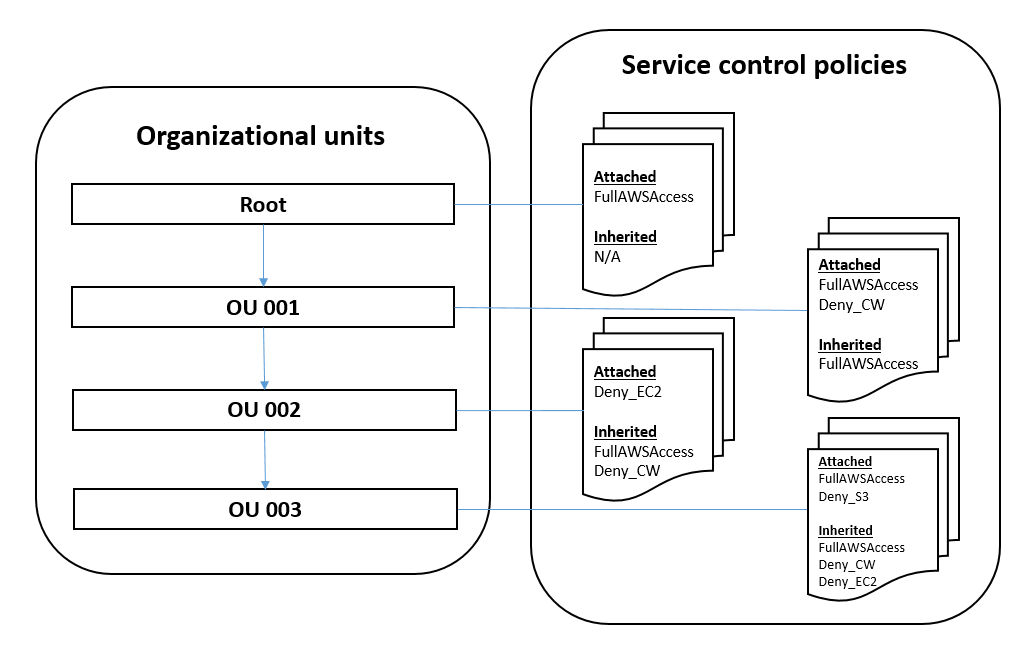
- Any action that has an explicit Deny in an SCP can't be delegated to users or roles in the affected accounts. An explicit Deny statement overrides any Allow that other SCPs might grant.

- Any action that has an explicit Allow in an SCP (such as the default "\*" SCP or by any other SCP that calls out a specific service or action) can be delegated to users and roles in the affected accounts.

- Any action that isn't explicitly allowed by an SCP is implicitly denied and can't be delegated to users or roles in the affected accounts.

By default, an SCP named FullAWSAccess is attached to every root, OU, and account. This default SCP allows all actions and all services. So in a new organization, until you start creating or manipulating the SCPs, all of your existing IAM permissions continue to operate as they did. As soon as you apply a new or modified SCP to a root or OU that contains an account, the permissions that your users have in that account become filtered by the SCP. Permissions that used to work might now be denied if they're not allowed by the SCP at every level of the hierarchy down to the specified account.

As stated in the documentation of AWS Organizations, ***SCPs DO NOT affect any service-linked role. Service-linked roles enable other AWS services to integrate with AWS Organizations and can't be restricted by SCPs.***



The option that says: ***The default SCP grants all permissions attached to every root, OU, and account. To apply stricter permissions, this policy is required to be modified*** is incorrect because the scenario already implied that the administrator created a **Deny** policy. By default, an SCP named *FullAWSAccess* is attached to every root, OU, and account. This default SCP allows all actions and all services. However, you specify a **Deny** policy if you want to create a blacklist that blocks all access to the specified services and actions. The explicit **Deny** on specific actions in the blacklist policy overrides the **Allow** in any other policy, such as the one in the default SCP.

