Solutions Architect Professional Questions 4

Question 1

A company intends to transition its existing VMware cluster, consisting of 120 VMs with various operating systems and custom software, to AWS. Additionally, they have a 10 TB NFS server on-premises. To facilitate this move, the company has already established a 10 Gbps AWS Direct Connect link.

The task at hand is to determine the most efficient method to migrate these resources to AWS, considering both time efficiency and the nature of the existing infrastructure. The available options are:

A. VM Export and Snowball Edge Approach: Export the VMs to an Amazon S3 bucket, and then use VM Import/Export to convert them into AMIs. Concurrently, order an AWS Snowball Edge device to transfer the NFS server data, subsequently restoring it to an NFS-configured Amazon EC2 instance.

B. AWS Application Migration Service and DataSync Method: Utilize AWS Application Migration Service to replicate the VMs directly from the VMware cluster. Simultaneously, use AWS DataSync to transfer the NFS server data to an Amazon Elastic File System (EFS) over the existing Direct Connect link.

C. Manual EC2 Recreation and DataSync to FSx for Lustre: Manually recreate each VM as an Amazon EC2 instance, reinstalling necessary software. Use AWS DataSync to migrate the NFS data to an Amazon FSx for Lustre file system via Direct Connect.

D. Dual Snowball Edge and VM Import/Export Strategy: Order two AWS Snowball Edge devices to transfer both the VMs and NFS server data. Post-transfer, use VM Import/Export for the VM data and move the NFS data to an Amazon EFS.

The most efficient solution is B:

- B. AWS Application Migration Service and DataSync Method: This approach leverages AWS Application Migration Service for direct VM replication, avoiding the time-consuming process of exporting and importing VMs. AWS DataSync efficiently transfers NFS data over Direct Connect to Amazon EFS, ensuring a swift and streamlined migration process.

Reasons why the other options are less suitable:

- A. VM Export and Snowball Edge Approach: Exporting VMs and using Snowball Edge introduces additional steps and potential delays, especially in exporting and importing VMs, which is more time-consuming compared to direct replication.
- C. Manual EC2 Recreation and DataSync to FSx for Lustre: Manually recreating VMs is highly time-consuming and prone to inconsistencies. While DataSync is efficient, the manual recreation process negates its time-saving benefits.
- D. Dual Snowball Edge and VM Import/Export Strategy: Utilizing two Snowball Edge devices adds complexity and time to the migration process. While Snowball is effective for massive data transfers, it's less efficient for migrating VMs compared to direct replication methods.

Question 2

An online survey company operating in the AWS Cloud utilizes a distributed application composed of microservices running on a scalable Amazon ECS cluster. The cluster is linked to an Application Load Balancer, which in turn, serves as a custom origin for an Amazon CloudFront distribution. The company conducts surveys involving sensitive data, necessitating encryption during transit within the application. Specifically, only the data-handling microservice should have the capability to decrypt this data.

To fulfill these encryption requirements, the following solutions are considered:

A. AWS KMS Key with Field-Level Encryption in CloudFront: Use a dedicated symmetric key from AWS KMS for the data-handling microservice. Establish a field-level encryption profile and configuration in CloudFront, integrating it with the KMS key.

B. RSA Key Pair with CloudFront Integration: Generate an RSA key pair exclusively for the data-handling microservice. Upload the public key to the CloudFront distribution. Develop a field-level encryption profile and configuration, then apply this configuration to the CloudFront cache behavior.

C. AWS KMS Key with Lambda@Edge for Encryption: Create a dedicated symmetric AWS KMS key for the data-handling microservice and develop a Lambda@Edge function. Program this function to employ the KMS key for encrypting sensitive data.

D. RSA Key Pair with Lambda@Edge for Encryption: Generate an RSA key pair for the data-handling microservice and create a Lambda@Edge function. Design the function to use the RSA pair's private key for data encryption.

The most suitable solution is B:

- B. RSA Key Pair with CloudFront Integration: This approach allows for the secure transmission of sensitive data using field-level encryption in CloudFront. By uploading

the public key of the RSA pair to CloudFront and associating it with the cache behavior, data can be encrypted at the edge, ensuring only the data-handling microservice with the corresponding private key can decrypt it. This solution aligns with the requirement of having only one microservice capable of decryption.

Reasons why the other options are less ideal:

- A. AWS KMS Key with Field-Level Encryption in CloudFront: While using AWS KMS for encryption is a secure method, it doesn't specify how the encrypted data is handled at the edge level by CloudFront, potentially allowing other microservices access to the data.
- C. AWS KMS Key with Lambda@Edge for Encryption: Employing a KMS key in Lambda@Edge for encryption adds unnecessary complexity and does not align with CloudFront's native field-level encryption capabilities.
- D. RSA Key Pair with Lambda@Edge for Encryption: Similar to Option C, this approach introduces complexity by using Lambda@Edge for encryption. It does not leverage CloudFront's built-in field-level encryption which is more streamlined for this use case.

Question 3

A solutions architect is tasked with configuring DNS settings for an existing VPC that utilizes the 10.24.34.0/24 CIDR block and Amazon Route 53 Resolver. The new requirements specify the use of private hosted zones and the assignment of public hostnames to instances with public IP addresses. The goal is to ensure accurate domain name resolution within the VPC.

Considering these requirements, the following solutions are evaluated:

A. Set Up Private Hosted Zone and Modify VPC Settings: Establish a private hosted zone and enable both the `enableDnsSupport` and `enableDnsHostnames` features for the VPC. Alter the VPC DHCP options set to specify `domain-name-servers=10.24.34.2`.

B. Associate Private Hosted Zone with VPC and Update DHCP Options: Create a private hosted zone and link it with the VPC. Enable the `enableDnsSupport` and `enableDnsHostnames` attributes for the VPC. Formulate a new VPC DHCP options set with `domain-name-servers=AmazonProvidedDNS` and associate this set with the VPC.

C. Disable DNS Support and Modify DHCP Options: Deactivate the `enableDnsSupport` attribute while activating the `enableDnsHostnames` attribute for the VPC. Generate a new VPC DHCP options set with `domain-name-servers=10.24.34.2` and link this set with the VPC.

D. Set Up Private Hosted Zone with Specific VPC and DHCP Settings: Establish a private hosted zone and associate it with the VPC. Turn on the `enableDnsSupport` feature but

disable the `enableDnsHostnames` feature for the VPC. Revise the VPC DHCP options set to include `domain-name-servers=AmazonProvidedDNS`.

The most appropriate solution is B:

- B. Associate Private Hosted Zone with VPC and Update DHCP Options: This option correctly combines the establishment of a private hosted zone with the necessary VPC settings. By activating both `enableDnsSupport` and `enableDnsHostnames`, the VPC is configured to handle DNS queries using Route 53 Resolver and assign public hostnames to instances with public IPs. Updating the DHCP options set to use `AmazonProvidedDNS` ensures that the default Amazon DNS server is used, which is required for the proper functioning of the Route 53 Resolver in the VPC.

Reasons why the other options are less ideal:

- A. Set Up Private Hosted Zone and Modify VPC Settings: Specifying a specific DNS server IP (`10.24.34.2`) in the DHCP options set is unnecessary and can lead to misconfiguration, as the VPC should use Amazon's DNS server for proper integration with Route 53.
- C. Disable DNS Support and Modify DHCP Options: Disabling `enableDnsSupport` is counterproductive as it will prevent the VPC from using Amazon's DNS servers, which are needed for resolving domain names in conjunction with Route 53 Resolver.
- D. Set Up Private Hosted Zone with Specific VPC and DHCP Settings: Disabling the `enableDnsHostnames` attribute will prevent the assignment of public hostnames to instances with public IP addresses, which is a key requirement.

Question 4

A data analytics company utilizes an Amazon Redshift cluster with reserved nodes for its operations. Recently, the cluster has been experiencing spikes in usage due to a team conducting a complex audit analysis report, leading to CPU-intensive read queries. The business requirement is to ensure that the cluster remains capable of handling both read and write queries consistently. The task is to find a solution that can manage these usage bursts in a cost-effective manner.

Here are the evaluated solutions:

A. Set Up an Amazon EMR Cluster for Complex Data Processing: Create an Amazon EMR cluster to offload the intensive data processing tasks from the Redshift cluster.

B. Use AWS Lambda for Classic Resize at High CPU Utilization: Implement an AWS Lambda function to expand the Redshift cluster using a classic resize operation when CPU usage reaches 80%, as indicated by Amazon CloudWatch.

C. Use AWS Lambda for Elastic Resize at High CPU Utilization: Deploy an AWS Lambda function to increase the Redshift cluster's capacity using an elastic resize operation upon reaching 80% CPU utilization, as monitored by CloudWatch.

D. Enable Concurrency Scaling on Amazon Redshift: Activate the Concurrency Scaling feature on the Amazon Redshift cluster to manage usage spikes.

The most suitable solution is D:

- D. Enable Concurrency Scaling on Amazon Redshift: This option is the most cost-effective and efficient for handling sporadic bursts of read query demands without impacting the cluster's ability to handle both read and write operations. Concurrency Scaling adds additional cluster capacity during demand spikes, ensuring seamless scaling without the need for manual intervention or complex configurations.

Reasons why the other options are less suitable:

- A. Set Up an Amazon EMR Cluster for Complex Data Processing: While EMR can offload processing tasks, it introduces additional complexity and potential costs. This solution might not be the most cost-effective, especially for handling temporary spikes in demand.
- B. Use AWS Lambda for Classic Resize at High CPU Utilization: Classic resize operations are time-consuming and result in downtime, which violates the requirement for the cluster to service queries at all times.
- C. Use AWS Lambda for Elastic Resize at High CPU Utilization: Although elastic resize is faster than classic resize and provides more flexibility, it is not as immediate or seamless as Concurrency Scaling and may not be as cost-effective for sporadic bursts of activity.

Question 5

A research center has recently transitioned its extensive on-premises object storage, around 1 petabyte, to Amazon S3. In this storage, each of their 100 scientists stores work-related documents in individual folders. These scientists are collectively part of a single IAM user group. However, the center's compliance officer is concerned about unauthorized access among the scientists and the requirement to track who accesses what document. The team tasked with overseeing this has limited AWS expertise and seeks an uncomplicated, low-maintenance solution.

The possible actions to address these requirements are (choose two):

A: Implement an IAM Identity Policy with Read and Write Access: Develop an IAM policy that permits read and write actions, with an added condition that the S3 paths must start with the

specific scientist's username. This policy should be applied to the IAM group of scientists.

B: Set Up AWS CloudTrail for Object-Level Event Capturing: Configure AWS CloudTrail to monitor and log every object-level interaction within the S3 bucket. The captured data should be stored in a separate S3 bucket, with Amazon Athena employed for log analysis and report generation.

C: Activate S3 Server Access Logging: Turn on server access logging for the S3 bucket, directing the logs to another designated S3 bucket. Use Amazon Athena to analyze these logs and produce the required reports.

D: Establish an S3 Bucket Policy for Group Access: Create a bucket policy granting read and write access to the S3 bucket for all members of the scientist's IAM user group.

E: Use AWS CloudTrail with Amazon CloudWatch: Configure AWS CloudTrail to record all object-level events in the S3 bucket and send this data to Amazon CloudWatch. Employ Amazon Athena with the CloudWatch connector for log analysis and report generation.

The most appropriate combination of actions is A and B:

- A: Implement an IAM Identity Policy with Read and Write Access: This action ensures that each scientist can access only their personal folder, thereby maintaining privacy and preventing unauthorized access to others' work. The conditional access based on the username in the policy effectively isolates each scientist's access.
- B: Set Up AWS CloudTrail for Object-Level Event Capturing: By capturing all interactions with the S3 bucket, this allows the research center to maintain detailed logs of who accessed or modified which documents. Storing these logs in another bucket and using Athena for analysis offers a straightforward and comprehensive solution for monitoring and reporting access patterns.

The reasons why the other options are less suitable:

- C: Activate S3 Server Access Logging: While this option does provide logging, it doesn't offer the detailed object-level tracking necessary for the research center's specific reporting requirements.
- D: Establish an S3 Bucket Policy for Group Access: This policy does not restrict scientists' access to their own folders, potentially allowing access to each other's work, which is against the center's policy.
- E: Use AWS CloudTrail with Amazon CloudWatch: This approach, although feasible, introduces unnecessary complexity by involving CloudWatch, and doesn't provide a significant advantage over option B.

Question 6

A company managing a large and growing number of AWS accounts through AWS Organizations is developing an application that relies on Docker images. These images will be stored in Amazon Elastic Container Registry (ECR), and it's essential that only accounts within the company's organization have access to them. In addition, while the company intends to keep all tagged Docker images indefinitely, it wants to automate the deletion of untagged images, retaining only the five latest ones. They seek a solution that involves minimal operational effort.

The potential solutions to consider are:

A: Set Up a Private ECR Repository with a Permissions Policy: Establish a private ECR repository and create a policy that grants access to necessary ECR operations. Include a condition that aligns the permitted operations with the company's organization ID (using the `aws:PrincipalOrgID` condition key). Implement a lifecycle policy in the repository to automatically delete untagged images beyond the five most recent ones.

B: Create a Public ECR Repository with an IAM Role: Establish a public ECR repository and create an IAM role in the ECR account. Set permissions allowing any account to assume this role if it belongs to the company's organization, as indicated by the organization ID. Implement a lifecycle policy in the repository for managing untagged images as described.

C: Private ECR Repository with Scheduled Lambda Function: Create a private ECR repository and a permissions policy allowing specific ECR operations for all accounts in the organization. Set up a daily EventBridge rule to trigger a Lambda function that will delete untagged images exceeding the count of five.

D: Public ECR Repository with Interface VPC Endpoint and Scheduled Lambda Function: Establish a public ECR repository. Configure ECR to use a VPC interface endpoint with an endpoint policy that grants necessary permissions, conditional on the account being part of the company's organization. Schedule a daily EventBridge rule to trigger a Lambda function for managing untagged images.

The most suitable solution is A:

- **A: Set Up a Private ECR Repository with a Permissions Policy**: This solution is optimal due to its operational simplicity and security. The private ECR repository ensures that the images are not publicly accessible. The permissions policy effectively restricts access to accounts within the organization, aligning with the company's security requirements. The lifecycle rule for untagged images automates image retention management, reducing manual effort.

Why the other options are less appropriate:

- B: Create a Public ECR Repository with an IAM Role: This approach exposes the Docker images to potential public access, which is a significant security concern. Managing access through an IAM role adds unnecessary complexity.
- C: Private ECR Repository with Scheduled Lambda Function: While this option offers a private repository, the need for a daily scheduled Lambda function to manage image retention adds unnecessary operational overhead and complexity.
- D: Public ECR Repository with Interface VPC Endpoint and Scheduled Lambda Function: This option similarly involves a public repository, which is not ideal for security reasons. The combination of a VPC endpoint and a scheduled Lambda function for image retention is overly complex and not as efficient as a lifecycle policy.

Question 7

A company currently automates daily snapshots of their Amazon RDS DB instances, keeping these snapshots for a week. They require a new solution to perform snapshots every six hours and retain them for a month. Moreover, they utilize AWS Organizations for managing their AWS accounts and need a unified view to monitor the health of these RDS snapshots.

The potential solutions to address this requirement are:

A: Implement AWS Backup with Cross-Account Management: Activate the cross-account management feature in AWS Backup. Develop a backup plan aligning with the specified frequency and retention requirements, utilizing tagging for DB instances. Apply this backup plan using tags. AWS Backup will then provide a centralized platform to oversee backup status.

B: Utilize Amazon RDS with a Global Snapshot Policy: Enable the cross-account management feature in Amazon RDS. Set up a global snapshot policy that adheres to the required frequency and retention schedule. Monitor backup statuses through the RDS console in the management account.

C: Use AWS CloudFormation with a Custom Lambda Function: Enable cross-account management in AWS CloudFormation. Deploy a stack set from the management account with a backup plan from AWS Backup, detailing the required frequency and retention. Create a Lambda function in the management account for backup monitoring. Establish an EventBridge rule in each account to periodically trigger this Lambda function.

D: Independent AWS Backup Configuration in Each Account: Set up AWS Backup in each individual account. Formulate an Amazon Data Lifecycle Manager policy meeting the frequency and retention criteria, targeting the DB instances. Monitor backup statuses using the Data Lifecycle Manager console in each member account.

The most suitable solution is A:

- A: Implement AWS Backup with Cross-Account Management: This approach is the most operationally efficient and straightforward, offering centralized management and monitoring of backups across all accounts. It ensures compliance with the specified backup frequency and retention period, and leverages tagging for easy application of backup policies.

Why the other options are less suitable:

- B: Utilize Amazon RDS with a Global Snapshot Policy: While this option might seem viable, Amazon RDS does not currently offer a global snapshot policy feature that can be managed across multiple accounts in AWS Organizations, making this solution infeasible.
- C: Use AWS CloudFormation with a Custom Lambda Function: This solution introduces unnecessary complexity and operational overhead. Managing backups via CloudFormation stack sets and custom Lambda functions for monitoring is less streamlined compared to using AWS Backup's built-in features.
- D: Independent AWS Backup Configuration in Each Account: Configuring AWS Backup separately in each account and using Amazon Data Lifecycle Manager increases operational complexity and does not provide a consolidated view for monitoring, as required by the company.

Question 8

company operating within AWS Organizations has a setup involving multiple accounts, governed by various policy types including Service Control Policies (SCPs), resource-based policies, identity-based policies, trust policies, and session policies. The task at hand is to enable an IAM user in one account (Account A) to assume a role in another account (Account B).

The potential steps to accomplish this are:

A: Adjust Account A's SCP to Permit the Action: Modify the SCP associated with Account A to allow the action of assuming the role in Account B.

B: Alter Resource-Based Policies to Permit the Action: Change resource-based policies to enable the action of the IAM user assuming the role.

C: Modify Identity-Based Policy on User in Account A: Update the identity-based policy attached to the IAM user in Account A to grant permission to assume the role in Account B.

D: Update Identity-Based Policy on User in Account B: Change the identity-based policy on a user in Account B to allow the action.

E: Adjust Trust Policy on Target Role in Account B: Modify the trust policy of the target role in Account B to trust the IAM user from Account A, thereby allowing the role assumption.

F: Configure Session Policy for the Action: Establish a session policy that permits the role assumption action and can be passed programmatically through the GetSessionToken API operation.

The correct combination of steps is C, E, F:

- C: Modify Identity-Based Policy on User in Account A: This is a crucial step as the identity-based policy on the IAM user in Account A needs to explicitly allow the action of assuming the specified role in Account B.
- E: Adjust Trust Policy on Target Role in Account B: The trust policy on the role in Account B must be configured to establish trust with the IAM user from Account A. This step is essential to enable cross-account role assumption.
- F: Configure Session Policy for the Action: A session policy can be used to further refine permissions for the session when assuming the role. This step ensures that the permissions are appropriately scoped during the session established by the assumed role.

Why the other options are incorrect:

- A: Adjust Account A's SCP to Permit the Action: SCPs are used to manage permissions in all accounts within an AWS Organization. While SCPs can restrict actions, allowing an action in an SCP is not sufficient on its own for cross-account role assumption.
- B: Alter Resource-Based Policies to Permit the Action: Resource-based policies are attached to resources, not IAM users or roles. They are not directly involved in granting an IAM user in one account permission to assume a role in another account.
- D: Update Identity-Based Policy on User in Account B: The identity-based policy on a user in Account B is not relevant in this scenario as we're focusing on granting permissions to a user in Account A to assume a role in Account B.

Question 9

A company is looking to integrate its existing NFS-compatible on-premises file storage system with Amazon S3 for backup purposes. The organization requires a system that supports NFS and aims to transition these backups to an archival storage option after 5 days. In case of disaster recovery, the company is prepared to endure a retrieval time of a few days for these archived files. The goal is to implement the most cost-effective solution to meet these specific needs.

Here are the potential solutions:

A: Implement an AWS Storage Gateway file gateway linked to an S3 bucket. Transfer files from the on-premises storage to this file gateway. Set up an S3 Lifecycle policy to shift the files to S3 Standard-Infrequent Access (S3 Standard-IA) after 5 days.

B: Set up an AWS Storage Gateway volume gateway connected to an S3 bucket. Relocate files from the on-premises storage to the volume gateway. Establish an S3 Lifecycle rule to transition the files to S3 Glacier Deep Archive post 5 days.

C: Implement an AWS Storage Gateway tape gateway associated with an S3 bucket. Transfer files from the on-premises storage to the tape gateway. Configure an S3 Lifecycle policy to move the files to S3 Standard-Infrequent Access (S3 Standard-IA) after 5 days.

D: Deploy an AWS Storage Gateway file gateway connected to an S3 bucket. Move files from the on-premises storage to this file gateway. Formulate an S3 Lifecycle rule to transfer the files to S3 Glacier Deep Archive after 5 days.

The correct solution is D:

- **D: Deploy an AWS Storage Gateway file gateway linked to an S3 bucket, and transfer files from on-premises to the file gateway with an S3 Lifecycle rule for Glacier Deep Archive**: This option is the most appropriate as it leverages the file gateway to provide NFS support, enabling seamless integration with the existing file storage system. The S3 Lifecycle rule moving files to S3 Glacier Deep Archive after 5 days meets the company's archival requirements and ensures cost-effective long-term storage, considering the company's willingness to wait a few days for data retrieval in disaster recovery scenarios.

Reasons why the other options are less suitable:

- **A: Storage Gateway file gateway to S3 Standard-IA**: While this option supports NFS and uses a file gateway, transitioning to S3 Standard-IA is not as cost-effective for long-term archival as S3 Glacier Deep Archive. This option might result in higher costs compared to Glacier Deep Archive, especially for data not accessed frequently.
- **B: Volume gateway to S3 Glacier Deep Archive**: The volume gateway is typically used for block storage and might not be the most straightforward solution for integrating with an NFS-based file storage system. This option could introduce unnecessary complexity.
- **C: Tape gateway to S3 Standard-IA**: The tape gateway simulates a virtual tape library (VTL) and is not the ideal choice for direct NFS integration. Moreover, moving files to S3 Standard-IA does not provide the same cost benefits as Glacier Deep Archive for archival purposes.

Question 10

A company operates its application using Amazon EC2 instances and AWS Lambda functions. The EC2 instances have a consistent load pattern, whereas the Lambda functions experience fluctuating and unpredictable load. The application also incorporates Amazon MemoryDB for Redis as its caching solution. The primary goal is to identify a strategy that effectively reduces the company's monthly expenses.

Here are the potential solutions:

A: Opt for an EC2 instance Savings Plan to accommodate the costs associated with the EC2 instances. Implement a Compute Savings Plan targeting the baseline utilization of the Lambda functions. Additionally, invest in reserved nodes for the MemoryDB cache nodes.

B: Acquire a Compute Savings Plan for the EC2 instances. Secure reserved concurrency arrangements for the Lambda functions, aligning with expected usage levels. Also, invest in reserved nodes for the MemoryDB cache nodes.

C: Purchase a single Compute Savings Plan that covers the anticipated expenses for the EC2 instances, Lambda functions, and MemoryDB cache nodes.

D: Obtain a Compute Savings Plan for both the EC2 instances and MemoryDB cache nodes. Secure reserved concurrency for the Lambda functions based on anticipated usage.

The most suitable solution is A:

- **A: EC2 Savings Plan, Compute Savings Plan for Lambda, and Reserved Nodes for MemoryDB**: This approach is the most effective in minimizing costs. An EC2 Savings Plan is ideal for the stable load of the EC2 instances, providing flexibility while offering lower prices. A Compute Savings Plan for Lambda is appropriate for managing the variable load of the Lambda functions, allowing for cost savings on the predictable portion of their use. Purchasing reserved nodes for MemoryDB ensures cost efficiency for the caching layer, which likely has a steady utilization pattern.