The option that says: ***There is an SCP attached to a higher-level OU that permits the actions of the service-linked role. This permission would therefore be inherited by the current OU, and override the SCP placed by the administrator*** is incorrect because even if a higher-level OU has an SCP attached with an **Allow** policy for the service, the current set up should still have restricted access to the service. Creating and attaching a new **Deny** SCP to the new account's OU will not be affected by the pre-existing Allow policy in the same OU.

The option that says: ***The ECS service is being run outside the jurisdiction of the organization. SCPs affect only the principals that are managed by accounts that are part of the organization***is incorrect because the service-linked role must have been created within the organization, most notably by the root account of the organization. It also does not make sense if we take the assumption that the service is indeed outside of the organization's jurisdiction, because the *Principal* element of a policy specifies which entity will have limited permissions. But the scenario tells us that it should be the new account that is denied certain actions, not the service itself.

**References:**

<https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scp.html>

<https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_about-scps.html>

**Service Control Policies (SCP) vs IAM Policies:**

<https://tutorialsdojo.com/aws-cheat-sheet-service-control-policies-scp-vs-iam-policies/>

**Comparison of AWS Services Cheat Sheets:**

<https://tutorialsdojo.com/comparison-of-aws-services-for-udemy-students/>

Bottom of Form

Top of Form

Question 71: Skipped

An electronics company has an on-premises network as well as a cloud infrastructure in AWS. The on-site data storage which is used by their enterprise document management system is heavily being used, and **they are looking at utilizing the storage services in AWS for cost-effective backup and rapid disaster recovery**. You are tasked to set up a storage solution that will provide a low-latency access to the enterprise document management system. Most of the documents uploaded in their system are printed circuit board (PCB) designs and schematic diagrams which are frequently used and accessed by their engineers, QA analysts, and their Research and Design department. Hence, you also have to ensure that these employees can access the entire dataset quickly, without sacrificing durability.

How can you satisfy the requirement for this scenario?

* ​

In AWS Storage Gateway, create a File gateway that enables you to store and retrieve objects in Amazon S3 using industry-standard file protocols such as Network File System (NFS) and Server Message Block (SMB).

* ​

Use a Cached volume gateway to retain low-latency access to your entire data set as well as your frequently accessed data.

* ​

Create an S3 bucket and use the sync command to synchronize the data to and from your on-premises file server.

* ​

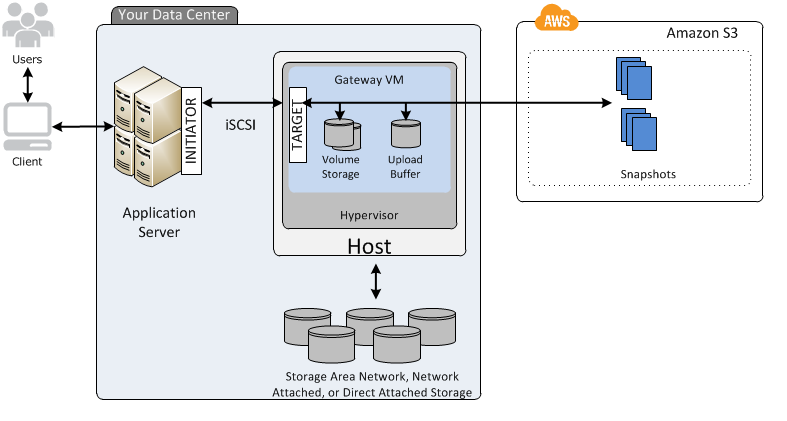
Use a Stored Volume Gateway to provide cloud-backed storage volumes that you can mount as Internet Small Computer System Interface (iSCSI) devices from your on-premises application servers.

**(Correct)**

**Explanation**

In this scenario, the electronics company is looking for a storage service which can locally store the entire dataset and provide easier and faster access without sacrificing durability. This requirement can be fulfilled by setting up a Stored Volume gateway in AWS Storage Gateway.