Reasons why the other options are less suitable:

- **B: Compute Savings Plan for EC2, Lambda Reserved Concurrency, Reserved Nodes for MemoryDB**: While this option does address cost savings, a Compute Savings Plan is generally more flexible but might not offer the same cost benefits for EC2 instances with a stable load as an EC2 Savings Plan would.
- **C: Single Compute Savings Plan for All Services**: A single Compute Savings Plan might not yield the optimal savings across varied usage patterns of EC2, Lambda, and MemoryDB, as it does not account for the distinct utilization patterns and pricing models of these services.

- **D: Compute Savings Plan for EC2 and MemoryDB, Lambda Reserved Concurrency**: This combination does not utilize the EC2 Savings Plan, which could be more cost-effective for the consistent load of the EC2 instances. Moreover, MemoryDB might benefit more from reserved nodes rather than being included in a Compute Savings Plan.

Question 11

A gaming company is launching a new online game globally and intends to host it on Amazon EC2 instances across three AWS Regions: us-east-1, eu-west-1, and ap-southeast-1. Critical game features such as leaderboards, player inventory, and event status need to be accessible and synchronized across these regions. The system should scale to manage the load from all regions and direct users to the region with the lowest latency. The goal is to find a solution that requires minimal operational effort.

Here are the proposed solutions:

A: Utilize EC2 Spot Fleet connected to a Network Load Balancer (NLB) in each region. Employ AWS Global Accelerator pointing to the NLBs and Route 53 for latency-based routing to the Global Accelerator IP. Store game metadata in Amazon RDS for MySQL with read replicas across regions.

B: Establish Auto Scaling groups for EC2 instances linked to an NLB in each region. Use Amazon Route 53 with geoproximity routing for each region's NLB. Store game metadata in MySQL on EC2 instances, with replication set up between regions.

C: Set up Auto Scaling groups for EC2 instances, attaching them to an NLB in each region. Use Amazon Route 53 with latency-based routing for the NLBs in each region. Store game metadata in an Amazon DynamoDB global table.

D: Implement EC2 Global View with EC2 instances deployed in each region and connected to NLBs. Deploy DNS servers on EC2 instances in each region with custom logic for low-latency redirection. Store game metadata in an Amazon Aurora global database.

The most suitable solution is **C:**

- **C: Auto Scaling Groups, NLB, Route 53 with Latency-Based Routing, DynamoDB Global Table**: This option effectively meets the requirements for global availability and scalability with minimal overhead. Auto Scaling groups ensure that the game can scale to handle the load. NLBs distribute traffic within each region. Route 53 with latency-based routing efficiently directs users to the nearest region, reducing latency. DynamoDB global tables provide a fully managed, multi-region, and multi-active database solution, ideal for synchronizing game metadata across regions without the need for complex database replication strategies.

Reasons why the other options are less suitable:

- A: Spot Fleet, Global Accelerator, Route 53, RDS for MySQL with Read Replicas: While this option does provide global scalability, managing RDS for MySQL with read replicas across regions can be operationally complex and may not offer real-time synchronization.
- B: Auto Scaling, Geoproximity Routing, MySQL on EC2 with Replication: This solution adds operational complexity due to the need to manage database replication across regions manually. Geoproximity routing may not always result in the lowest latency connections compared to latency-based routing.
- D: EC2 Global View, Custom DNS Servers, Aurora Global Database: Implementing custom DNS logic adds unnecessary complexity. Moreover, EC2 Global View is not a standard AWS service, and managing an Aurora global database might be more complex compared to using DynamoDB global tables.

Question 12

A business is implementing a third-party firewall solution from the AWS Marketplace to oversee and safeguard its AWS environment's outbound internet traffic. The solution will be integrated into a central shared services VPC, and it's essential to ensure that all external traffic is routed through these firewall appliances. The solutions architect is tasked with devising a deployment strategy that emphasizes reliability and reduces the time it takes to switch between firewall appliances within a single AWS Region. Routing has already been established from the shared services VPC to other VPCs in the company's setup.

The following steps are proposed for this deployment:

A: Deploy a pair of firewall appliances in the shared services VPC, placing one in each of two different Availability Zones.

B: Set up a Network Load Balancer within the shared services VPC. Create a target group and associate it with the Network Load Balancer. Then, add each of the firewall appliance instances to this target group.

C: Establish a Gateway Load Balancer in the shared services VPC. Attach a new target group to this Gateway Load Balancer and include each of the firewall appliance instances in the target group.

D: Implement a VPC interface endpoint. Modify the route table in the shared services VPC to route traffic that enters from other VPCs through this new endpoint.

E: Deploy two firewall appliances within the same Availability Zone in the shared services VPC.

F: Create a VPC Gateway Load Balancer endpoint. Adjust the route table in the shared services VPC to make this new endpoint the next hop for traffic coming into the shared services VPC from other VPCs.

The most appropriate combination of steps is A, C, F:

- - A. Deploy Two Firewall Appliances in Separate Availability Zones:
    - Why It's Right: Placing firewall appliances in different Availability Zones ensures high availability. Each Availability Zone is an isolated location within a region, and deploying in separate zones protects against failures in one zone. This setup guarantees that if one appliance or its hosting zone fails, the other can take over, thus ensuring continuous protection and minimal service disruption.
- C. Create a New Gateway Load Balancer and Target Group:
    - Why It's Right: A Gateway Load Balancer is specifically designed for scenarios like these. It's a fully managed service that operates at the third layer of the OSI model and is designed to handle traffic routing and distribution for network appliances, including firewalls. It simplifies deployment and management of these appliances and automatically handles the distribution of traffic across multiple appliances, ensuring efficient load balancing and failover.
- F. Create a VPC Gateway Load Balancer Endpoint:
    - Why It's Right: By adding a Gateway Load Balancer endpoint to the VPC's route table, the company ensures that all outbound traffic from the VPC is routed through the firewall appliances. This setup centralizes traffic inspection and allows for consistent enforcement of security policies, improving overall network security posture.

Incorrect Answers:

- B. Network Load Balancer with Target Group:
    - Why It's Wrong: While Network Load Balancers are effective for distributing TCP/UDP traffic, they are not specialized for handling traffic meant for network appliances like firewalls. Gateway Load Balancers are more suitable for this task as they are designed for seamless integration with third-party virtual appliances and for handling traffic at the network layer.
- D. VPC Interface Endpoint:
    - Why It's Wrong: VPC Interface Endpoints are used for privately connecting VPCs to supported AWS services without using the public internet. They are not designed for routing traffic through network appliances like firewalls. This option doesn't align with the need to route outbound internet traffic through firewall appliances.
- E. Deploy Appliances in the Same Availability Zone:

- **Why It's Wrong:** Deploying both appliances in the same Availability Zone doesn't provide the redundancy and high availability required for critical security infrastructure. If the Availability Zone experiences issues, both appliances could become unreachable, leaving the network unprotected.

Question 13

A solutions architect is tasked with moving a legacy application, currently running on two servers behind a load balancer on-premises, to AWS. This application is unique in that it requires a license file linked to the MAC address of the server's network adapter. Additionally, the application uses configuration files with static IP addresses for database access, and the software vendor takes about 12 hours to issue new license files. The goal is to create a highly available architecture for the application servers in AWS while adhering to these specific requirements.

Possible Answers (choose two):

A. Create a group of Elastic Network Interfaces (ENIs), obtain license files for each ENI from the vendor, and store these files in Amazon S3. Develop a bootstrap script to download a license file and attach the corresponding ENI to an Amazon EC2 instance.

B. Set up a group of ENIs and secure license files for each ENI from the vendor, keeping these files on an Amazon EC2 instance. Create an AMI from this instance and use this AMI for all future EC2 instances.

C. Develop a bootstrap script to request a new license file from the vendor upon instance initiation. Once received, apply the license file to an Amazon EC2 instance.

D. Modify the bootstrap script to retrieve the database server IP address from AWS Systems Manager Parameter Store and insert this value into local configuration files.

E. Alter an Amazon EC2 instance to incorporate the database server IP address in its configuration files and create a new AMI to be used for future EC2 instances.

Correct Answers:

- **A:** This approach is ideal for maintaining high availability. It allows for the dynamic attachment of ENIs with pre-approved licenses to any EC2 instance, ensuring that new instances can be quickly launched with valid licenses, crucial for uninterrupted operation.
- **D:** Utilizing AWS Systems Manager Parameter Store for storing and retrieving the database IP address streamlines the configuration management process. This approach allows for centralized and easily updatable configuration without the need to individually update instances or AMIs.

Incorrect Answers:

- B: While this solution might initially seem convenient, it lacks the flexibility and scalability required for a highly available architecture. Any change in the license files or the need for additional licenses would necessitate the creation of a new AMI, introducing delays and potential downtime.
- C: Directly requesting new license files from the vendor for each instance launch is impractical due to the 12-hour wait time, which would severely impact the ability to scale quickly or recover the application in case of failure.
- E: Embedding the database IP address directly in the EC2 instance's configuration files and creating a new AMI for future instances is not an efficient approach. It would require creating a new AMI every time there's a change in the database IP address, leading to operational complexities and potential service interruptions.

Question 14

A company operating a sales reporting application in the United States using Amazon API Gateway, AWS Lambda, and Amazon RDS for MySQL is expanding to Europe. The frontend is hosted on Amazon S3 and accessed via Amazon CloudFront. Amazon Route 53, configured with simple routing, directs traffic to the API Gateway. The company plans to cater to its new European audience, where most database traffic is read-only. They have established an API Gateway API and Lambda functions in Europe and now seek a solution to minimize latency for European users accessing reports.

Here are the proposed solutions:

A: Use AWS Database Migration Service (AWS DMS) for full-load replication to the European database and change Route 53 to latency-based routing for the API Gateway API.

B: Implement AWS DMS with full-load plus change data capture (CDC) replication and update Route 53 to geolocation routing for the API Gateway API.

C: Configure a cross-region read replica for the RDS database in Europe and change Route 53 to latency-based routing for the API Gateway API.

D: Set up a cross-region read replica for the RDS database in Europe and modify Route 53 to geolocation routing for the API Gateway API.

The most suitable solution is C:

- C: Cross-Region Read Replica, Latency-Based Routing in Route 53: This solution is optimal as it allows for efficient handling of the predominant read-only traffic by placing a read replica closer to European users. Utilizing latency-based routing in Route 53

ensures that requests are directed to the API Gateway API with the lowest latency, providing a responsive experience for users downloading reports.

Reasons why the other options are less suitable:

- A: Full-Load Replication with DMS, Latency-Based Routing: While DMS can replicate the primary database, this approach doesn't provide continuous synchronization. Latency-based routing is appropriate, but without ongoing data replication, the European database may have outdated data.
- B: Full-Load Plus CDC Replication with DMS, Geolocation Routing: DMS with CDC ensures continuous data synchronization. However, geolocation routing in Route 53 may not always provide the lowest latency, as it routes based on geographic location rather than network latency.
- D: Cross-Region Read Replica, Geolocation Routing in Route 53: The read replica is beneficial, but geolocation routing might not always minimize latency as effectively as latency-based routing, as it bases routing decisions on geographic location rather than network latency.

Question 15

A software company is developing a system for automatically deploying and removing short-lived testing environments in AWS. Each environment includes an Amazon EC2 instance within an Auto Scaling group. These environments need to connect to a central server in an on-premises data center for reporting test results. The company has set up a transit gateway with a VPN to the on-premises network. The goal is to enable the creation and deletion of these test environments without manual intervention, with minimal operational complexity.

Here are the evaluated solutions:

A: Use AWS CloudFormation to design a template with a transit gateway attachment and routing configurations. Implement a CloudFormation stack set for each VPC in the account, and create a new VPC for every test environment.

B: Utilize a single VPC dedicated to test environments, incorporating a transit gateway attachment and routing configurations. Deploy all test environments within this VPC using AWS CloudFormation.

C: Establish a new Organizational Unit (OU) in AWS Organizations specifically for testing. Develop a CloudFormation template encompassing a VPC, networking resources, a transit gateway attachment, and routing configurations. Utilize CloudFormation StackSets for deployments across each account under this testing OU, creating a new account for each test environment.

D: Transform the EC2 instances of test environments into Docker images. Use CloudFormation to set up an Amazon Elastic Kubernetes Service (Amazon EKS) cluster in a new VPC, along with a transit gateway attachment and routing configurations. Manage test environment deployments and lifecycles through Kubernetes.

The most suitable solution is B:

- B: Single VPC with Transit Gateway Attachment, AWS CloudFormation: This approach effectively meets the requirements for simplicity and operational efficiency. By using a single VPC for all test environments, the company avoids the complexity of managing multiple VPCs or AWS accounts. The inclusion of a transit gateway attachment and routing configurations in the VPC ensures seamless connectivity to the on-premises server. Using CloudFormation for deployment streamlines the process of creating and deleting environments as needed.

Reasons why the other options are less suitable:

- A: Multiple VPCs with Stack Sets: Creating a new VPC for each test environment introduces unnecessary complexity and overhead. Managing multiple VPCs can be operationally challenging, especially for short-lived test environments.
- C: New OU with Multiple Accounts: Establishing a new account for each test environment is an over-complication. This method involves significant overhead in account management and may not be practical for short-lived environments.
- D: Docker Images and Amazon EKS: While containerization offers benefits, converting EC2 instances to Docker images and managing them through an EKS cluster is an overly complex solution for the company's needs. This approach would require additional expertise in Kubernetes and container management, which may not align with the company's current capabilities or objectives.

Question 16

A company is enhancing its AWS-deployed API to support an active-active configuration across multiple AWS Regions. The current setup includes Amazon API Gateway with a Regional API endpoint, an AWS Lambda function, external API data retrieval, data storage in an Amazon DynamoDB global table, and an API key stored in AWS Secrets Manager, encrypted with a customer-managed AWS Key Management Service (AWS KMS) key. The goal is to modify the API components for multi-region support while maintaining minimal operational complexity.

Here are the proposed modifications:

A: Implement the API in multiple regions and configure Amazon Route 53 with custom domain names for each regional API endpoint. Utilize Route 53's multivalue answer routing policy for

traffic distribution.

B: Establish a new KMS multi-region customer-managed key and create corresponding KMS customer-managed replica keys in each target region.

C: Replicate the existing Secrets Manager secret to additional regions and associate each replicated secret with the appropriate regional KMS key.

D: Generate a new AWS-managed KMS key in each target region or convert the existing key into a multi-region key, then use this multi-region key across regions.

E: Create a new Secrets Manager secret in each target region and manually copy the secret value from the existing region to the new secrets.

F: Adjust the Lambda function deployment process to replicate deployments across the target regions. Enable the multi-region option for the existing API and link it to the Lambda function deployed in each region.

The most suitable combination is A, B, C:

- A: Deploy API to Multiple Regions with Route 53 Multivalue Answer Routing: This approach enables the distribution of traffic across regional API endpoints, offering high availability and reduced latency. The multivalue answer routing policy in Route 53 effectively handles traffic routing to the closest regional endpoint.
- B: KMS Multi-Region Customer Managed Key: Creating a multi-region KMS key allows for seamless encryption and decryption of the API key across multiple regions, ensuring consistency and security without the need to manage multiple independent keys.
- C: Replicate Secrets Manager Secret with Regional KMS Keys: Replicating the secret in Secrets Manager across regions and linking it with the appropriate regional KMS key ensures that the API key is available and securely encrypted in each region.

Reasons why the other options are less suitable:

- D: New AWS-Managed KMS Key or Conversion to Multi-Region Key: Creating new AWS-managed KMS keys or converting an existing key adds unnecessary complexity compared to directly creating a multi-region customer-managed key.
- E: New Secrets Manager Secret in Each Region: Manually copying the secret value to new secrets in each region is operationally intensive and prone to errors, unlike the replication approach that automates the process.
- F: Modify Lambda Deployment and Enable Multi-Region API Option: This option is not feasible as AWS API Gateway does not offer a "multi-region option" for APIs. Deploying Lambda functions across regions requires a different approach, such as using

infrastructure as code or manual replication, not linked to the API Gateway's configuration.

Question 17

An e-commerce business currently operates its web application and MySQL database on a single server in its local data center. To cater to an expanding customer base due to increased marketing activities, the business plans to transition its application and database to AWS. The goal is to enhance the architecture's dependability.

Considered solutions are:

A: Shift the database to Amazon RDS MySQL with Multi-AZ deployment. Host the application on Amazon EC2 instances within an Auto Scaling group, paired with an Application Load Balancer, and manage sessions using Amazon Neptune.

B: Transition to Amazon Aurora MySQL for the database. Utilize an Auto Scaling group of Amazon EC2 instances with an Application Load Balancer for the application, and use Amazon ElastiCache for Redis for session storage.

C: Opt for Amazon DocumentDB with MongoDB compatibility for the database. Run the application on EC2 instances within an Auto Scaling group, behind a Network Load Balancer, and employ Amazon Kinesis Data Firehose for session management.

D: Migrate to an Amazon RDS MariaDB Multi-AZ DB instance for the database. Deploy the application on EC2 instances within an Auto Scaling group, using an Application Load Balancer, and store sessions in Amazon ElastiCache for Memcached.

The most appropriate solution is B:

- B: Amazon Aurora MySQL, EC2 Auto Scaling, Application Load Balancer, ElastiCache for Redis: This setup provides high reliability and performance. Aurora MySQL offers enhanced durability and availability compared to standard RDS, making it ideal for critical applications. The use of EC2 Auto Scaling ensures that the application can handle varying loads efficiently. The Application Load Balancer distributes traffic optimally across EC2 instances. ElastiCache for Redis is a robust in-memory data store, suitable for managing user sessions with high performance.

Reasons the other options are less suitable:

- A: RDS MySQL Multi-AZ, Neptune for Sessions: While RDS MySQL with Multi-AZ deployment is reliable, Neptune is primarily a graph database and is not typically used for session management, making this combination less optimal for the described scenario.

- C: DocumentDB, Network Load Balancer, Kinesis Data Firehose: DocumentDB, being a MongoDB-compatible database, might not be the direct choice for a MySQL-based application without significant modifications. Also, Kinesis Data Firehose is not commonly used for session management.
- D: RDS MariaDB Multi-AZ, ElastiCache for Memcached: Although RDS MariaDB with Multi-AZ deployment is a robust choice, MariaDB may require changes in the application initially designed for MySQL. Additionally, Redis generally offers more features and flexibility compared to Memcached for session management.

Question 18

A business needs a secure method to enable users to remotely access Amazon EC2 Windows instances in a VPC, with user management tied to its existing on-premises Active Directory. The users will connect over the internet, and the company possesses the necessary equipment to set up an AWS Site-to-Site VPN.

The proposed solutions are:

A: Use AWS Directory Service for Microsoft Active Directory to create a managed Active Directory and establish a trust relationship with the on-premises Active Directory. Set up an EC2 bastion host in the VPC, joined to the domain, for RDP access to the target instances.

B: Integrate AWS IAM Identity Center (AWS Single Sign-On) with the on-premises Active Directory via AWS Directory Service's AD Connector. Use AWS Systems Manager with configured permission sets for user groups to facilitate RDP access to target instances through Systems Manager Fleet Manager.

C: Create a VPN between the on-premises network and the VPC. Join the target instances to the on-premises Active Directory domain across the VPN, and configure RDP access to the instances through the VPN.

D: Deploy AWS Directory Service for Microsoft Active Directory and establish a trust with the on-premises Active Directory. Use an AWS Quick Start to deploy a Remote Desktop Gateway in AWS, joined to the domain, for RDP access to the target instances.

The most cost-effective and suitable solution is B:

- B: AWS IAM Identity Center (AWS Single Sign-On) with AWS Directory Service's AD Connector and AWS Systems Manager: This solution efficiently integrates with the existing on-premises Active Directory without additional infrastructure. AWS IAM Identity Center offers centralized user management, and Systems Manager Fleet Manager provides secure and managed RDP access to EC2 instances. This approach minimizes

the need for additional EC2 instances or infrastructure, reducing operational overhead and costs.

Reasons the other options are less suitable:

- A: EC2 Bastion Host with Managed Active Directory: Although this solution provides secure RDP access, it involves the additional cost and complexity of maintaining an EC2 bastion host, making it less cost-effective compared to using Systems Manager.
- C: VPN with RDP Access: This option requires a VPN setup and does not provide centralized management of RDP access through AWS services, leading to potential security and management challenges.
- D: Remote Desktop Gateway with Managed Active Directory: Deploying a Remote Desktop Gateway adds unnecessary complexity and cost, especially when AWS services like Systems Manager offer a more integrated and cost-effective solution for secure RDP access.

Question 19

A company's compliance review found that some of their Amazon Elastic Block Store (Amazon EBS) volumes created within their AWS account weren't encrypted. To address this, a solutions architect is tasked with ensuring all new EBS volumes are encrypted by default with minimal effort.

The potential solutions are:

A: Implement an Amazon EventBridge rule to monitor the creation of unencrypted EBS volumes. Trigger an AWS Lambda function to delete any non-compliant volumes.

B: Utilize AWS Audit Manager with data encryption capabilities.

C: Set up an AWS Config rule to identify newly created EBS volumes. Use AWS Systems Manager Automation to encrypt these volumes.

D: Enable EBS encryption by default in every AWS Region.

The most straightforward and effective solution is D:

- D: Enable EBS encryption by default in all AWS Regions: This approach ensures all new EBS volumes are encrypted automatically in every region without requiring additional steps or manual intervention. It's a simple setting change that applies globally, aligning with the goal of minimal effort.

Reasons why the other options are less suitable:

- A: EventBridge rule and Lambda function: While this method can enforce encryption, it involves deleting non-compliant volumes, which could lead to data loss. It's also more complex and requires ongoing management of the rule and function.
- B: AWS Audit Manager: Audit Manager is designed for auditing and compliance checks but does not directly enforce encryption of EBS volumes. It would inform about non-compliance but wouldn't automatically resolve the issue.
- C: AWS Config rule with Systems Manager Automation: This solution also ensures encryption but is more complicated than simply enabling default encryption. It involves creating and managing additional AWS services and automation workflows.

Question 20

A research company conducting daily simulations in the AWS Cloud utilizes several hundred Amazon EC2 instances running on Amazon Linux 2. Occasionally, simulations become unresponsive, necessitating SSH intervention by a cloud operations engineer. The company's policy mandates unique SSH keys for each EC2 instance and requires all SSH connections to be logged in AWS CloudTrail. The task is to devise a method that aligns with these requirements.

The available options are:

A: Set up each EC2 instance with an individual SSH key, stored in AWS Secrets Manager. Create an IAM policy allowing engineers to retrieve SSH keys via Secrets Manager, enabling them to connect using any SSH client.

B: Use AWS Systems Manager to generate and apply a unique SSH key on each EC2 instance. Create an IAM policy allowing engineers to execute Systems Manager documents and set up SSH keys, connecting via any SSH client subsequently.

C: Deploy EC2 instances without predefined SSH keys. Implement EC2 Instance Connect for each instance. Develop an IAM policy permitting engineers to use the SendSSHPublicKey action. Guide engineers to connect via a browser-based SSH client from the EC2 console.

D: Employ AWS Secrets Manager for storing SSH keys, with a Lambda function creating and updating keys daily via AWS Systems Manager Session Manager. Configure Secrets Manager for automatic daily key rotation. Instruct engineers to retrieve SSH keys from Secrets Manager for SSH connections.

The most appropriate solution is C:

- C: EC2 Instance Connect, No Predefined SSH Keys, IAM Policy for SendSSHPublicKey Action: This approach adheres to the company's policy by not requiring a preconfigured SSH key for each instance. EC2 Instance Connect generates a temporary SSH key for

each session, meeting the unique key requirement. Connections are logged in AWS CloudTrail, ensuring compliance. This method simplifies operations as engineers can connect directly through the EC2 console without managing SSH keys manually.

Reasons why the other options are less optimal:

- **A: Individual SSH Keys Stored in Secrets Manager:** While this method ensures unique SSH keys, managing individual keys for hundreds of instances via Secrets Manager could become unwieldy and time-consuming.
- **B: Systems Manager for Unique SSH Key Setup:** Generating and setting unique SSH keys through Systems Manager adds operational complexity and might not provide the necessary audit trail in CloudTrail for SSH connections.
- **D: Secrets Manager with Lambda for SSH Key Rotation:** This solution introduces unnecessary complexity with daily key rotation and Lambda functions. It also doesn't guarantee logging of each SSH session in CloudTrail, as required by company policy.

Question 21

A company relocating its mobile banking applications to Amazon EC2 instances in a Virtual Private Cloud (VPC) requires these applications to resolve DNS requests to an on-premises Active Directory domain. This Active Directory domain operates in the company's data center, which is connected to AWS through Direct Connect. The challenge is to establish a DNS resolution mechanism between the VPC-hosted applications and the on-premises Active Directory with minimal administrative effort.

The potential solutions are:

A: Set up EC2 instances as caching DNS servers in the VPC across two Availability Zones. These servers will handle DNS queries from the application servers in the VPC.

B: Establish an Amazon Route 53 private hosted zone and configure Name Server (NS) records to direct to the on-premises DNS servers.

C: Use Amazon Route 53 Resolver to create DNS endpoints. Implement conditional forwarding rules for resolving DNS namespaces between the on-premises data center and the VPC.

D: Deploy a new Active Directory domain controller in the VPC, establishing a bidirectional trust with the existing on-premises Active Directory domain.

The most effective solution is C:

- **C: Amazon Route 53 Resolver with Conditional Forwarding Rules:** This approach is highly efficient and requires minimal administrative overhead. Route 53 Resolver

facilitates seamless DNS query resolution between AWS and on-premises environments. Conditional forwarding rules enable specific DNS queries to be directed to the appropriate on-premises or cloud DNS servers, ensuring accurate resolution of the Active Directory domain requests.

Reasons why other options are less suitable:

- A: EC2 Instances as Caching DNS Servers: While this can technically resolve DNS queries, it introduces significant administrative overhead. Managing EC2 instances as DNS servers requires manual setup, maintenance, and scaling, which is more complex and less efficient compared to using a managed service like Route 53 Resolver.
- B: Route 53 Private Hosted Zone with NS Records: This solution doesn't directly address the need to resolve DNS requests to an on-premises Active Directory domain. It is more suited for managing DNS within AWS and doesn't efficiently bridge the DNS resolution between AWS and on-premises environments.
- D: New Active Directory Domain Controller in VPC: Setting up an additional domain controller in the VPC adds unnecessary complexity and maintenance. It's an over-engineered solution for the problem of DNS resolution and goes beyond the requirements of simple DNS query handling.

Question 22

A company specializing in environmental data collection has deployed sensors across a city to gather continuous streams of data. The data, available in JSON format, needs to be sent in real-time to a database that can handle dynamic schemas. The company is looking for an AWS-based solution that can efficiently handle this data flow and storage requirement.

The possible solutions are:

A: Implement Amazon Kinesis Data Firehose to transfer the data to Amazon Redshift.

B: Utilize Amazon Kinesis Data Streams for relaying the data to Amazon DynamoDB.

C: Employ Amazon Managed Streaming for Apache Kafka (Amazon MSK) to route the data to Amazon Aurora.

D: Deploy Amazon Kinesis Data Firehose for directing the data to Amazon Keyspaces (for Apache Cassandra).

The optimal solution is B:

- B: Amazon Kinesis Data Streams to Amazon DynamoDB: This combination is ideal for the company's need to process streaming data in real-time and store it in a schema-less database. Kinesis Data Streams is designed to handle real-time data streaming, and

DynamoDB, being a NoSQL database, can efficiently store data without a fixed schema. This setup provides the scalability and flexibility required for handling varying data structures from environmental sensors.

Reasons why the other options are not as suitable:

- A: Kinesis Data Firehose to Amazon Redshift: Redshift is a data warehousing service that's more suited for analytical processing rather than real-time data streaming and storage. It also requires predefined schemas, which doesn't align with the company's requirement for a schema-less database.
- C: Amazon MSK to Amazon Aurora: While Amazon MSK is a robust service for handling streaming data, Aurora is a relational database that requires predefined schemas. This setup doesn't meet the need for a schema-less data storage solution.
- D: Kinesis Data Firehose to Amazon Keyspaces: Although Amazon Keyspaces is a scalable NoSQL database service, using Kinesis Data Firehose for this purpose is not as direct and efficient as using Kinesis Data Streams, especially when considering the real-time data processing requirement.

Question 23

A company is transitioning its legacy application, which currently uses MongoDB for its key-value store, from an on-premises setup to AWS. The company's policies dictate that all Amazon EC2 instances must operate within a private subnet and have no direct internet access. Additionally, all communications between the application and the database must be encrypted. The database also needs to be scalable in response to varying demands.

The possible solutions are:

A: Migrate to Amazon DocumentDB (with MongoDB compatibility), using tables with Provisioned IOPS volumes, and connect via the instance endpoint.

B: Shift to Amazon DynamoDB, setting up tables with on-demand capacity, and establish connectivity through a gateway VPC endpoint.

C: Transition to Amazon DynamoDB, creating tables with on-demand capacity, and use an interface VPC endpoint for connectivity.

D: Switch to Amazon DocumentDB (with MongoDB compatibility), using tables with Provisioned IOPS volumes, and connect through the cluster endpoint.

The most fitting solution is B:

- B: Amazon DynamoDB with On-Demand Capacity and Gateway VPC Endpoint: This setup aligns with the company's requirements. DynamoDB is a fully managed NoSQL

database service that can easily scale based on demand. Using on-demand capacity ensures that the database scales automatically to accommodate varying workloads. The gateway VPC endpoint allows secure, private connections to DynamoDB from within the private subnet, adhering to the no internet access policy, and ensuring encrypted communication.

Reasons why the other options are less appropriate:

- **A and D: Amazon DocumentDB with Provisioned IOPS:** While DocumentDB provides MongoDB compatibility, using it in this scenario might not offer the same key-value store capabilities as native MongoDB or DynamoDB. Additionally, the requirement for encryption and no internet connectivity might be better served by DynamoDB with a VPC endpoint.
- **C: DynamoDB with On-Demand Capacity and Interface VPC Endpoint:** Although DynamoDB is a suitable choice, the use of an interface VPC endpoint is not as straightforward and cost-effective as using a gateway VPC endpoint. Gateway VPC endpoints do not incur additional charges for data processing or hourly endpoint charges, making them more cost-effective for the company's use case.

Question 24

A business currently operates an application on Amazon EC2 within AWS, using a MongoDB database set up in a replica set configuration. This database is hosted on-premises and connects to the AWS environment via an AWS Direct Connect link. The task is to transition this MongoDB database to Amazon DocumentDB, which is compatible with MongoDB, while ensuring a smooth migration.

The potential strategies are:

A: Set up a cluster of EC2 instances, install MongoDB Community Edition, and configure continuous synchronous replication with the existing on-premises MongoDB database.

B: Utilize AWS Database Migration Service (AWS DMS) to establish a replication instance. Create a source endpoint for the on-premises MongoDB using change data capture (CDC) and a target endpoint for Amazon DocumentDB. Execute a DMS migration task.

C: Implement a data transfer pipeline using AWS Data Pipeline, designating data nodes for both the on-premises MongoDB and Amazon DocumentDB. Schedule regular execution of this data pipeline.

D: Employ AWS Glue crawlers to create a source endpoint for the on-premises MongoDB and configure continuous asynchronous replication to Amazon DocumentDB.

The best strategy is B:

- B: AWS Database Migration Service (AWS DMS) with CDC: This option is optimal as AWS DMS is specifically designed for efficient and secure database migrations to AWS. It supports MongoDB as a source and Amazon DocumentDB as a target. Using CDC allows for real-time data replication, ensuring that the migration is seamless and that the DocumentDB instance remains up-to-date throughout the migration process.

Reasons why the other strategies are less appropriate:

- A: EC2 MongoDB Community Edition with Synchronous Replication: While feasible, this approach is more complex and resource-intensive. It requires manual setup and maintenance of EC2 instances and MongoDB installations, which is not as streamlined as using a dedicated migration service like AWS DMS.
- C: AWS Data Pipeline for Data Transfer: AWS Data Pipeline is a viable option for data transfer but is not specifically tailored for database migrations. It might not handle the complexities and nuances of a MongoDB to DocumentDB migration as effectively as AWS DMS.
- D: AWS Glue Crawlers for Asynchronous Replication: While AWS Glue is a powerful ETL service, it is not the best fit for this specific database migration scenario. It is more suited for data extraction, transformation, and loading tasks, rather than real-time database replication and migration.

Question 25

A business is transitioning its application infrastructure to AWS and has several Amazon EC2 instances in its setup. The development team requires varying access levels, and the company aims to enforce a policy mandating that all Windows EC2 instances join an AWS-based Active Directory domain. Additionally, the company seeks to enhance security with measures like multi-factor authentication (MFA) and prefers using managed AWS services wherever feasible.

The potential solutions are:

A: Implement AWS Directory Service for Microsoft Active Directory. Utilize an Amazon Workspace for connecting and carrying out domain security configuration activities.

B: Set up AWS Directory Service for Microsoft Active Directory. Deploy an Amazon EC2 instance and use it for performing domain security configuration tasks.

C: Opt for AWS Directory Service Simple AD. Deploy an Amazon EC2 instance for executing domain security configuration operations.

D: Choose AWS Directory Service Simple AD. Use an Amazon Workspace for domain security configuration purposes.

The most suitable solution is B:

- B: AWS Directory Service for Microsoft Active Directory with EC2 Instance: This approach is ideal as AWS Directory Service for Microsoft Active Directory is a fully managed service that supports features like MFA and integrates seamlessly with Windows-based environments, including EC2 instances. Using an EC2 instance for configuration tasks allows for flexibility and control needed for complex domain security configurations, aligning well with the company's requirements for enhanced security and managed service utilization.

Reasons why the other options are less suitable:

- A and D: Amazon Workspaces for Configuration Tasks: While Amazon Workspaces is a managed desktop-as-a-service solution, it's not primarily designed for infrastructure configuration tasks. Workspaces are more suited for end-user computing rather than administrative domain configurations.
- C: AWS Directory Service Simple AD: Simple AD might not support all the features required for advanced domain security configurations, such as MFA. It's a more basic directory service compared to AWS Directory Service for Microsoft Active Directory and might not align with the company's need for enhanced security measures and integration with Windows environments.

Question 26

A business plans to transition its on-premises application to AWS and needs to separate its structured product data from transient user session data. Additionally, the company aims to establish replication in a different AWS Region to enable disaster recovery. The primary goal is to achieve optimal performance.

Here are the available options:

A: Set up an Amazon RDS DB instance with distinct schemas for product and session data. Implement a read replica of the RDS instance in a different AWS Region.

B: Use an Amazon RDS DB instance for storing product data with a read replica in another Region. Employ Amazon ElastiCache for Memcached as a global datastore for user session data.

C: Create two Amazon DynamoDB global tables: one for product data and another for user session data. Utilize DynamoDB Accelerator (DAX) for enhanced caching.

D: Deploy an Amazon RDS DB instance for product data with a read replica in another Region. Use an Amazon DynamoDB global table for managing user session data.

The most suitable solution is D:

- D: Amazon RDS for Product Data, DynamoDB Global Table for Session Data: This option effectively separates the structured product data and the transient user session data. Amazon RDS is well-suited for handling complex queries and transactions, which is typical for product data. The read replica in a different AWS Region addresses disaster recovery requirements. Amazon DynamoDB is highly scalable and ideal for managing transient user session data. Its global table feature ensures high availability and replication across regions, providing robust disaster recovery capabilities for the session data.

Why the other options are less suitable:

- A: Single RDS Instance for Both Data Types: Utilizing one RDS instance for both data types doesn't effectively decouple these datasets. This could lead to performance issues, especially with the fluctuating nature of session data.
- B: RDS for Product Data, ElastiCache for Session Data: While ElastiCache delivers high performance, it's important to note that ElastiCache with Memcached does not support Multi-AZ deployments or global datastores. This limitation makes it less suitable for disaster recovery and high availability across multiple regions. Additionally, ElastiCache is typically used for caching rather than as a primary datastore.
- C: DynamoDB Global Tables for Both Data Types: While DynamoDB is a highly capable NoSQL database, it may not be optimal for the structured product data that could benefit from the relational features and complex querying capabilities of RDS.

Question 27

A corporation manages a diverse AWS account infrastructure through AWS Control Tower, involving AWS Organizations, AWS Config, and AWS Trusted Advisor. It has set up a specific Organizational Unit (OU) for development accounts, used by a large team of developers for experimentation. The company aims to optimize costs within these development accounts by ensuring only burstable Amazon EC2 and Amazon RDS instances are used and by restricting access to non-essential services.

To achieve this, the following options are considered:

A. Implement a tailored SCP (Service Control Policy) in AWS Organizations. This policy should specifically permit only the use of burstable instances and prohibit non-essential services. Apply this SCP to the development-focused OU.

B. Develop a custom detective control (guardrail) within AWS Control Tower. Configure this control to allow only burstable instance deployment while excluding irrelevant services. This control should then be applied to the development OU.

C. Establish a custom preventive control (guardrail) in AWS Control Tower. This control should be set to enable only the deployment of burstable instances and exclude unrelated services. It should be applied to the development OU.

D. Set up an AWS Config rule within the AWS Control Tower account. This rule should allow only the deployment of burstable instances and block irrelevant services. Deploy this rule across the development OU using AWS CloudFormation StackSets.

The most appropriate solution is **A: Implement a tailored SCP in AWS Organizations.**

- **A: Custom SCP in AWS Organizations**: SCPs are effective for enforcing permissions across all accounts within an OU. They can explicitly allow or deny actions, making them ideal for restricting the deployment of specific types of instances and services. In this scenario, SCPs would directly control and restrict the actions developers can perform in their accounts, ensuring adherence to the cost optimization strategy.

The other options are less suitable for the following reasons:

- **B: Custom Detective Control in AWS Control Tower**: Detective controls are more about monitoring and alerting rather than preventing actions. They would identify non-compliance but wouldn't actively restrict the deployment of non-burstable instances or irrelevant services.
- **C: Custom Preventive Control in AWS Control Tower**: While preventive controls are designed to stop non-compliant actions before they occur, they are generally broader in scope. Implementing such controls specifically for burstable instances and service restrictions might not be straightforward and could lack the necessary granularity.
- **D: AWS Config Rule in AWS Control Tower Account**: AWS Config rules are great for configuration compliance checking, but they don't prevent actions from being taken. They would identify non-compliant resources after they are deployed, not prevent their deployment.

Question 28

A financial services company utilizes a complex, multi-tier application hosted on Amazon EC2 instances and AWS Lambda functions, with temporary data stored in Amazon S3. The S3 objects have a short lifespan of 45 minutes and are programmatically deleted after 24 hours. The company uses AWS CloudFormation stacks for deploying each application version. When a new version is successfully deployed and validated, they proceed to delete the

CloudFormation stack of the previous version. However, they encountered a problem where the CloudFormation stack deletion failed due to the inability to remove an S3 bucket.

To address this issue while maintaining the current architecture, the following solutions are proposed:

A. Develop a Lambda function that empties all contents from a specified S3 bucket. Integrate this Lambda function as a custom resource in the CloudFormation stack, ensuring it has a dependency on the S3 bucket.

B. Alter the CloudFormation template to switch from using Amazon S3 to Amazon Elastic File System (EFS) for temporary file storage. Adjust Lambda functions to operate within the same VPC as the EFS, and set up EFS mounts for both EC2 instances and Lambda functions.

C. Revise the CloudFormation stack to establish an S3 Lifecycle policy that automatically expires objects after 45 minutes. Add a dependency attribute to ensure the policy is linked to the S3 bucket.

D. Update the CloudFormation stack to include a 'DeletionPolicy' attribute with a 'Delete' value for the S3 bucket.

The most effective solution is **A: Implement a Lambda function to clear the S3 bucket, integrated as a custom resource in CloudFormation.**

- **A: Lambda Function as a Custom Resource**: This approach ensures that the S3 bucket is emptied before the stack deletion, overcoming the deletion failure issue. By integrating it as a custom resource with a dependency on the S3 bucket, CloudFormation can trigger this function to clear the bucket contents, enabling successful stack deletion.

The other options are less suitable for the following reasons:

- **B: Switch to Amazon EFS for Temporary Storage**: This change involves significant alterations to the application's architecture and does not directly address the S3 bucket deletion issue. Additionally, it introduces complexity and potential cost implications.
- **C: S3 Lifecycle Rule for Object Expiration**: While Lifecycle policies manage object expiration, they do not guarantee the deletion of all objects at the time of stack deletion. This could still lead to the same issue of stack deletion failure due to non-empty buckets.
- **D: S3 Bucket DeletionPolicy Attribute**: Adding a 'Delete' DeletionPolicy to the S3 bucket in CloudFormation could potentially solve the issue. However, if the bucket isn't empty, CloudFormation will still fail to delete the bucket. This solution doesn't guarantee the bucket will be empty at the time of stack deletion.

Question 29

A mobile gaming company is facing challenges with its current backend infrastructure, hosted on virtual machines in an on-premises data center. The backend is accessed via a REST API, which is crucial for the game's functionality, and player session data is centralized. The company experiences insufficient server capacity during peak hours, leading to latency issues. The management seeks a cloud-based solution to address these problems without altering the existing API model.

Here are the potential solutions:

A: Use a Network Load Balancer (NLB) for the REST API, with Amazon EC2 instances handling the business logic and Amazon Aurora Serverless for storing player session data.

B: Utilize an Application Load Balancer (ALB) for the REST API, with AWS Lambda for business logic execution and Amazon DynamoDB with on-demand capacity for player session data storage.

C: Implement the REST API through Amazon API Gateway, execute business logic with AWS Lambda, and use Amazon DynamoDB with on-demand capacity for player session data.

D: Deploy the REST API via AWS AppSync, run business logic on AWS Lambda, and use Amazon Aurora Serverless for player session data storage.

The most appropriate solution is **C: API Gateway, AWS Lambda, and DynamoDB with On-Demand Capacity.**

- **C: Amazon API Gateway, AWS Lambda, and DynamoDB:** This combination offers a highly scalable and serverless solution, ideal for handling varying loads. API Gateway efficiently manages the REST API, ensuring scalability and reliability. AWS Lambda allows for flexible and scalable execution of business logic. DynamoDB with on-demand capacity provides low-latency access to player session data and automatically adjusts to the fluctuating load.

Why the other options are less suitable:

- **A: NLB with EC2 and Aurora Serverless:** While this setup can handle varying loads, it involves managing EC2 instances, which adds operational complexity. Aurora Serverless is a good choice for variable workloads, but it might not provide the lowest latency for session data compared to DynamoDB.
- **B: ALB with Lambda and DynamoDB:** This approach is close to the ideal solution, but ALB is more suited for load balancing HTTP/HTTPS traffic and doesn't offer the same level of API management capabilities as API Gateway.
- **D: AppSync with Lambda and Aurora Serverless:** AWS AppSync is tailored for GraphQL APIs, not REST APIs. This makes it less suitable for the company's requirement to

maintain the current API model.

Question 30

A company is transitioning an application to AWS. This application currently operates in an on-premises data center, generating thousands of images nightly that are stored on an NFS file system. As part of the migration, the application will be hosted on an Amazon EC2 instance, utilizing Amazon Elastic File System (EFS) for storage. The company has established an AWS Direct Connect link for this purpose. Before finalizing the migration, it's necessary to devise a method for replicating the new images from the on-premises system to the EFS file system efficiently.

Potential solutions are:

A: Implement a routine process to execute `aws s3 sync` for transferring images from the on-premises file system to Amazon S3. Use an AWS Lambda function to respond to S3 event notifications and transfer images from S3 to EFS.

B: Install an AWS Storage Gateway file gateway with an NFS mount point. Integrate this mount point with the on-premises server, and establish a periodic process for copying images to this mount point.

C: Deploy an AWS DataSync agent on an on-premises server with NFS access. Utilize a public Virtual Interface (VIF) over Direct Connect to transfer data to an S3 bucket. Set up an AWS Lambda function to handle S3 event notifications and replicate images from S3 to EFS.

D: Install an AWS DataSync agent on an on-premises server with NFS access. Use a private VIF over Direct Connect to connect to an AWS PrivateLink interface VPC endpoint for Amazon EFS. Schedule a DataSync task to replicate images to EFS every 24 hours.

The best solution is **D: AWS DataSync with Private VIF to EFS**:

Why It's Correct: This method uses AWS DataSync, a service specifically designed for efficient data transfer between on-premises storage systems and AWS storage services. By deploying a DataSync agent directly on an on-premises server that has access to the NFS file system, the company ensures an optimized transfer of images. Utilizing a private Virtual Interface (VIF) over AWS Direct Connect enhances security and maintains a private network connection, which is crucial for sensitive data transfers. DataSync's ability to schedule tasks fits well with the company's requirement for daily image replication, making it an operationally efficient and direct solution for this use case.

Why the other options are less suitable:

A: `aws s3 sync`, Lambda, S3 to EFS

- Why It's Incorrect: This approach introduces an intermediary (Amazon S3) and relies on AWS Lambda for processing event notifications and transferring data from S3 to EFS. This two-step process (on-premises to S3, then S3 to EFS) adds unnecessary complexity and potential latency. It also requires managing Lambda functions and S3 event notifications, which could be operationally cumbersome for handling large volumes of images nightly.

B: AWS Storage Gateway File Gateway

- Why It's Incorrect: While AWS Storage Gateway's file gateway can facilitate on-premises to AWS storage transfers, it's not as direct or efficient as AWS DataSync for this scenario. The file gateway would involve setting up a storage volume that syncs to AWS, but it lacks the optimized data transfer capabilities of DataSync. Additionally, using the file gateway adds an additional layer of complexity compared to the more streamlined and automated process offered by DataSync.

C: DataSync, Public VIF, S3, Lambda

- Why It's Incorrect: Similar to option A, this method involves multiple steps and components: using DataSync to transfer data to an S3 bucket (over a less secure public VIF), and then relying on AWS Lambda to move data from S3 to EFS. This increases the complexity and potential points of failure. The use of a public VIF also raises security concerns compared to a private VIF, which provides a more secure and direct connection to AWS services.

Question 31

A company has shifted its web application from a local data center to AWS. The application's architecture on AWS includes Amazon CloudFront for content distribution, leading to an Application Load Balancer (ALB), which further directs traffic to an Amazon ECS setup. A recent security evaluation uncovered that the application can be accessed directly through both the CloudFront and ALB URLs, which is against the company's policy. The company wants to ensure that the web application is only reachable through the CloudFront URL.

Here are the possible solutions:

A: Assign a new security group to the CloudFront distribution and modify the ALB security group rules to permit access solely from this CloudFront security group.

B: Modify the ALB security group's ingress rules to allow access only from the CloudFront managed prefix list identified by `com.amazonaws.global.cloudfront.origin-facing`.

C: Set up a `com.amazonaws.region.elasticloadbalancing` VPC interface endpoint for Elastic Load Balancing and alter the ALB to be internal rather than internet-facing.

D: Retrieve CloudFront IP addresses from the AWS ip-ranges.json file and update the ALB security group's ingress rules to permit access only from these CloudFront IPs.

The most appropriate solution is B:

- B: Update ALB Security Group for CloudFront Managed Prefix List: This is the most straightforward and effective approach. AWS CloudFront provides a managed prefix list (`com.amazonaws.global.cloudfront.origin-facing`) that represents CloudFront's IP ranges. By updating the ALB security group to allow inbound traffic only from this prefix list, you ensure that the ALB is only accessible through CloudFront, aligning with the company's requirement. This solution is dynamically managed by AWS, meaning it automatically updates as CloudFront's IP ranges change, reducing the need for manual maintenance.

Reasons why the other options are less suitable:

- A: New Security Group for CloudFront: CloudFront does not support associating security groups with its distributions. This approach is not feasible in AWS.
- C: Internal ALB with VPC Interface Endpoint: Changing the ALB from internet-facing to internal and using a VPC interface endpoint doesn't address the requirement. It would make the ALB inaccessible from the internet, but it doesn't specifically restrict access to only CloudFront.
- D: Manual IP Updates from ip-ranges.json: Although this method can theoretically restrict ALB access to CloudFront IPs, it is operationally inefficient. The ip-ranges.json file must be manually monitored for changes to CloudFront IPs, and the ALB security group would need regular updates. This process is error-prone and does not scale well.