By using stored volumes, you can store your primary data locally, while asynchronously back up that data to AWS. Stored volumes provide your on-premises applications with low-latency access to the entire datasets. At the same time, they provide durable, offsite backups. You can create storage volumes and mount them as iSCSI devices from your on-premises application servers. Data written to your stored volumes are stored on your on-premises storage hardware. These data are asynchronously backed up to Amazon S3 as Amazon Elastic Block Store (Amazon EBS) snapshots. Hence, the correct answer is ***using a Stored Volume Gateway to provide cloud-backed storage volumes that you can mount as Internet Small Computer System Interface (iSCSI) devices from your on-premises application servers.***



***Creating an S3 bucket and using the sync command to synchronize the data to and from your on-premises file server*** is incorrect because S3 is not a preferred AWS storage service for supporting hybrid networks for this scenario. Although S3 is highly scalable, it would not be able to handle the integration needed for the on-premises document management system. It is still better to use Stored Volumes in AWS Storage Gateway instead.

***Using a Cached volume gateway to retain low-latency access to your entire data set as well as your frequently accessed data*** is incorrect because Cached volume gateway provides you low-latency access to your frequently accessed data but not to the entire data.

The option that says: ***In AWS Storage Gateway, create a File gateway that enables you to store and retrieve objects in Amazon S3 using industry-standard file protocols such as Network File System (NFS) and Server Message Block (SMB)***is incorrect because a File gateway does not provide you the required low-latency access to the entire dataset that the application needs.

**References:**

<https://docs.aws.amazon.com/storagegateway/latest/userguide/StorageGatewayConcepts.html#volume-gateway-concepts>

<https://docs.aws.amazon.com/storagegateway/latest/userguide/WhatIsStorageGateway.html>

**Check out this AWS Storage Gateway Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-storage-gateway/>

Bottom of Form

Top of Form

Question 72: Skipped

A legacy web application, which is composed of an Auto Scaling group of EC2 instances and an Application Load Balancer, will be decommissioned soon. You are tasked to set up a new serverless architecture that comprises of AWS Lambda, API Gateway, and DynamoDB. In addition, you are also required to build a CI/CD pipeline to automate the build process and support gradual deployments.

Which is the most suitable way to build, test, and deploy your new architecture in AWS?

* ​

Use the AWS Serverless Application Repository to organize related components, share configuration such as memory and timeouts between resources, and deploy all related resources together as a single, versioned entity.

* ​

Set up a CI/CD pipeline using CodeCommit, CodeBuild, CodeDeploy, and CodePipeline to build the CI/CD pipeline then use AWS Systems Manager Automation to automate the build process and support gradual deployments.

* ​

Use CloudFormation and OpsWorks for your build, deployment, and configuration management service.

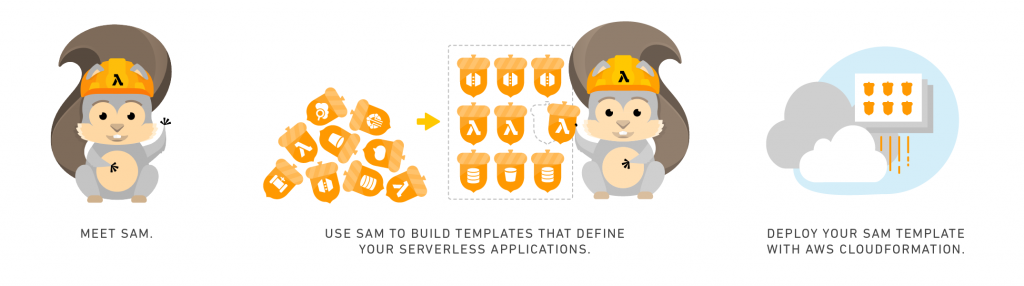
* ​

Use AWS Serverless Application Model (AWS SAM) and set up AWS CodeBuild, AWS CodeDeploy, and AWS CodePipeline to build a CI/CD pipeline.

**(Correct)**

**Explanation**

The ***AWS Serverless Application Model (AWS SAM)*** is an open-source framework that you can use to build serverless applications on AWS. It consists of the AWS SAM template specification that you use to define your serverless applications, and the AWS SAM command line interface (AWS SAM CLI) that you use to build, test, and deploy your serverless applications.