Question 32

A company operates a community forum hosted on AWS, utilizing an Application Load Balancer (ALB), Docker applications on Amazon ECS, and data storage on Amazon RDS for MySQL. The container image is stored in Amazon ECR. The company has established a disaster recovery (DR) policy requiring a Recovery Time Objective (RTO) of no more than 24 hours and a Recovery Point Objective (RPO) of no more than 8 hours. The company seeks the most cost-effective strategy to adhere to these DR requirements.

Proposed solutions:

A: Deploy duplicate ALB, EC2, ECS, and RDS in two regions using AWS CloudFormation. Execute RDS snapshots every 8 hours and utilize RDS multi-region replication for database

syncing. In a disaster, restore from the latest snapshot and redirect traffic to the secondary region's ALB using Route 53 DNS failover.

B: Store the Docker image in ECR across two regions. Implement an 8-hour snapshot schedule for RDS, copying snapshots to a secondary region. In a disaster, use CloudFormation to replicate infrastructure in the secondary region, restore from the latest snapshot, and redirect DNS to the secondary region's ALB.

C: Duplicate ALB, EC2, ECS, and RDS in a secondary region via CloudFormation. Schedule hourly RDS MySQL backups to S3 with cross-region replication. In a disaster, import Docker image to ECR in the secondary region, deploy on EC2, restore MySQL from the latest backup, and update DNS to the new ALB.

D: Set up a pilot light environment in a secondary region with an ALB and minimal EC2 resources for Docker in an Auto Scaling group. Create a cross-region RDS read replica. During a disaster, promote the replica to primary and update DNS to the secondary region's ALB.

The most suitable solution is B:

- B: ECR Storage in Two Regions, RDS Snapshots, CloudFormation Deployment: This approach aligns with the RTO and RPO goals, ensuring that the Docker image is readily available in two regions and that RDS snapshots are taken and stored every 8 hours. In case of a disaster, CloudFormation can rapidly deploy the necessary infrastructure in the secondary region using the latest RDS snapshot. This method offers a balanced approach between cost-effectiveness and operational readiness.

Reasons why the other options are less suitable:

- A: RDS Multi-Region Replication, Route 53 Failover: While this ensures immediate readiness, the continuous multi-region replication for RDS can be more expensive compared to periodic snapshots, thus not being the most cost-effective.
- C: Hourly MySQL Backups, Cross-Region S3 Replication: This option introduces complexity with hourly backups and cross-region S3 replication, increasing operational overhead without significant benefit over option B.
- D: Pilot Light Environment, RDS Read Replica: Maintaining a pilot light environment and a cross-region read replica of RDS might be more costly due to the continuous running of minimal resources and the replication process. It's more resource-intensive compared to the snapshot approach in option B.

Question 33

A company is transitioning its infrastructure to AWS while needing to adhere to various regulatory standards for distinct projects. The company requires a multi-account AWS environment that ensures a standardized baseline of management and security, yet is adaptable to different compliance needs within each AWS account. Additionally, the solution must integrate with the company's existing on-premises Active Directory Federation Services (AD FS) server, with an emphasis on minimizing operational complexity.

Proposed solutions:

A: Establish an AWS Organizations organization. Implement a universal SCP for restrictive access across all accounts. Create one OU for all accounts. Set up IAM for AD FS server federation. Establish a central logging account and enable AWS Config with conformance packs for all accounts.

B: Establish an AWS Organizations organization. Activate AWS Control Tower. Review existing controls (guardrails) for SCPs and adjust AWS Config as needed. Create OUs as required. Integrate AWS IAM Identity Center (AWS SSO) with the AD FS server.

C: Establish an AWS Organizations organization. Create SCPs for restrictive access and an OU structure for account grouping. Integrate AWS IAM Identity Center (AWS SSO) with the AD FS server. Set up a central logging account and enable AWS Config with aggregators and conformance packs in the central account.

D: Establish an AWS Organizations organization. Activate AWS Control Tower. Review existing controls (guardrails) for SCPs and adjust AWS Config as needed. Set up IAM for AD FS server federation.

The most suitable solution is B:

- B: AWS Control Tower, AWS IAM Identity Center (AWS SSO) Integration: This approach is the most efficient in terms of operational overhead. AWS Control Tower provides an automated way to set up and govern a secure, compliant multi-account AWS environment. It includes predefined SCPs and guardrails, simplifying the management of regulatory compliance requirements. The integration of AWS IAM Identity Center (AWS SSO) with the AD FS server streamlines user authentication and access management across the AWS environment.

Reasons why the other options are less suitable:

- A: Single SCP, Central Logging Account, AWS Config in Central Account: This setup lacks the streamlined governance and account management capabilities provided by AWS Control Tower. The single SCP approach might not be flexible enough for varying

compliance requirements, and manually managing AWS Config and logging across multiple accounts can be operationally complex.

- C: SCPs, OU Structure, Central Logging, AWS Config Aggregators: Similar to option A, this approach involves manual setup and lacks the automated governance features of AWS Control Tower. While it provides more granularity than a single SCP, it still requires more operational effort compared to using AWS Control Tower.
- D: AWS Control Tower, IAM Identity Provider for AD FS Federation: This option also leverages AWS Control Tower but doesn't mention the integration with AWS IAM Identity Center (AWS SSO), which is a crucial component for seamless federation with AD FS.

Question 34

An online magazine is preparing for the global launch of its latest edition. The magazine's website, which is dynamic, currently operates with an Application Load Balancer, a collection of Amazon EC2 instances for the web and application servers, and an Amazon Aurora MySQL database. The website features predominantly static content and is mostly accessed for reading. With the anticipated substantial increase in global internet traffic following the launch, the magazine prioritizes achieving optimal website performance, especially in the first week post-launch.

Proposed solutions to enhance global response times:

A: Replicate the Aurora MySQL database across regions logically. Replace web servers with Amazon S3 using cross-region replication.

B: Place web and application tiers in Auto Scaling groups. Implement AWS Direct Connect and deploy these tiers in various global regions.

C: Transition from Amazon Aurora to Amazon RDS for MySQL. Position all three application tiers – web, application, and database – in private subnets.

D: Implement an Aurora global database for physical cross-region replication. Use Amazon S3 with cross-region replication for static content. Deploy web and application tiers globally.

E: Deploy Amazon Route 53 with latency-based routing and Amazon CloudFront distributions. Place web and application tiers in Auto Scaling groups.

The most effective solutions are D and E:

- D: Aurora Global Database, Amazon S3 Cross-Region Replication: This solution leverages Aurora Global Database for efficient physical replication across regions, ensuring database availability and performance. Amazon S3 with cross-region replication

optimizes the delivery of static content. By deploying web and application tiers globally, the solution caters to a worldwide audience, enhancing response times.

- **E: Amazon Route 53, Amazon CloudFront, Auto Scaling Groups**: Route 53 with latency-based routing smartly routes users to the nearest endpoint, reducing latency. CloudFront distributions further improve content delivery speed for global users. Auto Scaling ensures that the web and application tiers can dynamically scale to handle traffic spikes.

Reasons why other options are less suitable:

- **A: Logical Replication, Replace Web Servers with S3**: Logical replication is not as efficient as physical replication for Aurora databases. Replacing web servers with S3 might not be feasible for a dynamic website and does not address global latency issues.
- **B: Auto Scaling, AWS Direct Connect, Global Deployment**: While Auto Scaling and global deployment are beneficial, Direct Connect is typically used for dedicated, private connectivity between on-premises and AWS, not for enhancing global website performance.
- **C: Migrate to Amazon RDS for MySQL, Private Subnets**: This does not inherently improve global performance. Private subnets for all tiers do not address the need for global content delivery and response time optimization.

Question 35

An online gaming company is seeking to reduce costs for its AWS-hosted workloads, including a production environment for its gaming application and an analytics application. The gaming application, hosted on Amazon EC2 instances, requires constant availability throughout the year. The analytics application, processing data from Amazon S3, can tolerate interruptions and can be paused and resumed as needed.

Cost-effective solutions for optimizing AWS workloads:

A: Secure an EC2 Instance Savings Plan for the gaming application instances and use On-Demand Instances for the analytics application.

B: Secure an EC2 Instance Savings Plan for the gaming application instances and use Spot Instances for the analytics application.

C: Employ Spot Instances for both the gaming and analytics applications, and set up a discount-provisioning catalog in AWS Service Catalog.

D: Utilize On-Demand Instances for the gaming application and Spot Instances for the analytics application, and set up a discount-provisioning catalog in AWS Service Catalog.

The optimal solution is **B:**

- **B: EC2 Instance Savings Plan for Gaming, Spot Instances for Analytics**: This solution is the most cost-effective. The EC2 Instance Savings Plan offers discounted rates for a committed use of specific instance types, ideal for the continuously running gaming application. Spot Instances, offering significant discounts compared to On-Demand pricing, are suitable for the analytics application, which can handle interruptions. This combination effectively balances cost savings with the operational needs of both applications.

Reasons why other options are less suitable:

- **A: Savings Plan for Gaming, On-Demand for Analytics**: While the Savings Plan is optimal for the gaming application, using On-Demand Instances for the interruptible analytics workload is not as cost-effective as Spot Instances.
- **C: Spot Instances for Both Applications, AWS Service Catalog**: Using Spot Instances for the gaming application, which requires constant availability, is risky due to the potential for instance interruptions. The AWS Service Catalog's role in providing discounts is unclear and doesn't align with standard AWS pricing models.
- **D: On-Demand for Gaming, Spot Instances for Analytics, AWS Service Catalog**: This option does not fully leverage cost savings for the gaming application. While Spot Instances are a good choice for the analytics application, the addition of AWS Service Catalog for discounts is unnecessary and does not align with established AWS pricing strategies.

Question 36

A company with a multitude of production AWS accounts is using AWS Organizations and has centralized its backup operations via AWS Backup. With increasing concerns about ransomware attacks, the company has adopted a new policy ensuring that backups remain secure even if privileged-user credentials in any production account are compromised.

To adhere to this policy, the following steps are proposed (choose three):

A: Implement cross-account backup using AWS Backup vaults in specific non-production accounts.

B: Apply a Service Control Policy (SCP) that limits modifications to AWS Backup vaults.

C: Utilize AWS Backup Vault Lock in compliance mode.

D: Apply the principle of least privilege to the IAM service role associated with AWS Backup.

E: Set backup frequency, lifecycle, and retention to guarantee the presence of at least one backup in the cold storage tier.

F: Configure AWS Backup to direct all backups to an Amazon S3 bucket in a specific non-production account, with S3 Object Lock enabled on the bucket.

The most effective combination is A, C, D:

- A: Cross-Account Backup with Non-Production Vaults: This approach isolates backups from production environments, reducing the risk of backups being compromised even if a production account is breached.
- C: AWS Backup Vault Lock in Compliance Mode: This feature ensures that backup policies and recovery points cannot be altered once locked, providing an additional layer of security against ransomware or malicious activities.
- D: Least Privilege for AWS Backup IAM Role: Restricting permissions to only what's necessary for the backup operation minimizes the risk of unauthorized access or manipulation of backup resources.

Reasons why the other options are less suitable:

- B: SCP Restricting Vault Modifications: While SCPs provide control over AWS service actions across accounts, they do not offer the same level of immutability and enforcement as the Vault Lock feature.
- E: Backup Configuration for Cold Storage: Ensuring that a backup always exists in cold storage is a good practice, but it doesn't directly address the requirement for resilience against credential breaches.
- F: Backups to a Non-Production S3 Bucket with Object Lock: While this could provide additional security, it involves more operational complexity compared to using AWS Backup's built-in features like Vault Lock. Additionally, AWS Backup does not natively support direct backups to an S3 bucket.

Question 37

A company is seeking an effective method to consolidate Amazon CloudWatch logs from multiple AWS accounts into a central account. The logs need to be kept within their original AWS Region. In the central account, the logs will be processed, standardized, and then sent to a security tool for further analysis. This solution needs to be scalable, particularly during peak business hours when log volume is high. An AWS Control Tower architecture is being considered to manage the multi-account logging structure.

The solutions architect proposes the following steps:

A: Establish an Amazon Kinesis data stream in the central logging account.

B: Set up an Amazon Simple Queue Service (Amazon SQS) queue in the central logging account.

C: Create an IAM role allowing CloudWatch Logs to add data to the Kinesis data stream. Include a trust policy in this role. In each AWS account, configure a subscription filter for each log group to forward data to the Kinesis data stream.

D: Create an IAM role permitting CloudWatch Logs to add data to the Amazon SQS queue. Include a trust policy in the role. In every account, set up a single subscription filter for all log groups to send data to the SQS queue.

E: Develop an AWS Lambda function in the central account to normalize the logs and forward them to the security tool.

F: Implement an AWS Lambda function in each member account to normalize logs and send them to the security tool.

The most suitable combination is A, C, E:

- A: Amazon Kinesis Data Stream in Central Account: Kinesis is ideal for handling large volumes of streaming data, like logs, and can scale according to the fluctuating load. This makes it an efficient choice for aggregating logs in the central account.
- C: IAM Role for CloudWatch Logs and Kinesis Stream: This step is crucial for enabling cross-account log data transfer to the Kinesis stream. Creating IAM roles with the necessary permissions and trust policies in each account ensures secure and authorized log data flow to the central Kinesis data stream.
- E: AWS Lambda for Normalization in Central Account: A Lambda function in the central account to normalize and process the logs is operationally efficient. It centralizes the processing workload and facilitates streamlined integration with the security tool.

Reasons why the other options are less suitable:

- B: Amazon SQS Queue in Central Account: While SQS is a robust messaging service, it's not as well-suited for log data streaming and real-time processing as Kinesis, especially for large-scale log data.
- D: IAM Role for CloudWatch Logs and SQS Queue: Similar to B, using SQS for log aggregation is less effective for streaming data. Additionally, setting a single subscription filter for all log groups in each account to an SQS queue might not be as efficient as using a Kinesis data stream.
- F: AWS Lambda for Normalization in Member Accounts: Implementing Lambda functions for log processing in each member account increases operational complexity and decentralizes the processing effort, which is less efficient compared to processing in the central account.

Question 38

A company is planning to move its legacy application, currently hosted in an on-premises data center, to AWS. This application is structured around an application server and a Microsoft SQL Server database server, each running on VMware VMs and utilizing 500 TB of data across various attached storage volumes. The company has already set up a 10 Gbps AWS Direct Connect link, connecting their on-premises data center to the nearest AWS Region, which is not yet utilized for other services. The primary goal is to transition the application to AWS while minimizing downtime.

The solutions architect suggests the following steps:

A: Migrate the database server VM to AWS using AWS Server Migration Service (AWS SMS) replication jobs.

B: Import the application server VM using VM Import/Export.

C: Export the VM images to an AWS Snowball Edge Storage Optimized device.

D: Migrate the application server VM to AWS using AWS Server Migration Service (AWS SMS) replication jobs.

E: Migrate the database to an Amazon RDS DB instance using AWS Database Migration Service (AWS DMS) replication instance.

The most appropriate combination is A and D:

- A: AWS SMS for Database Server VM Migration: AWS SMS provides an efficient method for migrating VMs to AWS. It supports incremental replication, which ensures that data is continuously synchronized without causing significant downtime. This approach is ideal for migrating the database server VM as it minimizes the downtime associated with large-scale data migrations.
- D: AWS SMS for Application Server VM Migration: Similarly, using AWS SMS for the application server ensures a seamless, incremental replication process. This method is effective for large-scale migrations and allows for ongoing synchronization until the final cutover, thus reducing downtime.

Reasons why the other options are less suitable:

- B: VM Import/Export for Application Server VM: While VM Import/Export is a viable option, it's less efficient for large-scale migrations compared to AWS SMS. VM Import/Export would require more downtime as it lacks the incremental replication feature of AWS SMS.
- C: AWS Snowball Edge for VM Image Export: Snowball Edge is typically used for transferring large amounts of data, especially when network constraints are an issue.

Given the existing 10 Gbps Direct Connect, Snowball Edge isn't necessary and would likely increase the complexity and duration of the migration.

- **E: AWS DMS for Database Migration**: While AWS DMS is a powerful tool for migrating databases, it's primarily used when transforming database schemas or changing database engines. In this scenario, where the SQL Server database is being migrated as-is, AWS SMS is more suited due to its VM-level replication capabilities.

Question 39

A company has a substantial on-premises server infrastructure and also runs a large number of Amazon EC2 instances across various AWS accounts, all managed under AWS Organizations. This setup involves hundreds of VPCs. The company has already established AWS Site-to-Site VPN connections to one of its AWS accounts and now aims to integrate its AWS accounts with its on-premises network. Additionally, the company seeks to effectively manage and control inter-VPC communication across different AWS accounts.

The solutions architect proposes the following steps:

A: Implement a transit gateway in one of the AWS accounts. Utilize AWS Resource Access Manager (AWS RAM) to share this transit gateway with other accounts within the organization.

B: Set up connections (attachments) to all VPCs and the existing VPNs.

C: Organize transit gateway route tables and associate these route tables with the respective VPCs and VPNs.

D: Establish VPC peering connections between VPCs.

E: Configure direct attachments between the VPCs and VPNs.

F: Arrange separate route tables for each of the VPCs and VPNs.

The optimal combination is **A, B, C:**

- **A: Transit Gateway with AWS RAM**: Setting up a transit gateway in one AWS account and sharing it across other accounts using AWS RAM offers a centralized, efficient way to manage inter-VPC and on-premises network connectivity. This approach reduces the complexity associated with creating multiple, individual connections.
- **B: VPC and VPN Attachments**: Establishing attachments for all VPCs and VPNs to the transit gateway simplifies the network topology and centralizes connectivity management. This step ensures all required connections are in place for effective network integration.

- C: Transit Gateway Route Tables: By configuring and associating route tables with the transit gateway, the company can control traffic flow between VPCs and the on-premises network. This setup allows for precise routing policies, enhancing both security and connectivity efficiency.

Reasons why the other options are less suitable:

- D: VPC Peering: While VPC peering is a method to connect VPCs, it's not scalable for hundreds of VPCs as it requires individual peering connections between each pair of VPCs. This approach would be operationally intensive and complex.
- E: Direct VPC and VPN Attachments: This option is redundant when using a transit gateway, which already handles attachments to VPCs and VPNs.
- F: Individual Route Tables for VPCs and VPNs: Setting up separate route tables for each VPC and VPN is more complex and doesn't provide the centralized control and simplicity of using transit gateway route tables.

Question 40

A company is running an application on AWS that extensively uses AWS Lambda functions and containers on Amazon Elastic Container Service (ECS) with AWS Fargate. The application primarily performs write operations and relies on an Amazon Aurora MySQL database. The application's traffic patterns are characterized by extended periods of inactivity followed by sudden and significant spikes in usage. The existing memory-optimized DB instance struggles to efficiently manage these fluctuating workloads.

The solutions architect is tasked with devising a strategy that scales effectively to accommodate the variable traffic while also being cost-effective. Here are the considered solutions:

A: Increase the number of read replicas in the Aurora database and invest in Instance Savings Plans and RDS Reserved Instances.

B: Transition to an Aurora DB cluster with multiple writer instances and invest in Instance Savings Plans.

C: Switch to an Aurora global database and purchase Compute Savings Plans and RDS Reserved Instances.

D: Migrate to Aurora Serverless v1 and invest in Compute Savings Plans.

The most suitable solution is D: Migrate to Aurora Serverless v1 and purchase Compute Savings Plans:

- D: Aurora Serverless v1: This option is ideal for handling variable workloads because it automatically adjusts the database's capacity based on actual usage, ensuring that the database scales seamlessly during traffic spikes and scales down during idle periods. This flexibility makes it highly cost-effective, especially for applications with uneven traffic patterns. Additionally, Aurora Serverless v1 supports write-heavy workloads.

Reasons why the other options are less suitable:

- A: Additional Read Replicas, Savings Plans, Reserved Instances: Adding more read replicas addresses read scalability but not write scalability. Moreover, investing in Reserved Instances may not be cost-effective for inconsistent workloads, as the company would pay for unused capacity during idle periods.
- B: Aurora DB Cluster with Multiple Writer Instances, Instance Savings Plans: While multiple writer instances might improve write scalability, managing capacity for variable workloads would still require manual intervention. Savings Plans might not yield the best cost savings for erratic traffic patterns.
- C: Aurora Global Database, Savings Plans, Reserved Instances: Aurora Global Database is designed for cross-region read replication and disaster recovery, not for handling variable workloads. The cost benefits of Savings Plans and Reserved Instances would be underutilized due to the application's irregular usage patterns.

Question 41

A company has moved its application to the AWS Cloud, where it's operated by two Amazon EC2 instances managed by an Application Load Balancer (ALB). The application's data storage relies on a MySQL database hosted on a separate EC2 instance, characterized by heavy read activity. Static content required by the application is stored on Amazon Elastic Block Store (EBS) volumes, which are attached to each EC2 instance and updated frequently. The application faces challenges in handling incoming requests during peak hours due to database read load constraints.

The company is exploring solutions to enhance the application's reliability, considering the fluctuating demand throughout the day. Here are the options:

A: Shift the application to AWS Lambda functions, configuring them as ALB targets. Utilize a single new EBS volume for static content, accessible by Lambda functions. Transition the database to an Amazon RDS for MySQL Multi-AZ DB cluster.

B: Transition the application to AWS Step Functions, with these as ALB targets. Employ Amazon Elastic File System (EFS) for static content, allowing state machines to access it. Move the database to Amazon Aurora MySQL Serverless v2, adding a reader DB instance.

C: Containerize the application and transfer it to an Amazon Elastic Container Service (ECS) cluster using AWS Fargate for task hosting. Create a new EBS volume for static content and attach it to the ECS cluster. Implement AWS Application Auto Scaling for the ECS cluster and set the ECS service as the ALB target. Migrate the database to an Amazon RDS for MySQL Multi-AZ DB cluster.

D: Containerize the application and deploy it on an Amazon Elastic Container Service (ECS) cluster, using AWS Fargate for hosting tasks. Establish an Amazon Elastic File System (EFS) for static content and integrate it with each container. Apply AWS Application Auto Scaling for the ECS cluster and designate the ECS service as the ALB target. Transition the database to Amazon Aurora MySQL Serverless v2 with a reader DB instance.

The most suitable solution is D:

- D: ECS Cluster with AWS Fargate, EFS for Static Content, Aurora Serverless v2 Database: This solution optimally addresses the application's need for scalability and handling heavy read loads. Containerizing the application allows for easier scaling and management, and using EFS for static content ensures all instances have updated and consistent access to static files. Aurora MySQL Serverless v2, with its ability to scale automatically, is ideal for handling the varying database read demands without manual intervention.

Reasons why the other options are less suitable:

- A: AWS Lambda Functions, EBS for Static Content, RDS MySQL Multi-AZ: While AWS Lambda can offer scalability, using a single EBS volume for static content isn't optimal as Lambda functions can't natively mount EBS volumes. Also, RDS MySQL, even with Multi-AZ, may not scale as efficiently as a serverless database for read-heavy workloads.
- B: AWS Step Functions, EFS, Aurora MySQL Serverless v2 with Reader: Step Functions are primarily designed for orchestrating workflows, not for serving as a backend for a web application. Though the database choice is suitable, the use of Step Functions for application hosting isn't aligned with the application's needs.
- C: ECS Cluster with AWS Fargate, EBS for Static Content, RDS MySQL Multi-AZ: Using an ECS cluster with Fargate is a good approach, but the use of a single EBS volume for static content is impractical as EBS volumes can't be shared across multiple instances or containers. The RDS MySQL Multi-AZ setup, while offering high availability, may not handle the fluctuating read loads as effectively as a serverless option.

Question 42

A solutions architect is working on securing a new Amazon API Gateway endpoint to ensure that only AWS users or roles with appropriate permissions can access it. Additionally, the

architect needs to monitor each request comprehensively, analyze the latency, and generate service maps.