Because AWS SAM is an extension of AWS CloudFormation, you get the reliable deployment capabilities of AWS CloudFormation. You can define resources by using AWS CloudFormation in your AWS SAM template. Also, you can use the full suite of resources, intrinsic functions, and other template features that are available in AWS CloudFormation.

You can use AWS SAM with a suite of AWS tools for building serverless applications. To build a deployment pipeline for your serverless applications, you can use CodeBuild, CodeDeploy, and CodePipeline. You can also use AWS CodeStar to get started with a project structure, code repository, and a CI/CD pipeline that's automatically configured for you. To deploy your serverless application, you can use the Jenkins plugin, and you can use Stackery.io's toolkit to build production-ready applications.

**References:**

<https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/what-is-sam.html>

<https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/automating-updates-to-serverless-apps.html>

**Check out this AWS Serverless Application Model Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-aws-serverless-application-model-sam/>

Bottom of Form

Top of Form

Question 73: Skipped

The telecommunications company that you are working for will do a major public announcement for a new phone offer and it is expected that there would be millions of people who will access their website to get the new offer. Their e-commerce platform is running on an Auto Scaling group of On-Demand EC2 instances deployed across multiple Availability Zones. For the database tier, the platform is using an Amazon RDS database in a Multi-AZ deployments configuration. Their e-commerce site performs a high number of small reads and writes per second to handle customer transactions and relies on an eventual consistency model. The Operations team identified that there is read contention on RDS MySQL database after conducting a series of performance tests.

Which combination of options should you implement to provide a fast, cost-efficient, and scalable solution? (Select TWO.)

* ​

Modify the Amazon RDS Multi-AZ deployments configuration to launch multiple standby database instances. Distribute the incoming traffic to the standby instances to improve the database performance.

* ​

Set up Read Replicas in each Availability Zone.

**(Correct)**

* ​

Migrate the database to Amazon Redshift and use its massively parallel query execution capability to improve the read performance of the application.

* ​

Implement an in-memory cache using Amazon ElastiCache

**(Correct)**

* ​

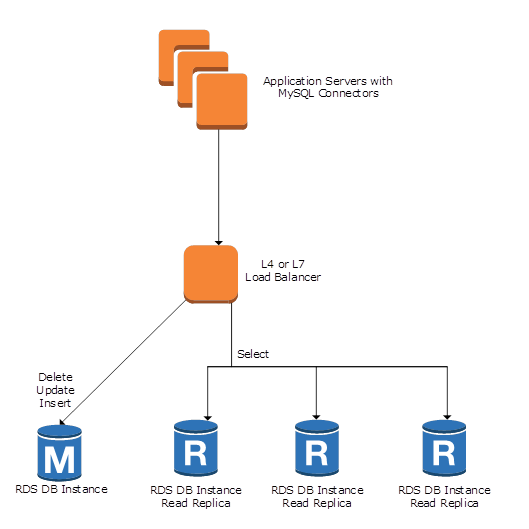
Vertically scale your RDS MySQL Instance by upgrading its instance size with provisioned IOPS.

**Explanation**

For this scenario, the optimal services to use are **Amazon ElastiCache** and **RDS Read Replicas**. Amazon ElastiCache can be used to significantly improve latency and throughput for many read-heavy application workloads (such as social networking, gaming, media sharing and Q&A portals) or compute-intensive workloads (such as a recommendation engine) by allowing you to store the objects that are often read in cache. Moreover, with Redis’ support for advanced data structures, you can augment the database tier to provide features (such as leaderboard, counting, session and tracking) that are not easily achievable via databases in a cost-effective way.

Amazon RDS Read Replicas provide enhanced performance and durability for database (DB) instances. This feature makes it easy to elastically scale out beyond the capacity constraints of a single DB Instance for read-heavy database workloads.

You can create one or more replicas of a given source DB Instance and serve high-volume application read traffic from multiple copies of your data, thereby increasing aggregate read throughput. Read replicas can also be promoted when needed to become standalone DB instances. Read replicas are available in Amazon RDS for MySQL, MariaDB, Oracle, and PostgreSQL as well as Amazon Aurora.



You can reduce the load on your source DB instance by routing read queries from your applications to the read replica. Read replicas allow you to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads.

To further maximize read performance, Amazon RDS for MySQL allows you to add table indexes directly to Read Replicas, without those indexes being present on the master.

Because read replicas can be promoted to master status, they are useful as part of a sharding implementation. To shard your database, add a read replica and promote it to master status, then, from each of the resulting DB Instances, delete the data that belongs to the other shard.

***Setting up Read Replicas in each Availability Zone*** is correct because Read Replicas are used to elastically scale out beyond the capacity constraints of a single DB instance for read-heavy database workloads hence, improving the read performance.

***Implementing an in-memory cache using Amazon ElastiCache*** is correct because ElastiCache is an in-memory caching solution which reduces the load on the database and improves the read performance.

***Migrating the database to Amazon Redshift and using its massively parallel query execution capability to improve the read performance of the application*** is incorrect because Amazon Redshift is more suitable for OLAP-type applications and not for online transaction processing (OLTP). Redshift is also not suitable to host your MySQL database.

***Modifying the Amazon RDS Multi-AZ deployments configuration to launch multiple standby database instances and distributing the incoming traffic to the standby instances to improve the database performance*** is incorrect because you cannot distribute the incoming traffic to the standby instances since these are not readable at all. These database instances are primarily used to improve the availability of your database and your application.

***Vertically scaling your RDS MySQL Instance by upgrading its instance size with provisioned IOPS*** is incorrect because although upgrading the instance size may improve the read performance to a certain extent, it is not as scalable compared with Read Replicas or ElastiCache.

**References**:

<https://aws.amazon.com/elasticache/>

<https://aws.amazon.com/rds/details/read-replicas/>

**Check out this Amazon Elasticache Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-elasticache/>

**Check out this Amazon RDS Cheat Sheet:**

<https://tutorialsdojo.com/aws-cheat-sheet-amazon-relational-database-service-amazon-rds/>

Bottom of Form

Top of Form

Question 74: Skipped

A company has several AWS accounts that are managed using AWS Organizations. The company created only one organizational unit (OU) so all child accounts are members of the Production OU. The Solutions Architects control access to certain AWS services using SCPs that define the restricted services. The SCPs are attached at the root of the organization so that they will be applied to all AWS accounts under the organization. The company recently acquired a small business firm and its existing AWS account was invited to join the organization. Upon onboarding, the administrators of the small business firm cannot apply the required AWS Config rules to meet the parent company’s security policies.

Which of the following options will allow the administrators to update the AWS Config rules on their AWS account without introducing long-term management overhead?

* ​

Update the SCPs applied in the root of the AWS organization and remove the rule that restricts changes to the AWS Config service. Deploy a new AWS Service Catalog to the whole organization containing the company’s AWS Config policies.

* ​

Remove the SCPs on the organization’s root and apply them to the Production OU instead. Create a temporary Onboarding OU that has an attached SCP allowing changes to AWS Config. Add the new account to this temporary OU and make the required changes before moving it to Production OU.

**(Correct)**

* ​

Add the new account to a temporary Onboarding organization unit (OU) that has an attached SCP allowing changes to AWS Config. Perform the needed changes while on this temporary OU before moving the new account to Production OU.