Here are the proposed solutions:

A: Set up the API Gateway method's authorization to AWS_IAM. Assign execute-api:Invoke permission for the REST API resource to the necessary IAM user or role. Require API callers to sign requests with AWS Signature for accessing the endpoint. Use AWS X-Ray to trace and analyze the requests made to the API Gateway.

B: Enable CORS on the API Gateway resource and configure it to return only the company's domain in the Access-Control-Allow-Origin headers. Provide execute-api:Invoke permission for the REST API resource to the relevant IAM user or role. Employ Amazon CloudWatch for tracing and analyzing requests to the API Gateway.

C: Develop an AWS Lambda function to serve as a custom authorizer. Require API clients to send a key and secret in their calls, and validate these against the IAM system using Lambda. Utilize AWS X-Ray for request tracing and analysis on the API Gateway.

D: Generate a client certificate for API Gateway and distribute it to AWS users and roles who require access to the endpoint. Ensure API callers include this client certificate when accessing the endpoint. Use Amazon CloudWatch for request tracing and analysis on the API Gateway.

The optimal solution is A:

- A: API Gateway Authorization with AWS_IAM, AWS Signature, and AWS X-Ray: This approach effectively ensures that only authorized AWS users or roles can access the API Gateway endpoint. It leverages AWS IAM for authentication and AWS Signature for additional security in signing requests. AWS X-Ray is specifically designed for in-depth tracing and analysis of requests in AWS services, which fulfills the requirement for comprehensive monitoring and service map creation.

Why other options are less suitable:

- B: CORS with Domain Restriction, CloudWatch Monitoring: While CORS adds a layer of security for cross-origin requests, it isn't a comprehensive access control method. CloudWatch is helpful for monitoring but doesn't provide the same level of detailed tracing as AWS X-Ray.
- C: Lambda Custom Authorizer, Key/Secret Validation, AWS X-Ray: Implementing a Lambda custom authorizer for key/secret validation adds unnecessary complexity compared to using AWS IAM for straightforward authorization. AWS X-Ray is suitable for tracing, but the initial access control mechanism could be more streamlined.

- D: Client Certificate Distribution, CloudWatch Monitoring: Distributing client certificates is secure but can be operationally challenging for large-scale management. CloudWatch lacks the detailed request tracing capabilities offered by AWS X-Ray.

Question 43

A company uses AWS CodePipeline for continuous integration and deployment (CI/CD) of an application onto an Amazon EC2 Auto Scaling group. AWS CloudFormation templates define all their AWS resources. The application's artifacts are stored in an Amazon S3 bucket and deployed through instance user data scripts. However, as the application has grown in complexity, changes to the CloudFormation templates have unexpectedly led to downtime.

The following solutions have been proposed to enhance the CI/CD pipeline and minimize the risk of downtime due to template changes:

A: Modify the deployment scripts to identify and report any errors encountered during CloudFormation deployments. Develop a testing protocol to be executed by a dedicated team in a non-production environment prior to approving changes for the production environment.

B: Integrate automated testing using AWS CodeBuild within a test environment. Employ CloudFormation change sets to review changes before they are deployed. Utilize AWS CodeDeploy to implement blue/green deployment strategies, allowing for thorough evaluation and rollback if necessary.

C: Employ IDE plugins for template error checking and validate templates using the AWS CLI. Revise the deployment scripts to detect and alert on errors. Deploy changes to a test environment followed by manual testing before approving them for the production environment.

D: Switch to AWS CodeDeploy with a blue/green deployment approach, using CloudFormation in place of user data deployment scripts. Require manual testing by having operators log into running instances to confirm application functionality.

The most effective solution is B:

- B: Automated Testing with AWS CodeBuild, CloudFormation Change Sets, and AWS CodeDeploy for Blue/Green Deployment: This approach significantly enhances the CI/CD process. Automated testing in a separate test environment ensures that changes are vetted thoroughly before being deployed. CloudFormation change sets provide a clear view of how changes will impact the existing setup, allowing for better decision-making. Blue/green deployment via AWS CodeDeploy enables the smooth transition of the new version into production with minimal risk and easy rollback capabilities.

Why the other options are less appropriate:

- A: Manual Testing and Deployment Script Error Reporting: While detecting and reporting errors is valuable, relying on manual testing in a separate environment can be time-consuming and less efficient. It lacks the automated and proactive approach of solution B.
- C: IDE Plugin Validation, CLI Template Checking, and Manual Testing: Using IDE plugins and CLI validation improves the accuracy of the templates, but this process is more reactive than proactive. It still depends on manual testing, which isn't as reliable or scalable as automated testing.
- D: Manual Testing with Blue/Green Deployment Using CodeDeploy: Although employing blue/green deployments is a good practice, the reliance on manual testing for verifying application functionality is less efficient and more prone to human error compared to automated testing.

Question 44

A company based on the East Coast of North America is launching a new web application on Amazon EC2 in the us-east-1 Region. To ensure dynamic scalability and resilience, they need a disaster recovery setup with the us-west-1 Region in an active-passive configuration.

To achieve this after setting up a VPC in the us-east-1 Region, the solutions architect should consider the following actions:

A: Establish a VPC in the us-west-1 Region and connect it to the us-east-1 VPC using inter-Region VPC peering. Deploy an Application Load Balancer (ALB) across multiple Availability Zones in the us-east-1 VPC. Set up EC2 instances in both regions across multiple Availability Zones in an Auto Scaling group served by the ALB, spanning both VPCs.

B: Install an Application Load Balancer (ALB) across multiple Availability Zones in the us-east-1 VPC. Set up EC2 instances in multiple Availability Zones as part of an Auto Scaling group linked to the ALB. Replicate this setup in the us-west-1 Region. Use Amazon Route 53 with a failover routing policy and health checks to ensure high availability across both Regions.

C: Create a VPC in the us-west-1 Region and connect it to the us-east-1 VPC using inter-Region VPC peering. Deploy an ALB that spans both VPCs. Establish EC2 instances across multiple Availability Zones in each VPC as part of an Auto Scaling group, all served by the ALB. Set up an Amazon Route 53 record pointing to the ALB.

D: Install an ALB across multiple Availability Zones in the us-east-1 VPC. Set up EC2 instances in multiple Availability Zones as part of an Auto Scaling group linked to the ALB. Implement the same solution in the us-west-1 Region. Create separate Amazon Route 53

records for each Region pointing to their respective ALBs, using Route 53 health checks to ensure high availability across both Regions.

The most suitable solution is **B:**

- **B: ALB, Auto Scaling, and Route 53 Failover Routing in Each Region:** This setup efficiently meets the active-passive disaster recovery requirement. The ALB and Auto Scaling group in each Region ensure scalability and resilience. The failover routing policy with health checks in Route 53 allows automatic rerouting of traffic to the us-west-1 Region in case of failure in the primary Region, ensuring high availability.

Why other options are less appropriate:

- **A: ALB and Auto Scaling Spanning Both Regions:** This setup is operationally complex due to the spanning of resources across Regions. It also doesn't provide a clear active-passive disaster recovery setup as both Regions would be active.
- **C: Single ALB Spanning Both Regions:** ALBs do not span multiple Regions. This setup is not feasible in AWS as of now, making it an incorrect choice.
- **D: Separate Route 53 Records Without Failover:** This approach lacks a failover mechanism, making it unsuitable for an active-passive disaster recovery configuration. It does not automatically redirect traffic to the secondary Region in case of a failure in the primary Region.

Question 45

A company is transitioning from a legacy application with multiple .NET Framework components, which currently use a shared Microsoft SQL Server database and Microsoft Message Queueing (MSMQ) for asynchronous communication. As part of their move to AWS, they plan to update to containerized .NET Core components, requiring intricate orchestration. They need comprehensive control over their network and host configurations. The application relies heavily on a relational database model.

To align with these requirements, the following solutions are considered:

A: Deploy the .NET Core components on AWS App Runner and use Amazon RDS for SQL Server for the database. Implement Amazon EventBridge for asynchronous messaging.

B: Utilize Amazon Elastic Container Service (Amazon ECS) with AWS Fargate for the .NET Core components. Opt for Amazon DynamoDB for the database. Employ Amazon Simple Notification Service (Amazon SNS) for asynchronous messaging.

C: Run the .NET Core components on AWS Elastic Beanstalk and use Amazon Aurora PostgreSQL Serverless v2 for the database. Incorporate Amazon Managed Streaming for

Apache Kafka (Amazon MSK) for asynchronous messaging.

D: Host the .NET Core components on Amazon Elastic Container Service (Amazon ECS) with the Amazon EC2 launch type. Use Amazon Aurora MySQL Serverless v2 for the database. Adopt Amazon Simple Queue Service (Amazon SQS) for asynchronous messaging.

The most appropriate solution is D:

- D: ECS with EC2, Aurora MySQL Serverless v2, and SQS: This option provides the necessary control over networking and host configuration while accommodating the need for a relational database and asynchronous messaging. ECS on EC2 offers detailed control over container orchestration. Aurora MySQL Serverless v2 supports a relational database model similar to SQL Server, and SQS provides a managed, scalable solution for asynchronous messaging, analogous to MSMQ.

Why the other options are less suitable:

- A: App Runner, RDS for SQL Server, and EventBridge: While App Runner simplifies deployment, it offers less control over networking and host configurations. EventBridge may not align with the specific asynchronous messaging needs of the application.
- B: ECS with Fargate, DynamoDB, and SNS: Fargate reduces control over host configurations. DynamoDB, being a NoSQL database, may not suit the strongly relational data model requirements of the application. SNS is more suited for publish-subscribe messaging patterns.
- C: Elastic Beanstalk, Aurora PostgreSQL Serverless, and MSK: Elastic Beanstalk offers less granular control compared to ECS on EC2. Aurora PostgreSQL Serverless might introduce compatibility issues for an application designed for SQL Server. MSK, while powerful, might be an overkill for the application's messaging requirements and could add unnecessary complexity.

Question 46

A solutions architect is facing challenges while attempting to expand an Amazon EC2 placement group within a single Availability Zone due to an "insufficient capacity" error encountered during the addition of new instances. The placement group already contains multiple instances and needs to accommodate increased system load.

The solutions architect is considering the following actions to resolve the issue:

A: Switch to using a spread placement group and ensure a minimum of eight instances per Availability Zone.

B: Restart the existing instances in the placement group by stopping and starting them, then retry adding new instances.

C: Establish a new placement group and combine it with the existing one.

D: Deploy the additional instances on Dedicated Hosts within the placement groups.

The most effective solution is B:

- **B: Restart Existing Instances and Retry:** This approach can potentially resolve the "insufficient capacity" issue. Restarting the instances may relocate them to different underlying hardware with more available capacity, allowing new instances to be added to the placement group. This is a common method for handling capacity constraints within a specific Availability Zone or placement group.

Why the other options are less suitable:

- **A: Use a Spread Placement Group:** A spread placement group is designed to reduce the risk of simultaneous failures by distributing instances across distinct hardware. However, it doesn't necessarily resolve the capacity issue within a specific Availability Zone and limits the number of instances per hardware.
- **C: Create and Merge Placement Groups:** AWS does not support merging placement groups. Even if it were possible, this wouldn't address the underlying capacity issue in the Availability Zone.
- **D: Use Dedicated Hosts:** While Dedicated Hosts provide physical EC2 servers dedicated for use, this approach may not address the immediate capacity issue and could introduce additional costs and complexity without guaranteeing the availability of additional capacity in the placement group.

Question 47

A company is facing challenges with maintaining security updates on their Amazon EC2 instances due to an Auto Scaling group configuration that replaces instances upon reboot. This issue arose after a security update necessitated a reboot, causing the Auto Scaling group to launch new, unpatched instances. To prevent similar issues in the future, the company is exploring solutions that will allow them to effectively manage security updates without disrupting their infrastructure managed through infrastructure as code (IaC).

The company is considering the following actions:

A: Update the Auto Scaling group policy to prioritize replacing instances with the oldest launch configuration.

B: Set up a secondary Auto Scaling group and patch both groups during maintenance windows, followed by instance reboots.

C: Implement an Elastic Load Balancer (ELB) in front of the Auto Scaling group and monitor to ensure that health checks are passed after the Auto Scaling group replaces terminated instances.

D: Develop scripts for automated patching of an Amazon Machine Image (AMI), updating the launch configuration, and initiating an Auto Scaling instance refresh.

E: Place an Elastic Load Balancer in front of the Auto Scaling group and enable termination protection on the instances.

The most effective solution is C and D:

- C: Elastic Load Balancer with Health Checks: Configuring an ELB in front of the Auto Scaling group and ensuring that health checks are correctly passed after instance replacement is a good practice. This ensures that new instances are healthy and operational before they start receiving traffic, but it does not directly address the issue of maintaining patched instances.
- D: Automated Patching and Auto Scaling Refresh: This approach is the most comprehensive. Automating the patching of an AMI, updating the launch configuration with the new AMI, and then triggering an Auto Scaling instance refresh ensures that all new instances launched by the Auto Scaling group are up-to-date with security patches.

Why the other options are less suitable:

- A: Update Policy for Oldest Launch Configuration: This does not ensure that new instances are launched with the latest patches. It only affects the replacement strategy within the Auto Scaling group.
- B: New Auto Scaling Group for Patch Maintenance: Creating an additional Auto Scaling group for patching adds unnecessary complexity and doesn't ensure that new instances in the original group will be patched.
- E: Termination Protection on Instances: While termination protection can prevent instances from being terminated, it contradicts the Auto Scaling group's purpose of dynamically managing instance counts and doesn't address the issue of ensuring instances are patched.

Question 48

A team of data scientists working with Amazon SageMaker instances within a VPC (without internet access) requires a secure method to access Python packages from the Python Package Index (PyPI) repository for their machine learning models. The team uses datasets

stored in an Amazon S3 bucket, accessed via interface VPC endpoints for both S3 and SageMaker APIs. The challenge is to enable access to PyPI while maintaining the isolation of the SageMaker instances from the internet.

The following solutions are considered:

A: Establish an AWS CodeCommit repository for each required Python package, synchronize it with the PyPI repository, and create a VPC endpoint for CodeCommit.

B: Set up a NAT gateway in the VPC, adjust VPC routes for internet access, and use network ACLs to restrict access to the PyPI repository endpoint.

C: Implement a NAT instance in the VPC, modify VPC routes for internet access, and restrict SageMaker notebook instance firewall rules to only allow PyPI repository endpoint access.

D: Create an AWS CodeArtifact domain and repository, link it with public:pypi for external PyPI access, configure the Python client for CodeArtifact use, and establish a VPC endpoint for CodeArtifact.

The most suitable solution is D:

- D: AWS CodeArtifact for PyPI Access: This approach effectively maintains the isolation of SageMaker instances from the internet while providing access to PyPI packages. AWS CodeArtifact acts as a secure and managed artifact repository service, which can be connected to PyPI. By configuring the Python client to use CodeArtifact and setting up a VPC endpoint for CodeArtifact, the data scientists can safely access the necessary Python packages without exposing the SageMaker instances to the internet.

Reasons why the other options are less suitable:

- A: CodeCommit Repository Synchronization: While this provides a way to access specific packages, it involves manual effort to synchronize each required package and doesn't offer the comprehensive and automated access to PyPI that CodeArtifact provides.
- B: NAT Gateway with Network ACLs: This method exposes the VPC to the internet, which contradicts the requirement of keeping SageMaker instances isolated. Additionally, managing network ACLs for specific endpoints can be complex and error-prone.
- C: NAT Instance with Firewall Rules: Similar to the NAT gateway, this approach also compromises the isolation by providing internet access. Firewall rules on SageMaker instances add complexity and do not guarantee the same level of security and simplicity as using a managed service like CodeArtifact.

Question 49

A government agency, adhering to strict disaster recovery protocols, requires all Amazon Elastic Block Store (Amazon EBS) snapshots to be stored in at least two different AWS Regions beyond the primary region, while maintaining minimal operational overhead.

The following solutions are considered:

A: Employ Amazon Data Lifecycle Manager (Amazon DLM) to automatically copy EBS snapshots to additional regions on a daily basis.

B: Utilize Amazon EventBridge to schedule an AWS Lambda function that copies the EBS snapshots to other regions.

C: Leverage AWS Backup for snapshot creation and employ Amazon S3 Cross-Region Replication to copy these snapshots to additional regions.

D: Utilize Amazon EC2 Image Builder on a daily schedule to create an AMI and replicate it across multiple regions.

The most appropriate solution is A:

- **A: Amazon Data Lifecycle Manager (Amazon DLM):** This option efficiently meets the agency's requirements. Amazon DLM automates the process of creating, retaining, and copying EBS snapshots, which ensures compliance with the disaster recovery policy while minimizing operational effort. It allows for the setup of policies that can handle the snapshot creation and cross-region copying without manual intervention.

Reasons why the other options are less suitable:

- **B: EventBridge with Lambda:** While this can automate the process, it requires additional setup and maintenance of both EventBridge and Lambda. It introduces more complexity compared to the straightforward policy configuration in Amazon DLM.
- **C: AWS Backup with S3 Replication:** AWS Backup handles the snapshot creation, but EBS snapshots are not directly stored in S3, making S3 Cross-Region Replication inapplicable. This approach misunderstands how EBS snapshots and S3 work together.
- **D: EC2 Image Builder:** This solution is tailored more towards AMI management rather than EBS snapshot replication. It also adds unnecessary complexity, as it involves creating and managing AMIs instead of directly handling EBS snapshots.

Question 50

A company is facing a challenge with its project where Amazon EC2 instances being launched are larger than necessary. Due to policy constraints, this project's AWS account must operate independently, outside of the company's AWS Organizations structure. The company needs

to ensure that developers can only launch t3.small EC2 instances, and these instances should be restricted to the us-east-2 Region.

The following solutions are proposed:

A: Establish a new developer account, relocate all EC2 instances, users, and assets to the us-east-2 Region, integrate the account into the company's AWS Organizations, and implement a tagging policy to indicate Region preference.

B: Formulate a Service Control Policy (SCP) that prohibits the launching of EC2 instances except for t3.small in the us-east-2 Region and apply this SCP to the project's account.

C: Acquire and assign a t3.small EC2 Reserved Instance for each developer in the us-east-2 Region, with each instance tagged with the respective developer's name.

D: Craft an IAM policy that specifically permits the launching of only t3.small EC2 instances in the us-east-2 Region, and attach this policy to the roles and groups utilized by developers in the project's account.

The most appropriate solution is D:

- D: IAM Policy for EC2 Instance Type and Region: This approach directly addresses the requirement by creating an IAM policy that specifically allows only the launch of t3.small instances in the us-east-2 Region. This policy can be attached to the IAM roles or groups that developers use, effectively restricting their ability to launch other types of instances or to use other regions. It is a straightforward and effective method to enforce the desired restrictions without altering the account's organizational structure or purchasing reserved instances.

Reasons why the other options are less suitable:

- A: New Account and Tagging Policy: While moving to a new account and enforcing a tagging policy might provide some level of control, it doesn't directly restrict the instance type or Region. This option also involves significant operational overhead and doesn't align with the requirement to keep the project outside of the corporate IT's AWS Organizations.
- B: SCP in AWS Organizations: Since the project's account cannot be part of the company's AWS Organizations due to policy restrictions, using an SCP is not feasible in this scenario.
- C: Reserved Instances for Each Developer: Purchasing reserved instances is a financial commitment and doesn't restrict developers from launching other types or sizes of instances. It also doesn't enforce the regional restriction. This approach would also incur unnecessary costs and complexity.

Question 51

A scientific organization requires rapid processing of text and image data gathered from various radar stations for a critical deep space mission. This data, uploaded by the radar stations to an Amazon S3 bucket, is categorized by unique radar station identifiers. The company also maintains a separate S3 bucket in another AWS account for compliance reasons, where all data from the original bucket is replicated.

One specific radar station's data, known for its high accuracy, needs special attention. The replication of this data to the destination S3 bucket must be tracked to ensure it's completed within 30 minutes of its upload.

To fulfill these requirements, the solutions architect should consider the following options:

A: Implement AWS DataSync to replicate the prefixed data from the source S3 bucket to the destination. Use all available bandwidth for the task and monitor the task's status. Utilize Amazon EventBridge to trigger an alert if the task status changes.

B: In the second account, create a new S3 bucket exclusively for the highly accurate radar station's data. Establish a separate replication rule for this bucket to distinguish it from the other stations. Monitor the replication duration to the destination and use Amazon EventBridge to alert if the time limit is exceeded.

C: Activate Amazon S3 Transfer Acceleration on the source bucket and configure the highly accurate radar station to utilize the accelerated endpoint. Monitor the TotalRequestLatency metric of the destination bucket and establish an Amazon EventBridge rule to alert on status changes.

D: On the source S3 bucket, set up a specific replication rule that filters for keys matching the precise radar station's prefix. Turn on S3 Replication Time Control (RTC) to ensure timely replication. Monitor the maximum replication time and use Amazon EventBridge to alert if the 30-minute threshold is exceeded.

The most appropriate solution is D:

- D: Specific S3 Replication Rule with S3 RTC: This approach directly targets the key requirement of ensuring timely replication for the specified radar station's data. By creating a dedicated replication rule with S3 RTC, the data can be replicated within the desired timeframe, and monitoring through Amazon EventBridge provides an efficient way to alert if the replication time exceeds 30 minutes.

Reasons why the other options are less suitable:

- A: Using AWS DataSync: While DataSync can replicate data effectively, it does not specifically cater to the requirement of monitoring and ensuring replication within a 30-minute window for a specific radar station.
- B: Separate S3 Bucket for Specific Radar Station: Creating a new bucket for one radar station adds complexity and doesn't leverage S3's built-in replication features like S3 RTC, which is specifically designed for time-sensitive replication needs.
- C: S3 Transfer Acceleration: This option focuses on upload speed to S3 and does not address the requirement of monitoring the replication time to the destination bucket within a specific time frame.

Question 52

A company is facing the challenge of transferring its extensive on-premises data center to the AWS Cloud. This data center includes numerous Linux and Windows virtual servers, SAN storage, and a range of applications built using Java and PHP, which utilize MySQL and Oracle databases. The complexity of the data center is compounded by the lack of detailed and current technical documentation, making it difficult to fully understand the existing setup and estimate the costs associated with migrating to the cloud. The company requires a solutions architect to thoroughly assess the current infrastructure and accurately forecast cloud resource costs post-migration.

The solutions architect has the following options to consider:

A: Implement AWS Application Discovery Service
B: Utilize AWS SMS
C: Apply AWS X-Ray
D: Use AWS Cloud Adoption Readiness Tool (CART)
E: Employ Amazon Inspector
F: Leverage AWS Migration Hub

The correct combination of tools for this scenario is A, D, F.

- A: AWS Application Discovery Service: This service is designed to help enterprises understand their existing on-premises resources, particularly in environments where documentation is lacking or outdated. It gathers detailed information about servers, storage, and networking, which is essential for planning a migration.
- D: AWS Cloud Adoption Readiness Tool (CART): CART assesses an organization's readiness for cloud adoption. It helps identify areas that need more preparation before undertaking a large-scale migration, such as the one described.
- F: AWS Migration Hub: Migration Hub provides a central location for tracking the progress of migrations from on-premises to AWS. It can be used in conjunction with the

Application Discovery Service and other tools to track and manage the migration process.

Reasons for not choosing the other options:

- B: AWS SMS: While AWS SMS (Server Migration Service) is useful for migrating servers to AWS, it's more of a migration execution tool and doesn't provide the initial discovery and planning capabilities needed in this scenario.
- C: AWS X-Ray: This service is focused on application performance analysis and troubleshooting, not suitable for migration planning.
- E: Amazon Inspector: Inspector is used for security assessments of AWS resources, not for planning and executing migrations from on-premises environments.

Question 53

A solutions architect is tasked with evaluating the resilience of an application before its launch. The application operates on an Amazon EC2 instance situated in a private subnet of a VPC. The EC2 instance is managed by an Auto Scaling group set to maintain a single instance at all times. Data storage for the application is handled by an Amazon RDS for MySQL DB instance. The VPC is equipped with subnets across three Availability Zones but relies on a solitary NAT gateway.