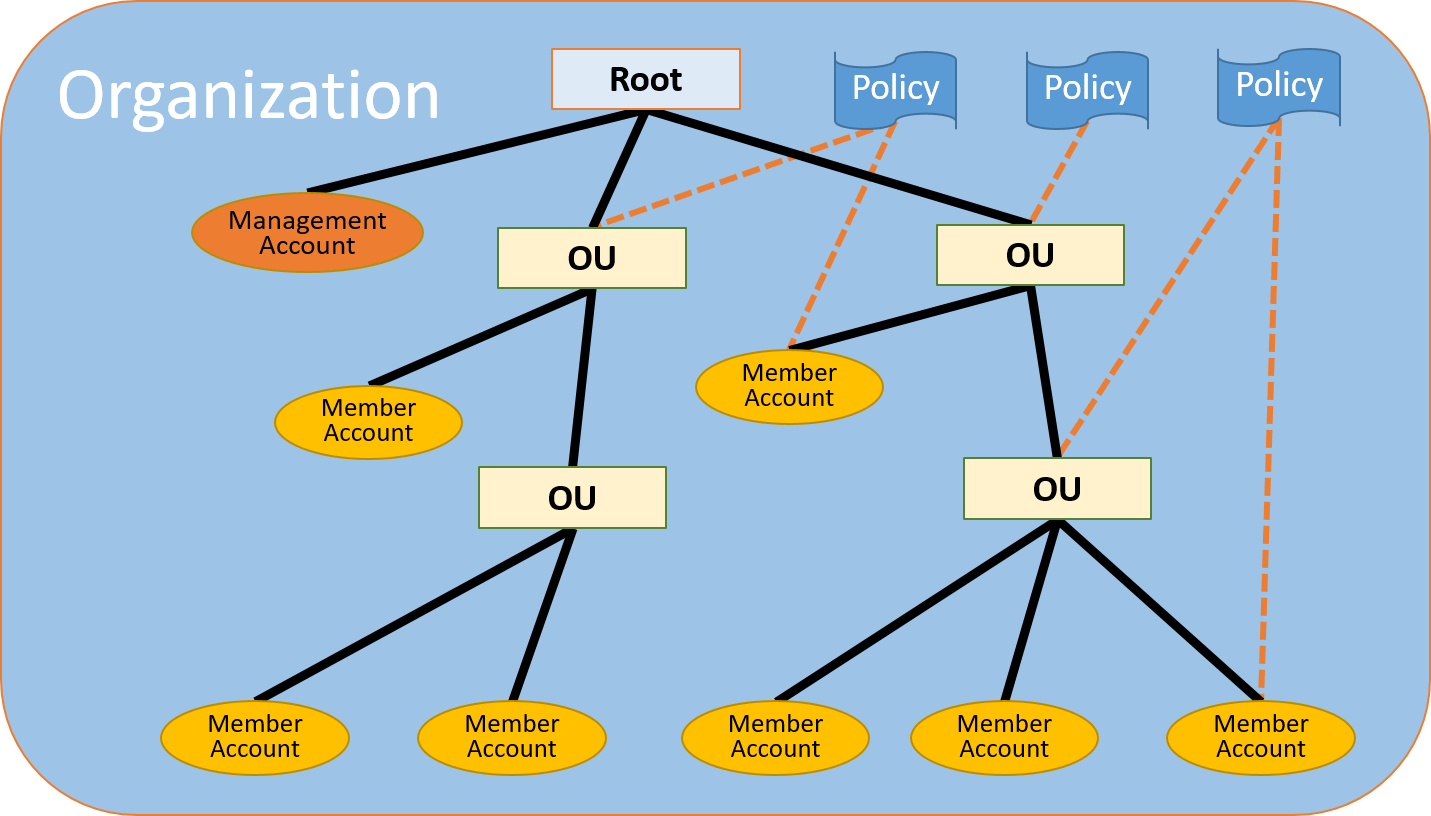
* ​

Instead of using a “deny list” to AWS services on the organization’s root SCPs, use an “allow list” to allow only the required AWS services. Temporarily add the AWS Config service on the “allow list” for the principals of the new account and make the required changes.

**Explanation**

**AWS Organizations** helps you centrally manage and govern your environment as you grow and scale your AWS resources. Using AWS Organizations, you can programmatically create new AWS accounts and allocate resources, group accounts to organize your workflows, apply policies to accounts or groups for governance, and simplify billing by using a single payment method for all of your accounts.

With AWS Organizations, you can consolidate multiple AWS accounts into an organization that you create and centrally manage. You can create member accounts and invite existing accounts to join your organization. You can organize those accounts into groups and attach policy-based controls.