The architect must develop a strategy that ensures the application's functionality across multiple Availability Zones. The aim is to enhance the application's resilience and reliability.

The potential solutions are:

A: Add extra NAT gateways in the remaining Availability Zones and adjust the route tables accordingly. Switch the RDS for MySQL DB instance to a Multi-AZ configuration. Adjust the Auto Scaling group to distribute instances across Availability Zones and set both the minimum and maximum capacity to 3.

B: Swap the NAT gateway for a virtual private gateway. Replace the RDS for MySQL DB instance with an Amazon Aurora MySQL DB cluster. Set the Auto Scaling group to deploy instances across all VPC subnets, with both minimum and maximum capacities set to 3.

C: Exchange the NAT gateway for a NAT instance. Transition from the RDS for MySQL DB instance to an RDS for PostgreSQL DB instance. Launch additional EC2 instances in the other Availability Zones.

D: Install additional NAT gateways in the remaining Availability Zones and revise the route tables. Alter the RDS for MySQL DB instance to enable automatic backups with a 7-day

retention period. Adjust the Auto Scaling group to deploy instances across all subnets in the VPC while maintaining the minimum and maximum capacities at 1.

The most suitable solution is A:

- A: Additional NAT Gateways, Multi-AZ RDS, Updated Auto Scaling Configuration: This approach enhances the application's resilience by ensuring high availability across multiple Availability Zones. Adding more NAT gateways prevents a single point of failure. Switching the RDS instance to a Multi-AZ configuration increases database resilience. Updating the Auto Scaling group to launch instances across Availability Zones with a higher capacity ensures that the application can handle increased load and remain operational if one zone experiences issues.

The reasons for not choosing the other options:

- B: Virtual Private Gateway, Aurora MySQL, Updated Auto Scaling: Replacing the NAT gateway with a virtual private gateway doesn't directly contribute to the application's resilience across Availability Zones. While Aurora MySQL offers high availability, it's not necessary for this specific scenario.
- C: NAT Instance, PostgreSQL DB, New EC2 Instances: Switching to a NAT instance and changing the database to PostgreSQL do not inherently improve cross-AZ resilience. Launching new EC2 instances without Auto Scaling adjustments does not provide the desired elasticity and resilience.
- D: Additional NAT Gateways, 7-day Backup Retention, Unchanged Auto Scaling: While adding more NAT gateways is beneficial, keeping the Auto Scaling group's capacity at 1 does not leverage the benefits of operating across multiple Availability Zones. The 7-day backup retention is a good practice but does not address the requirement for operational resilience across Availability Zones.

Question 54

A company is looking to move its on-site transaction-processing application, which currently operates within Docker containers on in-house VMs, to AWS. The application utilizes shared storage to log transaction data and needs to handle time-sensitive transactions. Due to fluctuating transaction volumes, the company requires a storage solution on AWS that is both low-latency and capable of scaling throughput as needed. The company wishes to avoid further application development or continued management of the Docker hosting environment.

Here are the potential solutions:

A: Shift the application containers to Amazon Elastic Kubernetes Service (Amazon EKS) and use Amazon S3 for storing the shared transaction data.

B: Transition the application containers to AWS Fargate under Amazon Elastic Container Service (Amazon ECS). Set up an Amazon Elastic File System (Amazon EFS) file system and integrate it into a Fargate task definition as a volume.

C: Move the application containers to AWS Fargate under Amazon Elastic Container Service (Amazon ECS). Establish an Amazon Elastic Block Store (Amazon EBS) volume and attach this volume to each running Fargate task.

D: Set up Amazon EC2 instances, install Docker, and migrate the application containers to these EC2 instances. Create an Amazon Elastic File System (Amazon EFS) file system and attach it to the EC2 instances.

The most suitable solution is B:

- B: AWS Fargate with ECS, Amazon EFS: This option best addresses the company's requirements. AWS Fargate eliminates the need for Docker container management and provides an elastic and serverless compute engine. Amazon EFS offers the necessary low-latency and scalable shared file storage, which can be easily integrated into the Fargate task definition. This combination provides the scalability and performance needed for the unpredictable volume of transactions.

Reasons why the other options are less appropriate:

- A: Amazon EKS, Amazon S3: While Amazon EKS would remove the burden of managing the Docker environment, using Amazon S3 for shared transaction data may not meet the low-latency requirements due to its object storage nature.
- C: AWS Fargate with ECS, Amazon EBS: Amazon EBS provides block storage, which is not suitable for shared storage needs in a containerized environment, especially with Fargate, where EBS volumes cannot be attached directly to tasks.
- D: Amazon EC2, Docker, Amazon EFS: This approach reintroduces the overhead of managing Docker on EC2 instances, which the company wants to avoid. While EFS provides the necessary shared file system, the management of EC2 instances and Docker contradicts the company's goal of reducing administrative tasks.

Question 55

A company is preparing to shift its various applications, hosted on a mix of physical and virtual Windows and Linux servers, to the AWS Cloud. A key challenge is the absence of a detailed inventory of their on-premises servers and applications. To effectively plan the migration and ensure proper resource allocation (right-sizing), the company needs comprehensive insights into network connections and interdependencies between applications. The solutions architect is tasked with evaluating the existing infrastructure to devise a strategic migration plan.

Here are the possible approaches:

A: Utilize Migration Evaluator for an AWS-led assessment of the current setup, and employ AWS Application Discovery Service Agentless Collector to feed data into a Migration Evaluator Quick Insights report.

B: Deploy AWS Migration Hub and install the AWS Application Discovery Agent on the servers. Use the Migration Hub Strategy Recommendations application data collector to compile a detailed report.

C: Implement AWS Migration Hub with the AWS Application Discovery Service Agentless Collector on the servers. Organize servers and databases using AWS Application Migration Service, and generate a report through Migration Hub Strategy Recommendations.

D: Utilize the AWS Migration Hub import tool to input details about the existing on-premises environment and create a report using Migration Hub Strategy Recommendations.

The most appropriate solution is B:

- **B: AWS Migration Hub with Application Discovery Agent, Strategy Recommendations:** This option provides a comprehensive solution for gathering detailed information about the on-premises environment. The AWS Application Discovery Agent, when installed on the servers, collects in-depth data regarding server utilization, network connections, and inter-application relationships. This data, analyzed through Migration Hub Strategy Recommendations, offers valuable insights for formulating an effective migration plan, including right-sizing recommendations.

Reasons why the other options are less suitable:

- **A: Migration Evaluator, Agentless Collector:** While Migration Evaluator offers valuable insights, it generally provides a higher-level evaluation compared to the detailed server-level analysis offered by AWS Migration Hub. The agentless collector might not capture as detailed data as the agent-based approach in option B.
- **C: AWS Migration Hub, Agentless Collector, AWS Application Migration Service:** The use of an agentless collector might not gather as comprehensive data as an agent-based solution. Additionally, grouping servers and databases through AWS Application Migration Service adds unnecessary complexity for the initial assessment phase.
- **D: AWS Migration Hub Import Tool:** Relying solely on an import tool might not capture the real-time and detailed operational data necessary for a thorough assessment, which is crucial for right-sizing and understanding application dependencies.

Question 56

A financial services company offering a SaaS platform for application compliance to global banks operates on AWS, utilizing multiple accounts managed through AWS Organizations. The platform extensively employs AWS resources worldwide. To comply with regulatory standards, the company must ensure all AWS API activities are audited, tracked for changes, and securely stored in a durable data repository.

Possible solutions to achieve these compliance requirements:

A: Establish a new AWS CloudTrail trail and use an already existing Amazon S3 bucket in the organization's management account for log storage. Activate the trail across all AWS Regions and implement MFA delete and encryption for the S3 bucket.

B: Create a new AWS CloudTrail trail in each member account of the organization. Set up new Amazon S3 buckets for logging in each account. Deploy the trail in every AWS Region, and enable MFA delete and encryption on all S3 buckets.

C: Implement a new AWS CloudTrail trail in the organization's management account. Designate a new Amazon S3 bucket with enabled versioning for log storage. Deploy this trail for all accounts within the organization, and ensure MFA delete and encryption are enabled on the S3 bucket.

D: Formulate a new AWS CloudTrail trail in the organization's management account. Allocate a new Amazon S3 bucket for storing logs. Set up Amazon Simple Notification Service (Amazon SNS) to alert an external management system about log-file deliveries. Activate MFA delete and encryption on the S3 bucket.

The most suitable option is C:

- **C: CloudTrail in Management Account, New S3 Bucket, Versioning, MFA Delete, Encryption:** This solution offers a centralized, efficient approach to audit and track API calls across the entire organization with minimal operational overhead. By establishing a single CloudTrail in the management account, the company ensures comprehensive coverage of all member accounts. Utilizing a dedicated S3 bucket with versioning enhances the security and durability of the logs, while MFA delete and encryption provide additional layers of data protection.

Why other options are less favorable:

- **A: CloudTrail, Existing S3 Bucket, MFA Delete, Encryption:** Although this option uses an existing S3 bucket, which might seem convenient, it lacks the enhanced security and organization benefits of creating a new, dedicated S3 bucket with versioning as in option C.

- **B: Individual CloudTrail in Each Account, New S3 Buckets, MFA Delete, Encryption:** Creating individual trails in each member account increases operational complexity and overhead, contrary to the requirement for minimal operational effort.
- **D: CloudTrail, New S3 Bucket, SNS Notifications, MFA Delete, Encryption:** While this option centralizes the trail creation in the management account, the use of Amazon SNS for log-file delivery notifications introduces unnecessary complexity and does not directly contribute to the primary requirement of auditing and secure log storage.

Question 57

A company is implementing a distributed in-memory database across a group of Amazon EC2 instances. This setup includes a primary node that manages cluster health, handles user requests, allocates tasks to worker nodes, and compiles responses for clients. Additionally, there are eight worker nodes that interact for data partition replication. The company's top priority is to minimize network latency to maximize performance.

Available solutions to optimize for low networking latency:

A: Utilize memory-optimized EC2 instances within a partition placement group.

B: Employ compute-optimized EC2 instances within a partition placement group.

C: Deploy memory-optimized EC2 instances in a cluster placement group.

D: Use compute-optimized EC2 instances within a spread placement group.

The most appropriate option is C:

- **C: Memory-Optimized EC2 Instances in a Cluster Placement Group:** This approach is ideal for the company's requirements. Memory-optimized instances are well-suited for in-memory databases due to their high memory capacity. A cluster placement group ensures that instances are physically located close to each other within the same Availability Zone, significantly reducing network latency and offering high-bandwidth networking, which is crucial for the performance of a distributed in-memory database.

Why the other options are less suitable:

- **A & B: Partition Placement Group (Memory or Compute Optimized Instances):** While partition placement groups are useful for spreading out instances to reduce the impact of hardware failures, they do not offer the same low-latency, high-bandwidth network performance as cluster placement groups. This makes them less optimal for a use case that prioritizes low network latency.
- **D: Compute-Optimized EC2 Instances in a Spread Placement Group:** Spread placement groups are designed to isolate instances from each other, which can be

counterproductive for a use case that requires high-speed, low-latency communication between nodes. Additionally, compute-optimized instances may not provide the necessary memory resources for an in-memory database.

Question 58

A company has around 10 TB of data in approximately 1 million .csv files stored on a virtual machine (VM) on-premises. This data is expanding by 1 TB every week. The company is looking to automate the daily backup of this data to the AWS Cloud, but only wants to back up specific subsets of data from certain source directories. They have an AWS Direct Connect set up for connectivity.

The solutions to consider for this scenario are:

A: Utilize the Amazon S3 CopyObject API operation with multipart upload for the initial data transfer to Amazon S3, followed by daily usage of the same API operation for new data.

B: Employ AWS Backup to create a backup plan for sending data to Amazon S3, scheduling daily backups.

C: Install the AWS DataSync agent on the on-premises VM. Set up a daily task with DataSync to synchronize the data with Amazon S3.

D: Use an AWS Snowball Edge device for the first backup, followed by AWS DataSync for incremental daily backups to Amazon S3.

The most suitable solution is C:

- C: AWS DataSync Agent and Daily DataSync Tasks: This option efficiently meets the requirements with minimal operational overhead. AWS DataSync can be installed directly on the on-premises VM, allowing for automated, scheduled daily backups. It also supports custom filters to backup only specified data, aligning with the company's need to back up subsets of data. DataSync efficiently handles large datasets and changes over a Direct Connect connection.

Reasons why the other options are less suitable:

- A: Amazon S3 CopyObject API Operation: While this method can be used for data transfer, it does not provide an automated, scheduled solution for daily backups. It would require manual intervention or additional scripting, increasing operational complexity.
- B: AWS Backup: AWS Backup is designed for AWS resources and doesn't directly support backing up on-premises file systems to S3. This makes it unsuitable for the company's current setup.

- D: AWS Snowball Edge and DataSync: While Snowball Edge is an efficient solution for the initial large data transfer, it's unnecessary for daily backups, especially with a Direct Connect already in place. DataSync alone is sufficient for both initial and incremental backups.

Question 59

A financial services company is developing an analytical solution on Amazon EMR to analyze survey data from its global customer base. The solution involves different roles: an Administrator responsible for provisioning EMR clusters, a Data Engineer for running ETL scripts, and a Data Analyst for executing SQL and Hive queries. The company needs a secure solution that provides least privilege access for these roles, allows only approved applications to be launched, and ensures all created resources are tagged appropriately.

The proposed solutions are:

A: Establish IAM roles for each persona, attaching identity-based policies for role-specific actions. Utilize AWS Config to monitor for noncompliant resources and notify the Administrator for remediation.

B: Implement Kerberos-based authentication for EMR clusters with cluster-specific Kerberos configurations.

C: Utilize AWS Service Catalog to manage available EMR versions, cluster configurations, and permissions for each user persona.

D: Use AWS CloudFormation to launch EMR clusters, attaching resource-based policies during creation. Employ AWS Config to monitor clusters and S3 buckets for compliance, alerting the Administrator for any noncompliance.

The most suitable solution is C:

- C: AWS Service Catalog for Controlled EMR Deployment: This approach effectively meets all the requirements. AWS Service Catalog allows the company to pre-define and approve specific EMR versions and configurations. It can enforce the use of tagging and restrict access based on user roles, ensuring that each persona only accesses the resources necessary for their job. This ensures a least privilege model, adherence to approved applications, and consistent resource tagging.

Reasons why the other options are less suitable:

- A: IAM Roles and AWS Config: While IAM roles provide necessary permissions, this option doesn't inherently control which EMR versions or configurations can be used.

AWS Config can monitor compliance but doesn't prevent the initial use of unauthorized resources or applications.

- **B: Kerberos Authentication for EMR:** Kerberos provides strong authentication but doesn't address the need to restrict users to approved EMR versions or configurations. It also doesn't ensure resource tagging.
- **D: CloudFormation and AWS Config:** Using CloudFormation ensures standardized deployments, but it doesn't restrict users to only approved EMR versions or configurations like AWS Service Catalog does. AWS Config monitors compliance but doesn't prevent unauthorized resource usage upfront.

Question 60

A SaaS company, hosting its service in the AWS eu-west-2 Region using AWS PrivateLink, has acquired a new customer in the us-east-1 Region. Their service runs on EC2 instances behind a Network Load Balancer (NLB) in private subnets across multiple Availability Zones in eu-west-2. The company has set up a new VPC in us-east-1 and established inter-Region VPC peering between the two Regions. The company aims to provide access to its SaaS service to the new customer without deploying new EC2 resources in us-east-1 immediately.

The proposed solutions are:

A: Set up a PrivateLink endpoint service in us-east-1 linked to the existing NLB in eu-west-2, allowing specific AWS accounts to connect.

B: Establish a new NLB in us-east-1 with an IP target group pointing to the eu-west-2 EC2 instances. Configure a PrivateLink endpoint service in us-east-1 using this NLB, granting access to certain AWS accounts.

C: Create an Application Load Balancer (ALB) in eu-west-2 in front of the EC2 instances. Set up an NLB in us-east-1 linked to a target group associated with the eu-west-2 ALB. Configure a PrivateLink endpoint service in us-east-1 using the us-east-1 NLB and permit specific accounts to connect.

D: Utilize AWS Resource Access Manager (AWS RAM) to share the eu-west-2 EC2 instances. In us-east-1, create an NLB and a target group including the shared eu-west-2 EC2 instances. Set up a PrivateLink endpoint service in us-east-1 using this NLB, enabling specific accounts to connect.

The most suitable solution is **B:**

- **B: NLB in us-east-1 with IP Target Group to eu-west-2 EC2 Instances:** This approach effectively extends the reach of the SaaS service to the us-east-1 Region without deploying new EC2 resources. The NLB in us-east-1, pointing to the IP addresses of the

eu-west-2 instances, bridges the inter-Region gap. The PrivateLink endpoint service in us-east-1 using this NLB allows controlled access to the service, fulfilling the requirement.

Reasons why the other options are less suitable:

- **A: PrivateLink Endpoint Service in us-east-1 with eu-west-2 NLB:** AWS PrivateLink does not support cross-Region NLBs, making this option unfeasible.
- **C: Inter-Region ALB and NLB Setup:** This solution adds unnecessary complexity and isn't supported by AWS as ALBs can't be target groups for NLBs, nor does PrivateLink support cross-Region configurations.
- **D: Using AWS RAM and Cross-Region NLB:** AWS RAM does not facilitate sharing EC2 instances across Regions. Moreover, this setup is overly complex and doesn't align with AWS service capabilities.

Question 61

A company's Chief Information Security Officer (CISO) is focusing on enhancing their current Continuous Integration/Continuous Deployment (CI/CD) processes. The goal is to enable rapid and efficient patching of their web application, which is crucial for addressing vulnerabilities with minimal downtime. The application is hosted on a collection of Amazon EC2 instances behind an Application Load Balancer. The company uses GitHub for source code management and AWS CodeBuild for building the application. They plan to implement AWS CodePipeline to initiate builds upon GitHub commits, utilizing the existing CodeBuild project. The key requirement is to swiftly deploy patches while maintaining the ability to revert changes if necessary.

The potential CI/CD configurations are:

A: Implement CodePipeline with a deployment stage utilizing AWS CodeDeploy for in-place deployments. Continuously monitor the deployed code and, in case of any issues, release another code update.

B: Set up CodePipeline with a deployment stage using AWS CodeDeploy configured for blue/green deployments. Continuously monitor the deployed code and, if issues arise, manually roll back the deployment using CodeDeploy.

C: Configure CodePipeline with a deployment stage using AWS CloudFormation to manage separate pipelines for testing and production environments. Monitor the new code deployment and, if problems are detected, issue another code update.

D: Implement CodePipeline with a deployment stage using AWS OpsWorks for in-place deployments. Continuously monitor the deployed code and, in case of any issues, release

another code update.

The most suitable solution is **B:**

- **B: CodePipeline with CodeDeploy for Blue/Green Deployments:** This configuration aligns with the requirements for rapid patch deployment and minimal downtime. Blue/green deployments allow for a new version of the application to be deployed alongside the old version. This setup enables quick rollback to the previous version if any issues are encountered, ensuring minimal disruption to the service.

Reasons why the other options are less suitable:

- **A: CodePipeline with CodeDeploy for In-Place Deployments:** While this allows for continuous deployment, in-place updates lack the ability to quickly roll back changes without pushing another code update, which can be time-consuming and risky in case of critical vulnerabilities.
- **C: CodePipeline with CloudFormation for Test/Production Stacks:** This setup is more complex and doesn't provide the immediate rollback capability that blue/green deployments offer. It also requires pushing new code updates for rollback, which can be slower in addressing critical issues.
- **D: CodePipeline with OpsWorks for In-Place Deployments:** Similar to option A, in-place deployments through OpsWorks do not offer the immediate rollback capabilities of blue/green deployments. This can result in longer downtimes during rollback procedures.

Question 62

A company is managing an increasing number of Amazon S3 buckets across two different AWS Regions. One of their key requirements is to efficiently monitor and report on the encryption status of the objects stored in these S3 buckets. The company's internal compliance teams need access to a dashboard that displays this information for monitoring and audit purposes.

The potential solutions for achieving this are:

A: Set up individual S3 Storage Lens dashboards in each Region to monitor bucket and encryption metrics. Then, integrate and consolidate the data from these regional dashboards into a unified dashboard in Amazon QuickSight for the compliance teams.

B: Implement AWS Lambda functions in each Region. These functions will enumerate the number of buckets and check the encryption status of the objects within them. The gathered data will be stored in Amazon S3, and Amazon Athena queries will be used to create a custom dashboard in Amazon QuickSight for the compliance teams.

C: Utilize the default dashboard provided by S3 Storage Lens to track both bucket and encryption metrics. Directly grant access to this dashboard within the S3 console to the compliance teams.

D: Configure Amazon EventBridge to trigger on AWS CloudTrail events related to S3 object creation. Set up this rule to invoke an AWS Lambda function that will record encryption metrics in Amazon DynamoDB. Then, use Amazon QuickSight to create a dashboard displaying these metrics for the compliance teams.

The most suitable solution is C:

- C: Use S3 Storage Lens Default Dashboard: This option offers the least operational overhead. S3 Storage Lens provides built-in capabilities to track and report metrics on S3 usage and activity, including encryption status. By using the default dashboard, the company can quickly and efficiently provide the necessary data to their compliance teams without the need for additional setup or integration.

Reasons why the other options are less suitable:

- A: Separate Storage Lens Dashboards and QuickSight Integration: While this method provides detailed insights, it involves additional steps of creating separate dashboards for each region and then aggregating this data in QuickSight, increasing the operational complexity.
- B: AWS Lambda Functions and Athena Queries: This solution is more complex and requires custom development. It involves managing Lambda functions, storing data in S3, and then querying this data via Athena to present in QuickSight, which is operationally more demanding.
- D: EventBridge, Lambda Function, and QuickSight: This approach is also complex, requiring the setup of EventBridge rules, development of Lambda functions to process data, storing this data in DynamoDB, and then visualizing it in QuickSight. It's more time-consuming and operationally intensive compared to using the default S3 Storage Lens dashboard.

Question 63

A company oversees numerous AWS accounts under AWS Organizations, with different business divisions operating EC2 instances. Each instance must be tagged with a BusinessUnit tag for cost-tracking purposes. However, a recent check showed some instances lacked this tag, and the company had to manually add it.

To ensure future compliance with the tagging policy, a solutions architect should consider the following strategies:

A: Implement tag policies at the organizational level, create a tag policy for the BusinessUnit tag, check for proper capitalization, and apply this policy to the EC2 resource type. Attach this tag policy to the organization's root.

B: Similar to option A, but instead attach the tag policy to the management account of the organization.

C: Develop an SCP and associate it with the organization's root, using the following policy statement to deny the creation of EC2 instances without the appropriate tag:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyEC2Creation",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "Null": {
          "aws:RequestTag/BusinessUnit": "true"
        }
      }
    }
  ]
}
```

D: Create an SCP and attach the SCP to the organization’s management account. Include the following statement in the SCP:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "DenyEC2Creation",
      "Effect": "Deny",
      "Action": "ec2:RunInstances",
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": {
        "Null": {
          "aws:RequestTag/BusinessUnit": "false"
        }
      }
    }
```

```
    ]
}
```

The correct strategy is C:

This option ensures that EC2 instances cannot be launched without the required BusinessUnit tag. By applying this SCP at the root level, it's enforced across all AWS accounts within the organization, thus providing a centralized approach to compliance and preventing instances without the correct tagging from being created. The 'Deny' effect coupled with the 'Null' condition on the 'aws:RequestTag/BusinessUnit' enforces that the tag must be present upon EC2 instance creation.

Why other options are incorrect:

- A: While tag policies help enforce tagging compliance, they do not prevent the creation of resources without tags. They are also not as immediately enforceable as SCPs.
- B: Attaching the policy to the management account instead of the root does not guarantee enforcement across all accounts within the organization.
- D: The condition `"aws:RequestTag/BusinessUnit": "false"` would not enforce the presence of the tag. It would deny the creation only if the tag is explicitly set to 'false', which does not ensure that the tag is present.