**Service control policies (SCPs)** are a type of organization policy that you can use to manage permissions in your organization. SCPs offer central control over the maximum available permissions for all accounts in your organization. SCPs help you to ensure your accounts stay within your organization’s access control guidelines. SCPs are available only in an organization that has all features enabled.

An SCP restricts permissions for IAM users and roles in member accounts, including the member account's root user. Any account has only those permissions permitted by every parent above it. If a permission is blocked at any level above the account, either implicitly (by not being included in an Allow policy statement) or explicitly (by being included in a Deny policy statement), a user or role in the affected account can't use that permission, even if the account administrator attaches the **AdministratorAccess** IAM policy with \*/\* permissions to the user.

AWS strongly recommends that you don't attach SCPs to the root of your organization without thoroughly testing the impact that the policy has on accounts. Instead, create an OU that you can move your accounts into one at a time, or at least in small numbers, to ensure that you don't inadvertently lock users out of key services.

Therefore, the correct answer is: **Remove the SCPs on the organization’s root and apply them to the Production OU instead. Create a temporary Onboarding OU that has an attached SCP allowing changes to AWS Config. Add the new account to this temporary OU and make the required changes before moving it to Production OU.** It is not recommended to attach the SCPs to the root of the organization, so it is better to move all the SCPs to the Production OU. This way, the temporary Onboarding OU can have an independent SCP to allow the required changes on AWS Config. Then, you can move the new AWS account to the Production OU.

The option that says: **Update the SCPs applied in the root of the AWS organization and remove the rule that restricts changes to the AWS Config service. Deploy a new AWS Service Catalog to the whole organization containing the company’s AWS Config policies** is incorrect. Although AWS Service Catalog allows organizations to create and manage catalogs of IT services that are approved for use on AWS, this will cause possible problems in the future for the administrators. Removing the AWS Config restriction on the root of the AWS organization's SCP will allow all Admins on all AWS accounts to manage/change/update their own AWS Config rules.

The option that says: **Add the new account to a temporary Onboarding organization unit (OU) that has an attached SCP allowing changes to AWS Config. Perform the needed changes while on this temporary OU before moving the new account to Production OU** is incorrect. If the SCP applied on the organization's root has a "deny" permission, all OUs under the organization will inherit that rule. You cannot override an explicit "deny" permission with an explicit "allow" applied to the temporary Onboarding OU.

The option that says: **Instead of using a “deny list” to AWS services on the organization’s root SCPs, use an “allow list” to allow only the required AWS services. Temporarily add the AWS Config service on the “allow list” for the principals of the new account and make the required changes** is incorrect. This is possible however, this will cause more management problems in the future as you will have to update the "allow list" for any service that users may require in the future.

**References:**

<https://docs.aws.amazon.com/controltower/latest/userguide/organizations.html>

<https://aws.amazon.com/organizations/>

<https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps.html>

**Check out the AWS Organization and SCP comparison Cheat Sheets:**

[https://tutorialsdojo.com/aws-organizations/](https://tutorialsdojo.com/aws-organizations/?src=udemy)

[https://tutorialsdojo.com/service-control-policies-scp-vs-iam-policies/](https://tutorialsdojo.com/service-control-policies-scp-vs-iam-policies/?src=udemy)

Bottom of Form

Top of Form

Question 75: Skipped

In an effort to ensure and strengthen data security, your company has launched a company-wide bug bounty program to find and patch up security vulnerabilities in your web applications as well as the underlying cloud resources. You are working as a Cloud Engineer and decided to focus on checking system vulnerabilities of your AWS resources. Due to budget constraints, the company cannot afford to use AWS Shield Advanced.

Which of the following are the best techniques to help mitigate Distributed Denial of Service (DDoS) attacks for your cloud infrastructure hosted in AWS? (Select TWO.)

* ​

Add multiple Elastic Network Interfaces to each EC2 instance and use Enhanced Networking to increase the network bandwidth.

* ​

Use Reserved EC2 instances to ensure that each instance has the maximum performance possible. Use AWS WAF to protect your web applications from common web exploits that could affect application availability.