Question 64

A company is currently operating a significant workload involving numerous Amazon EC2 instances within a Virtual Private Cloud (VPC) that encompasses a variety of public and private subnets. In this setup, the public subnets are configured with a route to an internet gateway using 0.0.0.0/0, while the private subnets have a similar route set up but to a NAT gateway. The task at hand for a solutions architect is to transition this entire array of EC2 instances to IPv6. It's crucial that the EC2 instances located in the private subnets remain inaccessible from the public internet.

The solutions proposed to achieve this migration are:

A: Amend the current VPC by associating it and all its subnets with a custom IPv6 CIDR block. Following this, revise every route table in the VPC to include a ::/0 route pointing to the internet gateway.

B: Modify the existing VPC by linking it and all subnets with an IPv6 CIDR block provided by Amazon. Then, update the route tables for all the private subnets to introduce a ::/0 route that directs to the NAT gateway.

C: Update the existing VPC, associating it and all its subnets with an IPv6 CIDR block supplied by Amazon. Establish an egress-only internet gateway. Subsequently, adjust the route tables for all private subnets to incorporate a ::/0 route that leads to this egress-only internet gateway.

D: Alter the current VPC by attaching a custom IPv6 CIDR block to it and all its subnets. Then, create a new NAT gateway with IPv6 support and update all private subnet route tables to add a ::/0 route that connects to this IPv6-compatible NAT gateway.

The most appropriate solution is **C: Update VPC with Amazon-Provided IPv6 CIDR and Egress-Only Internet Gateway.**

- **C: Update VPC with Amazon-Provided IPv6 CIDR and Egress-Only Internet Gateway:** This approach is the most suitable as it ensures that EC2 instances in private subnets are not exposed to the public internet while enabling them to use IPv6. The key here is the use of an egress-only internet gateway, which allows outbound IPv6 traffic and blocks inbound traffic, thus maintaining the necessary privacy and security.

The reasons why the other options are less favorable:

- **A: Custom IPv6 CIDR Block and Internet Gateway Route:** This method exposes private subnet instances to the public internet by routing IPv6 traffic through the internet gateway, which contradicts the requirement of keeping private instances inaccessible from the public internet.
- **B: Amazon-Provided IPv6 CIDR and NAT Gateway Route:** The use of a NAT gateway for IPv6 traffic is not appropriate as NAT gateways do not support IPv6. Therefore, this option fails to provide a viable solution for the IPv6 migration.
- **D: Custom IPv6 CIDR Block and IPv6-Compatible NAT Gateway:** Although this option considers IPv6 support, the use of a custom IPv6 CIDR block is unnecessary since Amazon provides these blocks. Moreover, like option B, the use of a NAT gateway for IPv6 traffic is not feasible as it does not support IPv6, making this option impractical for the desired migration.

Question 65

A company is utilizing Amazon API Gateway for the deployment of a private REST API, which is designed to handle sensitive data. This API is intended to be exclusively reachable from an application running within a Virtual Private Cloud (VPC). After deploying the API, the company faces an issue where the API is not accessible from an Amazon EC2 instance situated within the same VPC.

To resolve the connectivity issue between the EC2 instance and the API, the company is considering the following solutions:

A: Set up an interface VPC endpoint for the API Gateway and attach a policy allowing all actions (apigateway:*). Turn off private DNS naming for this VPC endpoint. Implement a resource policy for the API that permits access from the VPC and use the VPC endpoint's DNS name for API access.

B: Establish an interface VPC endpoint specific to the API Gateway. Attach a policy that specifically permits the execute-api:Invoke action. Activate private DNS naming for the VPC endpoint. Apply a resource policy to the API that grants access from this VPC endpoint and access the API using its DNS names.

C: Implement a Network Load Balancer (NLB) along with a VPC link. Set up a private connection between the API Gateway and the NLB, accessing the API via the API endpoint's DNS names.

D: Deploy an Application Load Balancer (ALB) and create a VPC Link. Configure a private integration between the API Gateway and the ALB, using the ALB endpoint's DNS name to access the API.

The most suitable solution is **B: Establish Interface VPC Endpoint with Private DNS and Specific Policy.**

- **B: Establish Interface VPC Endpoint with Private DNS and Specific Policy:** This is the correct approach because it directly addresses the requirement of allowing secure, private access to the API from within the VPC. By creating an interface VPC endpoint, it enables the EC2 instance in the VPC to privately connect to the API Gateway. Enabling private DNS naming for the VPC endpoint facilitates straightforward access using the API's DNS names. The endpoint policy focused on the execute-api:Invoke action and a resource policy that allows access from this VPC endpoint ensure that only the necessary permissions are granted, aligning with security best practices.

Reasons why the other options are not suitable:

- **A: Interface VPC Endpoint with All Actions Policy:** This option inaccurately suggests disabling private DNS naming, which complicates accessing the API. Additionally, allowing all actions (apigateway:*) is generally not recommended due to broader security implications.
- **C: NLB and VPC Link for Private Integration:** While this setup can be used for certain scenarios, it is overly complex for the given requirement. It does not directly address the issue of the EC2 instance accessing the API within the same VPC.
- **D: ALB and VPC Link for Private Integration:** Similar to option C, this approach introduces unnecessary complexity. It involves additional components like ALB and VPC

Link, which are not required for the straightforward requirement of enabling EC2 instance access to the API within the same VPC.

Question 66

Following a merger between a prominent payroll firm and a smaller staffing agency, the combined entity now possesses several business units, each operating under its own AWS account. A solutions architect has been tasked with centralizing the management of billing and access policies for all these AWS accounts. To initiate this process, AWS Organizations has been configured, and invitations have been sent from a central management account to all member accounts within the company.

The next steps to be taken by the solutions architect to fulfill these requirements include:

A: In every member account, establish an IAM group named 'OrganizationAccountAccess', incorporating the required IAM roles for each administrator.

B: In each member account, create an IAM policy titled 'OrganizationAccountAccessPolicy'. Link these member accounts to the management account through cross-account access.

C: Formulate an IAM role named 'OrganizationAccountAccessRole' in each member account. Allow the management account permissions to assume this IAM role.

D: In the management account, create an IAM role called 'OrganizationAccountAccessRole'. Attach the 'AdministratorAccess' AWS managed policy to this IAM role and allocate this role to administrators in each member account.

The most appropriate action is **C: Formulate 'OrganizationAccountAccessRole' IAM Role in Each Member Account.**

- **C: Formulate 'OrganizationAccountAccessRole' IAM Role in Each Member Account:** This solution is ideal as it aligns with AWS best practices for managing multiple accounts within AWS Organizations. By creating the 'OrganizationAccountAccessRole' in each member account and granting the management account the ability to assume this role, centralized management of access and billing policies is facilitated. This role provides a secure way for the management account to access member accounts without needing individual user credentials.

The reasons why the other options are not suitable:

- **A: Creating IAM Group in Member Accounts:** This approach does not effectively centralize management. Creating IAM groups within each member account does not address the need for centralized control from the management account.

- B: Creating IAM Policy in Member Accounts: While creating a specific IAM policy in each member account is a part of access management, it does not by itself enable centralized control. This solution lacks the mechanism for the management account to assume roles within member accounts.
- D: Creating IAM Role in Management Account: This option misplaces the creation of the 'OrganizationAccountAccessRole'. The role needs to be created in each member account, not in the management account. Assigning this role to administrators in each member account does not facilitate centralized management from the management account.

Question 67

A company has modernized its application services by containerizing them and deploying them across several Amazon EC2 instances, each with its own public IP address. These services utilize an Apache Kafka cluster set up on the EC2 instances and a PostgreSQL database that has been transitioned to Amazon RDS for PostgreSQL. With the upcoming release of a new version of their flagship product, the company anticipates a substantial surge in orders on their platform. To prepare for this, the company is considering architectural changes that would minimize operational complexities and effectively support the product launch.

The proposed architectural modifications are:

A: Implement an EC2 Auto Scaling group and pair it with an Application Load Balancer. Enhance the database instance by adding more read replicas. Transition to Amazon Kinesis data streams for application services and directly handle static content through Amazon S3.

B: Set up an EC2 Auto Scaling group behind an Application Load Balancer. Switch the database instance to Multi-AZ mode with enabled storage auto-scaling. Integrate Amazon Kinesis data streams into the application services and directly serve static content from Amazon S3.

C: Migrate the application to a Kubernetes cluster established on the EC2 instances, managed through an Application Load Balancer. Configure the database instance for Multi-AZ deployment and enable storage auto-scaling. Switch to Amazon Managed Streaming for Apache Kafka for application services and utilize Amazon S3 for static content, delivering it via an Amazon CloudFront distribution.

D: Transition the application to Amazon Elastic Kubernetes Service (Amazon EKS) with AWS Fargate, enabling auto-scaling and pairing it with an Application Load Balancer. Add more read replicas to the database instance. Adopt Amazon Managed Streaming for Apache Kafka

for application services and store static content in Amazon S3, distributed through Amazon CloudFront.

The most effective solution is **D: Utilize Amazon EKS with AWS Fargate, Enable Auto Scaling, and Integrate Amazon Managed Streaming for Apache Kafka.**

- **D: Utilize Amazon EKS with AWS Fargate, Enable Auto Scaling, and Integrate Amazon Managed Streaming for Apache Kafka:** This option represents the best blend of scalability, operational efficiency, and performance optimization. Amazon EKS with AWS Fargate simplifies Kubernetes operations and eliminates the need to manage server infrastructure, reducing operational overhead. The use of auto-scaling ensures that the application can handle the increased load effectively. Amazon Managed Streaming for Apache Kafka provides a managed Kafka service that is easier to operate and scale than a self-managed Kafka cluster on EC2 instances. Storing static content in Amazon S3 and delivering it through Amazon CloudFront optimizes content delivery and further reduces load on the primary infrastructure.

The reasons why the other options are less suitable:

- **A: EC2 Auto Scaling with Additional Read Replicas and Amazon Kinesis:** While this provides scalability, it doesn't address the complexity of managing a Kafka cluster on EC2 instances. Additionally, the absence of a content delivery network (CDN) like CloudFront may not optimally serve static content.
- **B: Multi-AZ DB Instance and Kinesis Data Streams:** This option improves database availability but still retains the complexity of managing the Kafka cluster. It also lacks the use of a CDN for static content delivery.
- **C: Kubernetes on EC2 and Managed Kafka:** Although this introduces Kubernetes for container management and a managed Kafka service, the manual management of Kubernetes on EC2 instances adds operational complexity compared to using Amazon EKS with AWS Fargate. Moreover, it lacks the auto-scaling capabilities provided by AWS Fargate.

Question 68

A company is facing challenges with its existing VPN setup in an on-premises data center, which employees use to access files in their Windows home directories. With a substantial increase in remote work, the bandwidth demands during business hours are maxing out, impacting the company's operational efficiency. The goal is to develop an AWS-based solution that can accommodate the growing remote workforce, alleviate bandwidth constraints at the data center, and minimize operational complexities.

To achieve these objectives, the company is considering various steps:

A: Implement an AWS Storage Gateway Volume Gateway and link a volume from this Volume Gateway to the on-premises file server.

B: Transition the home directories to Amazon FSx for Windows File Server.

C: Shift the home directories to Amazon FSx for Lustre.

D: Move remote users over to AWS Client VPN.

E: Establish an AWS Direct Connect link between the on-premises data center and AWS.

The optimal combination of steps that fulfill these requirements with the least operational overhead are **B: Migrate to Amazon FSx for Windows File Server** and **D: Migrate to AWS Client VPN.**

- **B: Migrate to Amazon FSx for Windows File Server:** This step is highly effective because Amazon FSx for Windows File Server is specifically designed to support Windows-based file storage needs. Migrating home directories to FSx would provide remote workers direct access to their files on AWS, significantly reducing the bandwidth usage on the company's on-premises VPN. Additionally, FSx is fully managed, which helps in reducing operational overhead.
- **D: Migrate to AWS Client VPN:** Switching to AWS Client VPN allows remote employees to securely access the AWS environment. This migration would offload a significant amount of traffic from the on-premises data center, further reducing bandwidth constraints. AWS Client VPN is also easy to set up and manage, contributing to reduced operational complexity.

The reasons why the other options are less suitable:

- **A: AWS Storage Gateway Volume Gateway:** While this could integrate on-premises file servers with AWS, it doesn't directly address the high bandwidth usage issue as much as migrating to FSx for Windows File Server. It also adds some operational complexity in managing the gateway.
- **C: Amazon FSx for Lustre:** FSx for Lustre is tailored for high-performance computing and data-intensive workloads, not for typical Windows file storage needs. It's not the right fit for hosting Windows home directories.
- **E: AWS Direct Connect:** While Direct Connect provides a dedicated network connection to AWS, it's a more complex and costly solution that might not be necessary if the primary concern of bandwidth usage and operational overhead can be addressed by migrating to FSx for Windows File Server and AWS Client VPN. Direct Connect mainly benefits scenarios requiring consistent, high throughput with low latency rather than remote employee file access.

Question 69

A company utilizing multiple AWS accounts has identified a key security concern during a recent audit: numerous Amazon Elastic Block Store (Amazon EBS) volumes attached to Amazon EC2 instances are unencrypted. The company needs a solution not only to encrypt these existing unencrypted volumes but also to implement a system that proactively identifies and manages unencrypted volumes in the future. Furthermore, the company seeks a centralized management solution for its AWS accounts with a focus on compliance and security.

To address these requirements, the following steps are proposed:

A: Establish an organization in AWS Organizations and deploy AWS Control Tower. Activate the highly recommended controls (guardrails) and incorporate all AWS accounts into this organization, organizing them into Organizational Units (OUs).

B: Utilize the AWS Command Line Interface (CLI) to identify all unencrypted volumes across the AWS accounts and execute a script to encrypt these volumes in place.

C: Generate a snapshot for each unencrypted EBS volume. Create a new, encrypted volume from this snapshot. Then, detach the original volume and replace it with the newly encrypted one.

D: Form an organization in AWS Organizations and implement AWS Control Tower, enabling the mandatory controls (guardrails). Integrate all AWS accounts into this organization and categorize them into OUs.

E: Activate AWS CloudTrail and configure an Amazon EventBridge rule to automatically detect and encrypt unencrypted EBS volumes.

The optimal combination of steps to achieve these objectives with minimal operational overhead are **A: Establish AWS Organizations and Deploy AWS Control Tower with Recommended Controls** and **C: Encrypt Volumes via Snapshots and Replacement.**

- **A: Establish AWS Organizations and Deploy AWS Control Tower with Recommended Controls:** This approach provides a centralized framework for managing multiple AWS accounts with an emphasis on compliance and security. AWS Control Tower with its recommended guardrails offers proactive governance and oversight, which can include detecting and preventing the use of unencrypted EBS volumes across all accounts. This setup enhances security and compliance management across the organization.
- **C: Encrypt Volumes via Snapshots and Replacement:** This method directly addresses the issue of existing unencrypted EBS volumes. By creating encrypted volumes from snapshots of the unencrypted ones and then replacing the old volumes, this ensures that

all data is encrypted without interrupting services. It's a practical approach to securing existing data.

The reasons why the other options are less suitable:

- **B: Use AWS CLI for In-Place Encryption:** While using AWS CLI to list and encrypt unencrypted volumes seems straightforward, encrypting EBS volumes in place (without creating snapshots) isn't supported by AWS. Thus, this method isn't feasible.
- **D: Implement AWS Control Tower with Mandatory Controls:** While setting up AWS Control Tower is beneficial, relying solely on mandatory controls might not be sufficient for specific compliance and security needs, such as ensuring all EBS volumes are encrypted.
- **E: Activate AWS CloudTrail and EventBridge for Automatic Encryption:** AWS CloudTrail is instrumental in logging and monitoring account activity, and EventBridge can trigger responses to certain events. However, automatically encrypting unencrypted EBS volumes is not a native functionality provided by these services, making this option unviable for the specified requirements.

Question 70

A company operates an internal web application hosted on Amazon EC2 instances, which are managed through an Application Load Balancer (ALB). User authentication for this application is currently handled through an internal user database. The company now aims to shift the authentication process to leverage their existing AWS Directory Service for Microsoft Active Directory, ensuring that all users with accounts in this directory can access the application.

The company is evaluating the following solutions to achieve this:

A: Establish a new application client within the directory and set up a listener rule on the ALB using the 'authenticate-oidc' action. Configure this listener rule with the necessary details like issuer, client ID, and secret, as well as endpoint information pertinent to the Active Directory service. Also, configure the newly created app client with the callback URL provided by the ALB.

B: Implement an Amazon Cognito user pool and configure it with a federated identity provider (IdP) using metadata from the directory. Create an application client and associate it with this user pool. Then, establish a listener rule on the ALB using the 'authenticate-cognito' action and configure this rule to utilize the user pool and application client.

C: Integrate the directory as a new IAM identity provider (IdP) and create a new IAM role for SAML 2.0 federation. Set a role policy that permits access to the ALB and designate this new role as the default for authenticated users within the IdP. Then, create a listener rule on the ALB with the 'authenticate-oidc' action.

D: Activate AWS IAM Identity Center (formerly AWS Single Sign-On) and configure it with the directory as an external IdP using SAML, opting for automatic provisioning. Create a new IAM role for SAML 2.0 federation, attaching a role policy that allows ALB access, and link this role to all groups. Finally, establish a listener rule on the ALB with the 'authenticate-cognito' action.

The most suitable solution is **B: Implement Amazon Cognito User Pool with Federated IdP.**

- **B: Implement Amazon Cognito User Pool with Federated IdP:** This option is the best fit because Amazon Cognito seamlessly integrates with AWS Directory Service for Microsoft Active Directory. By configuring a Cognito user pool with a federated identity provider linked to Active Directory, the company can utilize existing user accounts for authentication. The integration of Cognito with the ALB through the 'authenticate-cognito' action allows for a smooth and secure authentication flow, meeting the requirement of using Active Directory accounts for application access.

The reasons why the other options are less suitable:

- **A: OIDC with ALB Listener Rule:** This approach assumes that AWS Directory Service for Microsoft Active Directory can act as an OpenID Connect (OIDC) provider, which is not typically the case. OIDC integration would require additional configurations that are not inherently supported by AWS Directory Service.
- **C: IAM IdP and OIDC Listener Rule:** Similar to option A, this method involves using OIDC with Active Directory, which is not a straightforward process. The use of SAML with IAM roles is more common for enterprise federation, but this does not align well with the ALB's OIDC authentication capabilities.
- **D: AWS IAM Identity Center with Cognito Action:** This option involves unnecessary complexity by introducing AWS IAM Identity Center (AWS SSO) and SAML federation, and then incorrectly pairing it with the 'authenticate-cognito' action on the ALB, which is not the standard practice for such integrations.

Question 71

A company operates a high-traffic website and has implemented a backend service architecture spanning a primary AWS Region and a separate disaster recovery (DR) Region. To deliver content, a single Amazon CloudFront distribution is in place. For backend service routing, the company utilizes Amazon Route 53, with a record set featuring health checks and a failover routing policy directed to the primary Region's backend service. This Route 53 record set is also designated as the origin for the CloudFront distribution. Additionally, there's a secondary failover record set pointing to the backend service in the DR Region, both record sets having a Time-to-Live (TTL) of 60 seconds. The current setup results in failover times exceeding 1 minute, and the company seeks a solution to expedite this process.

The company is considering the following options to achieve faster failover:

A: Introduce an additional CloudFront distribution and create a new Route 53 failover record set with health checks for both CloudFront distributions.

B: Adjust the TTL for the existing Route 53 record sets linked to the backend services in each region to 4 seconds.

C: Establish new record sets for the backend services using a latency routing policy and set these record sets as origins in the CloudFront distribution.

D: Configure a CloudFront origin group to include two origins, each corresponding to one of the backend service regions, and implement origin failover in the CloudFront distribution's cache behavior.

The most effective solution is **D: Configure a CloudFront Origin Group with Origin Failover.**

- **D: Configure a CloudFront Origin Group with Origin Failover:** This approach directly addresses the goal of rapid failover. By setting up an origin group in CloudFront that includes both the primary and DR region backend services as origins, and configuring origin failover, CloudFront can automatically route traffic to the secondary origin if the primary is unhealthy. This method provides a fast and seamless failover mechanism as the decision is made at the CDN level, rather than relying on DNS changes, which can be subject to propagation delays.

The reasons why the other options are less suitable:

- **A: Additional CloudFront Distribution and New Route 53 Record Set:** Creating an extra CloudFront distribution and associated Route 53 failover record sets adds unnecessary complexity and doesn't inherently provide a faster failover compared to using a single distribution with origin failover.
- **B: Reduce TTL to 4 Seconds:** Lowering the TTL of Route 53 record sets to 4 seconds can reduce the time it takes for DNS changes to propagate. However, this approach can lead to increased DNS queries and does not guarantee an immediate failover as it still depends on DNS propagation times.
- **C: New Record Sets with Latency Routing Policy:** Implementing latency routing policies for the backend services and using these as origins in CloudFront would optimize routing based on latency but doesn't address the failover speed requirement. Latency routing is different from failover routing and doesn't provide the same rapid response in the event of a primary region failure.

Question 72

A company managing multiple AWS accounts is looking to centralize the management of these accounts, particularly for their DevOps teams that operate both production and non-production workloads. They aim to limit the use of certain AWS services which the DevOps teams do not require, having already incorporated all their AWS accounts into AWS Organizations. The company's goal is to enable access to currently utilized services while explicitly denying access to specific services. Additionally, they wish to manage these multiple accounts as a unified entity.

To achieve these objectives, the company is contemplating the following actions:

A: Implement a Deny list approach.

B: Utilize AWS IAM Access Advisor to identify recently used services.

C: Consult the AWS Trusted Advisor report to find out which services have been recently used.

D: Eliminate the default FullAWSAccess Service Control Policy (SCP).

E: Organize member accounts into organizational units (OUs) within AWS Organizations.

F: Remove the default DenyAWSAccess SCP.

The most suitable combination of steps is **A: Implement a Deny List Strategy**, **B: Utilize AWS IAM Access Advisor**, and **E: Organize Accounts into OUs**.

- **A: Implement a Deny List Strategy:** This method involves creating Service Control Policies (SCPs) that explicitly deny access to specific AWS services which are not in use. This approach allows all other services by default, making it a practical way to restrict access to only a few selected services.
- **B: Utilize AWS IAM Access Advisor:** Access Advisor within IAM is a useful tool to review which services have been recently accessed by IAM users, roles, and groups. This information is critical in identifying the services currently in use, ensuring that these services are not mistakenly restricted when implementing the deny list SCPs.
- **E: Organize Accounts into OUs:** By structuring the AWS accounts into Organizational Units (OUs) within AWS Organizations, the company can efficiently administer multiple accounts as a single entity. This organization allows for the application of SCPs at the OU level, simplifying the management and enforcement of policies across multiple accounts.

The reasons why the other options are not suitable:

- **C: Consult AWS Trusted Advisor Report:** While Trusted Advisor provides valuable insights for optimizing AWS resources, it is more focused on cost optimization,

performance, and security, rather than providing detailed usage data of specific AWS services as IAM Access Advisor does.

- D: Eliminate Default FullAWSAccess SCP: The default FullAWSAccess SCP allows full access to all AWS services and actions. Removing this SCP without carefully configuring other SCPs could unintentionally block access to necessary services.
- F: Remove Default DenyAWSAccess SCP: AWS Organizations do not have a default DenyAWSAccess SCP. SCPs are used to whitelist or blacklist access to services, and the removal of a non-existent default deny policy is not applicable in this context.

Question 73

A company specializing in live events is developing a scalable solution for its ticketing application on AWS, catering to high traffic spikes during scheduled, one-time sale events. The application is currently deployed on Amazon EC2 instances within an Auto Scaling group and utilizes PostgreSQL for its database needs. The primary objective is to devise a scaling strategy that ensures maximum availability and performance during these high-demand sale events.