* ​

Use an Amazon CloudFront distribution for both static and dynamic content of your web applications. Add CloudWatch alerts to automatically look and notify the Operations team for high CPUUtilization and NetworkIn metrics, as well as to trigger Auto Scaling of your EC2 instances.

**(Correct)**

* ​

Use an Application Load Balancer (ALB) to reduce the risk of overloading your application by distributing traffic across many backend instances. Integrate AWS WAF and the ALB to protect your web applications from common web exploits that could affect application availability.

**(Correct)**

* ​

Use S3 as a POSIX-compliant storage instead of EBS Volumes for storing data. Install the SSM agent to all of your instances and use AWS Systems Manager Patch Manager to automatically patch your instances.

**Explanation**

The following options are the correct answers in this scenario as they can help mitigate the effects of DDoS attacks:

**- Use an Amazon CloudFront distribution for both static and dynamic content of your web applications. Add CloudWatch alerts to automatically look and notify the Operations team for high CPUUtilization and NetworkIn metrics, as well as to trigger Auto Scaling of your EC2 instances**.

**- Use an Application Load Balancer (ALB) to reduce the risk of overloading your application by distributing traffic across many backend instances. Integrate AWS WAF and the ALB to protect your web applications from common web exploits that could affect application availability**.

**Amazon CloudFront** is a content delivery network (CDN) service that can be used to deliver your entire website, including static, dynamic, streaming, and interactive content. Persistent TCP connections and variable time-to-live (TTL) can be used to accelerate delivery of content, even if it cannot be cached at an edge location. This allows you to use Amazon CloudFront to protect your web application, even if you are not serving static content. Amazon CloudFront only accepts well-formed connections to prevent many common DDoS attacks like SYN floods and UDP reflection attacks from reaching your origin.

Larger DDoS attacks can exceed the size of a single Amazon EC2 instance. To mitigate these attacks, you will want to consider options for load balancing excess traffic. With Elastic Load Balancing (ELB), you can reduce the risk of overloading your application by distributing traffic across many backend instances. ELB can scale automatically, allowing you to manage larger volumes of unanticipated traffic, like flash crowds or DDoS attacks.

Another way to deal with application layer attacks is to operate at scale. In the case of web applications, you can use ELB to distribute traffic to many Amazon EC2 instances that are overprovisioned or configured to auto scale for the purpose of serving surges of traffic, whether it is the result of a flash crowd or an application layer DDoS attack. Amazon CloudWatch alarms are used to initiate Auto Scaling, which automatically scales the size of your Amazon EC2 fleet in response to events that you define. This protects application availability even when dealing with an unexpected volume of requests.

The option that says: **Use S3 as a POSIX-compliant storage instead of EBS Volumes for storing data. Install the SSM agent to all of your instances and use AWS Systems Manager Patch Manager to automatically patch your instances** is incorrect because using S3 instead of EBS Volumes is mainly for addressing scalability to your storage requirements and not for avoiding DDoS attacks. In addition, Amazon S3 is not a POSIX-compliant storage.

The option that says:**Add multiple Elastic Network Interfaces to each EC2 instance and use Enhanced Networking to increase the network bandwidth** is incorrect because adding multiple ENIs and using Enhanced Networking does not protect you from DDoS Attacks.

The option that says: **Use Reserved EC2 instances to ensure that each instance has the maximum performance possible. Use AWS WAF to protect your web applications from common web exploits that could affect application availability** is incorrect because using Reserved EC2 instances do not provide any additional computing performance compared to other EC2 types.

**References:**

<https://docs.aws.amazon.com/waf/latest/developerguide/shield-chapter.html>

[https://docs.aws.amazon.com/waf/latest/developerguide/what-is-aws-waf.html](https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf)

<https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf>

**Check out this Amazon CloudFront Cheat Sheet:**

[https://tutorialsdojo.com/amazon-cloudfront/](https://tutorialsdojo.com/amazon-cloudfront/?src=udemy)

**Best practices on DDoS Attack Mitigation:**

<https://youtu.be/HnoZS5jj7pk>

Bottom of Form

Continue

Retake test

Fullscreen