The company is evaluating the following approaches:

A: Implement a predictive scaling policy for the EC2 instances. Use Amazon Aurora PostgreSQL Serverless v2 with Multi-AZ deployment and auto-scalable read replicas. Establish an AWS Step Functions state machine to activate parallel AWS Lambda functions for pre-warming the database prior to sale events, triggered by an Amazon EventBridge rule.

B: Apply a scheduled scaling policy for the EC2 instances. Employ Amazon RDS for PostgreSQL with a Multi-AZ DB instance and auto-scalable read replicas. Configure an Amazon EventBridge rule to trigger an AWS Lambda function, which scales up by creating a larger read replica before each sale event, and fails over to it. Post-event, use another EventBridge rule to invoke a Lambda function for scaling down the read replica.

C: Use a predictive scaling policy for the EC2 instances. Opt for Amazon RDS for PostgreSQL with a Multi-AZ DB instance and auto-scalable read replicas. Set up an AWS Step Functions state machine to execute parallel AWS Lambda functions for database pre-warming before sale events, initiated by an Amazon EventBridge rule.

D: Implement a scheduled scaling policy for the EC2 instances. Host the database on an Amazon Aurora PostgreSQL Multi-AZ DB cluster. Set up an Amazon EventBridge rule to call an AWS Lambda function for creating a larger Aurora Replica before the sale event and perform a failover to this replica. Post-event, another EventBridge rule triggers a Lambda function to downscale the Aurora Replica.

The most appropriate solution is **D: Scheduled Scaling with Amazon Aurora PostgreSQL and Dynamic Replica Management.**

- **D: Scheduled Scaling with Amazon Aurora PostgreSQL and Dynamic Replica Management:** This option aligns closely with the needs of the company. Scheduled scaling allows precise control over EC2 instance scaling based on the known timing of sale events. Using Amazon Aurora PostgreSQL offers robust performance and reliability, especially for high-demand scenarios. The strategy of dynamically adjusting the size of an Aurora Replica in response to anticipated load (scaling up before the event and down afterwards) offers a tailored approach to handle peak demand efficiently. The use of EventBridge and Lambda functions for managing these operations provides automation and reduces manual intervention.

The reasons why the other options are less suitable:

- **A: Predictive Scaling with Serverless Aurora:** While predictive scaling and serverless Aurora offer scalability, predictive scaling might not be as precise as scheduled scaling for known event times. Additionally, Aurora Serverless v2, although highly scalable, might not offer the same level of control and predictability as managing replicas in a standard Aurora deployment.
- **B: Scheduled Scaling with RDS for PostgreSQL:** This approach does address the need for scheduled scaling but falls short in database management. While RDS for PostgreSQL is a solid choice, the process of manually managing read replicas and failover adds complexity and potential latency compared to the more automated and integrated capabilities of Aurora.
- **C: Predictive Scaling with RDS for PostgreSQL:** Similar to option A, predictive scaling does not provide the same level of control for known event schedules as scheduled scaling. The use of RDS for PostgreSQL with pre-warming lacks the efficiency and integration benefits of Aurora's replication management.

Question 74

A company is looking to implement a cloud-based backup for its on-premises intranet application, choosing AWS Elastic Disaster Recovery as the solution. The company's primary requirements are that the replication traffic to AWS must not traverse the public internet, the application should remain inaccessible from the internet, and the backup solution should not monopolize all the available network bandwidth, as other applications also depend on it.

To fulfill these requirements, the company is contemplating a combination of the following actions:

A: Set up a Virtual Private Cloud (VPC) in AWS with a minimum of two private subnets, including two NAT gateways and a virtual private gateway.

B: Establish a VPC with at least two public subnets, adding both a virtual private gateway and an internet gateway.

C: Create an AWS Site-to-Site VPN connection between the on-premises network and the AWS network.

D: Implement an AWS Direct Connect link, along with a Direct Connect gateway, between the on-premises network and the AWS network.

E: Opt for private IP addresses for data replication during the replication servers' configuration.

F: Configure the launch settings for the target servers in a way that the Recovery instance's private IP address is aligned with the source server's private IP address.

The most suitable combination of steps is **A: Create a VPC with Private Subnets and Gateways**, **D: Implement AWS Direct Connect**, and **E: Use Private IP Addresses for Replication**.

- **A: Create a VPC with Private Subnets and Gateways:** This setup ensures that the replicated data remains within the private network space in AWS, without exposure to the public internet. The inclusion of NAT gateways and a virtual private gateway supports secure and private communication with the on-premises network.
- **D: Implement AWS Direct Connect:** Direct Connect provides a dedicated network connection between the on-premises network and AWS. This connection is crucial for meeting the requirement of keeping replication traffic off the public internet, offering a more reliable and consistent network experience, which is especially important for disaster recovery scenarios.
- **E: Use Private IP Addresses for Replication:** Configuring the replication servers to use private IP addresses ensures that the replication traffic remains internal and does not expose the application to the internet, aligning with the company's security requirements.

The reasons why the other options are less suitable:

- **B: VPC with Public Subnets and Gateways:** Creating a VPC with public subnets and an internet gateway would potentially expose the replication traffic to the internet, which contradicts the company's requirement for privacy.
- **C: Site-to-Site VPN Connection:** While a VPN connection would keep replication traffic off the public internet, it is generally less reliable and slower compared to Direct

Connect, especially for the bandwidth requirements of disaster recovery.

- **F: Matching Recovery Instance's Private IP with Source Server:** This step focuses on IP address alignment but does not directly contribute to the primary requirements of secure, private replication and bandwidth management.

Question 75

A company specializing in image storage services is planning to implement a customer-oriented solution on AWS, catering to millions of users. The solution will be responsible for processing large batches of image files: resizing these files and then storing them in an Amazon S3 bucket for a duration of up to 6 months. The solution needs to efficiently manage fluctuating demand levels and must be robust enough for enterprise-scale operations, including the capability to reprocess jobs in case of any failures.

The company is considering the following options to fulfill these requirements:

A: Utilize AWS Step Functions to trigger a process when an image is uploaded to S3. Employ an AWS Lambda function to resize the image, replacing the original in the S3 bucket. Implement an S3 Lifecycle policy to automatically delete images after 6 months.

B: Implement Amazon EventBridge to handle the event of an image upload to S3. Use an AWS Lambda function for resizing the image, overwriting the original file in the S3 bucket. Set up an S3 Lifecycle policy for expiring images after 6 months.

C: Configure S3 Event Notifications to invoke an AWS Lambda function upon image upload to S3. The Lambda function resizes the image, retaining the original in the S3 bucket. Establish an S3 Lifecycle policy to move images to S3 Standard-Infrequent Access (S3 Standard-IA) after 6 months.

D: Employ Amazon Simple Queue Service (SQS) to manage the S3 event triggered by image uploads. Use an AWS Lambda function to resize the image and store the resized version in an S3 bucket using S3 Standard-Infrequent Access (S3 Standard-IA). Set a Lifecycle policy to transfer images to S3 Glacier Deep Archive after 6 months.

The most cost-effective solution is **A: AWS Step Functions with S3 Event Trigger and Lambda for Resizing, and S3 Lifecycle Policy for Expiration.**

- **A: AWS Step Functions with S3 Event Trigger and Lambda for Resizing, and S3 Lifecycle Policy for Expiration:** This solution efficiently handles the varying demand and ensures reliability. AWS Step Functions can orchestrate the image processing workflow, providing a robust way to manage and rerun failed processing jobs. Using Lambda for image resizing is scalable and cost-effective, as it only incurs costs when it's running.

The S3 Lifecycle policy for automatically deleting images after 6 months is a simple and effective way to manage storage costs.

The reasons why the other options are less suitable:

- **B: EventBridge with Lambda and S3 Lifecycle Policy:** Although Amazon EventBridge could trigger the image processing workflow, it doesn't offer the same level of workflow orchestration and error handling as AWS Step Functions, which is crucial for managing large-scale, enterprise-grade processing jobs.
- **C: S3 Event Notifications with Lambda and Lifecycle to S3 Standard-IA:** This approach lacks an efficient workflow management system like Step Functions. Additionally, moving images to S3 Standard-IA after 6 months doesn't align with the requirement to store files only for up to 6 months.
- **D: SQS with Lambda and Lifecycle to S3 Glacier Deep Archive:** Using SQS for managing the S3 events introduces unnecessary complexity for this use case. Furthermore, moving images to S3 Glacier Deep Archive after 6 months contradicts the requirement to store them for only up to 6 months. This option also potentially incurs higher costs due to the usage of S3 Glacier Deep Archive.

Question 76

A company, organized into distinct departments each with their own AWS account under AWS Organizations, seeks to optimize and effectively manage its compute costs while maintaining transparency in billing at the departmental level. The company wishes to achieve this without sacrificing the flexibility in choosing compute resources that its various application teams currently enjoy.

The company is considering the following strategies to meet these goals:

A: Implement AWS Budgets for individual departments, utilize AWS Tag Editor for resource tagging, and invest in EC2 Instance Savings Plans.

B: Enable consolidated billing in AWS Organizations, establish a resource tagging system for departmental identification, enforce resource tagging through Service Control Policies (SCPs), and invest in EC2 Instance Savings Plans.

C: Enable consolidated billing in AWS Organizations, develop a resource tagging system to distinguish departments, apply tags to resources using AWS Tag Editor, and purchase Compute Savings Plans.

D: Set up AWS Budgets for each department, use SCPs for resource tagging, and opt for Compute Savings Plans.

The most comprehensive solution is **C: Consolidated Billing with Tagging and Compute Savings Plans.**

- **C: Consolidated Billing with Tagging and Compute Savings Plans:** This approach addresses all of the company's requirements effectively. Consolidated billing through AWS Organizations enables centralized management of costs across all departmental accounts. Implementing a consistent tagging strategy aids in tracking and allocating costs accurately to each department. Using AWS Tag Editor simplifies the application of these tags to resources. Opting for Compute Savings Plans offers cost savings on compute usage across a variety of AWS services and instance types, providing the desired flexibility in resource selection.

The reasons why the other options are less suitable:

- **A: AWS Budgets and EC2 Instance Savings Plans:** While AWS Budgets and Savings Plans are useful, this approach lacks the consolidated billing aspect, which is crucial for centralized cost management. EC2 Instance Savings Plans are also more restrictive compared to Compute Savings Plans as they are specific to EC2 instances.
- **B: SCPs for Tagging and EC2 Instance Savings Plans:** This option includes consolidated billing but relies on SCPs for tagging, which is not their primary use case. SCPs are more about granting or denying permissions rather than tagging. Additionally, like option A, the choice of EC2 Instance Savings Plans limits flexibility compared to Compute Savings Plans.
- **D: AWS Budgets and SCPs with Compute Savings Plans:** While this option includes Compute Savings Plans, it omits the benefits of consolidated billing and improperly uses SCPs for tagging. Without consolidated billing, managing costs across multiple accounts becomes more complex.

Question 77

A company operates a web application that facilitates secure uploading of pictures and videos to an Amazon S3 bucket. Currently, this process is designed to allow only authenticated users to upload content. The application accomplishes this by generating presigned URLs, through which users upload objects directly from a browser interface. However, there have been numerous complaints about slow upload speeds, particularly for files exceeding 100 MB. The company is seeking a solution to enhance upload performance while continuing to ensure that only authenticated users can post content.

The company is considering the following options:

A: Implement an Amazon API Gateway with an edge-optimized API endpoint, set up as an S3 service proxy. Configure the PUT method for this resource to directly perform the S3

PutObject operation. Secure access to the API Gateway using a COGNITO_USER_POOLS authorizer and modify the browser interface to upload objects via API Gateway instead of using presigned URLs.

B: Establish an Amazon API Gateway with a regional API endpoint, also configured as an S3 service proxy. Arrange for the PUT method on this resource to execute the S3 PutObject operation. Protect the API Gateway using an AWS Lambda authorizer and adjust the browser interface to utilize API Gateway for object uploads in place of presigned URLs.

C: Enable S3 Transfer Acceleration on the S3 bucket and generate presigned URLs incorporating the transfer acceleration endpoint. Adapt the browser interface to upload objects to this accelerated URL using the S3 multipart upload API.

D: Set up an Amazon CloudFront distribution targeting the S3 bucket, allowing PUT and POST methods in the CloudFront cache behavior. Alter the CloudFront origin to employ an origin access identity (OAI) and grant the OAI PutObject permissions via the bucket policy. Redirect the browser interface to upload objects through the CloudFront distribution.

The optimal solution is **C: Enable S3 Transfer Acceleration and Use Multipart Upload API.**

- **C: Enable S3 Transfer Acceleration and Use Multipart Upload API:** This approach directly targets the issue of slow upload speeds for large files. S3 Transfer Acceleration optimizes the transfer of files over long distances between the user's browser and the S3 bucket. By utilizing the multipart upload API, large files are divided into smaller parts and uploaded in parallel, significantly improving upload speeds. This method maintains the security requirement of allowing only authenticated users to post content, as presigned URLs, which include authentication, are still used.

The reasons why the other options are less suitable:

- **A & B: Amazon API Gateway with S3 Service Proxy (Edge-Optimized and Regional):** While using API Gateway with an S3 service proxy might seem like a viable approach, it introduces unnecessary complexity and potential latency due to the additional layer between the user and S3. Additionally, it does not specifically address the issue of slow uploads for large files.
- **D: Amazon CloudFront Distribution for S3 Bucket:** Configuring a CloudFront distribution for the S3 bucket might improve the performance of uploads due to CloudFront's distributed nature. However, it is more suited for content delivery (downloads) rather than uploads. Moreover, this approach does not leverage the multipart upload capability, which is particularly beneficial for large files.

Question 78

A large corporation is in the process of transitioning its entire IT infrastructure to AWS. Currently, each of its business units operates independently with its own AWS account, housing both development and testing environments. The company plans to introduce new accounts specifically for production workloads. From a financial standpoint, the company desires a unified payment system, yet it's essential for them to track and allocate expenses for each business unit separately. Simultaneously, the security team is seeking a unified approach to manage IAM (Identity and Access Management) across all AWS accounts in the organization.

To satisfy these needs with minimal effort, the company is contemplating the following strategies:

A: Implement a set of AWS CloudFormation templates with predefined IAM permissions and deploy them across each account. Mandate the use of these templates in both existing and new accounts to ensure adherence to a least privilege access model.

B: Leverage AWS Organizations to create a new organizational structure starting with a primary payer account. Develop an organizational unit (OU) hierarchy and invite existing accounts to join this organization, creating new accounts through AWS Organizations.

C: Maintain separate AWS accounts for each business unit. Implement tagging for each account and utilize AWS Cost Explorer to manage cost allocation and chargebacks.

D: Activate all features of AWS Organizations and implement service control policies (SCPs) to govern IAM permissions in subsidiary accounts.

E: Consolidate all business units into a single AWS account. Use tagging for billing and employ IAM's Access Advisor for enforcing a least privilege access model.

The most effective combination of steps is **B: Create Organization and OU Hierarchy in AWS Organizations** and **D: Enable All Features in AWS Organizations and Establish SCPs.**

- **B: Create Organization and OU Hierarchy in AWS Organizations:** This approach allows the company to centralize billing and account management, maintaining financial visibility for each group's spending. By setting up an organizational structure and inviting existing accounts to join, the company can also streamline the creation of new accounts for production workloads. This structure supports centralized governance while preserving the autonomy of individual business units.
- **D: Enable All Features in AWS Organizations and Establish SCPs:** Enabling all features in AWS Organizations, especially SCPs, provides the security team with a centralized mechanism to control IAM usage across all accounts. SCPs enable the security team to set permission boundaries for IAM entities, ensuring compliance with the company's security policies across its entire AWS environment.

The reasons why the other options are less suitable:

- **A: Parameterized CloudFormation Templates for IAM Permissions:** While CloudFormation templates can standardize IAM permissions, this approach requires significant effort to implement and maintain across multiple accounts. It also lacks the centralized control and scalability offered by AWS Organizations and SCPs.
- **C: Separate Business Unit Accounts with Cost Explorer for Chargebacks:** Maintaining separate accounts for each business unit without the structure of AWS Organizations makes centralized billing and cost allocation more complex and labor-intensive.
- **E: Single AWS Account with Tagging and Access Advisor:** Consolidating all units into a single AWS account is impractical for a large company due to operational and security risks. It significantly increases the complexity of managing resources, permissions, and billing.

Question 79

A company operating a weather data analysis service is encountering an issue with its current system, where data from numerous weather stations is transmitted via an Amazon API Gateway REST API. This API integrates with an AWS Lambda function, which in turn relies on a third-party service for initial data processing. However, this third-party service occasionally becomes overwhelmed, leading to pre-processing failures and consequent data loss. The company seeks a solution to enhance the system's resilience by ensuring no data loss occurs and that unprocessed data can be handled at a later time in case of such failures.

The company is evaluating these options:

A: Implement an Amazon Simple Queue Service (Amazon SQS) queue and configure it as the dead-letter queue (DLQ) for the API.

B: Create two Amazon SQS queues: a primary queue and a secondary DLQ. Modify the API to integrate with the primary queue, and set the Lambda function to trigger from the primary queue. Assign the secondary queue as the DLQ for the primary queue.

C: Establish two Amazon EventBridge event buses – a primary and a secondary. Update the API integration to target the primary event bus. Set up an EventBridge rule to respond to all events on the primary bus, designating the Lambda function as the rule's target. Use the secondary event bus as the failure destination for the Lambda function.

D: Set up a custom Amazon EventBridge event bus and configure it as the failure destination for the Lambda function.

The most effective solution is **B: Two SQS Queues with Primary Queue Integrated in API and Lambda Triggered from Primary Queue.**

- **B: Two SQS Queues with Primary Queue Integrated in API and Lambda Triggered from Primary Queue:** This approach provides a robust mechanism to handle data processing failures. By directing the incoming data to the primary SQS queue via the API, the data is safely stored in the queue, preventing any immediate loss due to third-party service failure. The Lambda function can then process the data from the primary queue. If processing fails, the data is moved to the secondary DLQ, ensuring that it can be retried or processed later. This method effectively decouples the data ingestion and processing stages, adding resilience to the system.

The reasons why the other options are less suitable:

- **A: SQS as DLQ for API:** Configuring an SQS queue as a DLQ for the API does not address the primary issue of Lambda function failures. This setup would only capture failed API requests, not failures in data processing by the Lambda function.
- **C: Two EventBridge Event Buses:** While using EventBridge event buses could provide a way to handle events, this setup complicates the architecture without offering significant benefits over the SQS-based solution. EventBridge is more suited for event routing and filtering rather than simple queueing and does not inherently provide the same queueing and retry mechanisms as SQS.
- **D: Custom EventBridge Event Bus as Failure Destination:** Configuring a custom EventBridge event bus as a failure destination for the Lambda function does not adequately address the issue of storing and retrying unprocessed data. It lacks the queuing functionality that SQS provides, which is crucial for ensuring data is not lost and can be processed later.

Question 80

A company managing job boards for seasonal workers is experiencing a surge in online traffic and activity. Their existing system comprises backend services running on two Amazon EC2 instances, coordinated by an Application Load Balancer, and utilizing Amazon DynamoDB for data storage. However, during peak season, the application experiences slow read and write operations. The company seeks a scalable architectural solution to efficiently manage the high seasonal demand while aiming for minimal development complexity.

The company is considering the following strategies:

A: Shift the backend services to AWS Lambda and enhance the read and write capacity of DynamoDB.

B: Transition the backend services to AWS Lambda and configure DynamoDB to employ global tables.

C: Implement Auto Scaling groups for the backend services coupled with DynamoDB auto scaling.

D: Apply Auto Scaling groups for the backend services and integrate Amazon Simple Queue Service (SQS) with an AWS Lambda function for DynamoDB writes.

The most fitting solution is **C: Auto Scaling Groups for Backend Services and DynamoDB Auto Scaling.**

- **C: Auto Scaling Groups for Backend Services and DynamoDB Auto Scaling:** This option directly addresses the need for scalability during peak seasons with minimal development effort. By using Auto Scaling groups, the number of EC2 instances can dynamically adjust based on the application load, providing the necessary compute resources. DynamoDB auto scaling adapts the database's read and write capacity in response to the changing workload patterns, ensuring efficient database performance. This approach offers a balanced and easily implementable solution to handle increased traffic without significant changes to the existing architecture.

The reasons why the other options are less suitable:

- **A: Migrate to AWS Lambda and Increase DynamoDB Capacity:** While migrating to AWS Lambda could offer scalability, it involves a significant shift in the architectural approach and potentially higher development efforts. Simply increasing the read/write capacity of DynamoDB doesn't provide the dynamic scalability needed for varying seasonal demands.
- **B: Migrate to AWS Lambda and Configure DynamoDB Global Tables:** This approach, similar to option A, requires a considerable change to the application architecture. DynamoDB global tables are designed for multi-region replication and might not be necessary for the company's current scalability issues.
- **D: Auto Scaling with SQS and Lambda for DynamoDB Writes:** Integrating SQS and Lambda for DynamoDB writes introduces additional complexity and might not directly address the slow read/write issues. While it could help decouple write operations, this setup requires more development work and doesn't address the scaling of read operations.

Question 81

An ecommerce company with a Java-based website hosted on AWS experienced user-reported errors and timeouts during a high-traffic promotional event. Their infrastructure consists of an Amazon CloudFront distribution, an Apache web server layer running on Amazon EC2 instances within an Auto Scaling group, and a backend Amazon Aurora MySQL database. Post-event, the operations team faced challenges in diagnosing the issues due to

the loss of logs from terminated EC2 instances and insufficient query performance data from the Aurora database. To enhance visibility into application performance during future peak traffic events, the company is exploring various solutions.

The company is considering these options:

A: Set up the Aurora MySQL database to forward slow query and error logs to Amazon CloudWatch Logs.

B: Integrate AWS X-Ray SDK to monitor incoming HTTP requests on the EC2 instances and trace SQL queries using the X-Ray SDK for Java.

C: Configure the Aurora MySQL database to stream slow query and error logs to Amazon Kinesis.

D: Install the Amazon CloudWatch Logs agent on the EC2 instances for transmitting Apache logs to CloudWatch Logs.

E: Activate and tailor AWS CloudTrail to gather and assess application activities from Amazon EC2 and Aurora.

F: Enable performance benchmarking for the Aurora MySQL database and stream the results to AWS X-Ray.

The best combination of steps to improve visibility into application performance is **A: Configure Aurora MySQL to Publish Logs to CloudWatch**, **B: Implement AWS X-Ray SDK for Tracing**, and **D: Use CloudWatch Logs Agent for EC2 Apache Logs.**

- **A: Configure Aurora MySQL to Publish Logs to CloudWatch:** Forwarding slow query and error logs from the Aurora database to CloudWatch Logs is an efficient way to track database performance issues. This allows for easier access and analysis of database logs, particularly useful in understanding the root causes of slow queries during high-load periods.
- **B: Implement AWS X-Ray SDK for Tracing:** Using AWS X-Ray SDK on EC2 instances helps in tracing HTTP requests and SQL queries. This provides a detailed overview of the application's performance, including insights into how backend processes such as database queries are affecting overall application response times.
- **D: Use CloudWatch Logs Agent for EC2 Apache Logs:** Installing the CloudWatch Logs agent on EC2 instances ensures that web server logs are consistently sent to CloudWatch Logs. This approach ensures log preservation even if the EC2 instances are terminated, allowing for post-event analysis.

The reasons why the other options are less suitable:

- **C: Stream Logs to Amazon Kinesis:** While streaming logs to Kinesis can be part of a complex log analysis solution, it is not necessary for this use case. CloudWatch Logs provides sufficient capabilities for log storage and analysis without the additional complexity of Kinesis.
- **E: Use AWS CloudTrail for EC2 and Aurora Activities:** CloudTrail is primarily used for monitoring API calls and management operations in AWS, not for application-level monitoring or log analysis. It would not provide the detailed application performance metrics needed.
- **F: Stream Aurora Performance Data to AWS X-Ray:** While X-Ray is a powerful tool for application performance analysis, it's not designed to directly receive and process database performance benchmarking data. X-Ray is better suited for tracing requests and analyzing service interactions.

Question 82

A company planning to transition to the cloud aims to assess the configurations of its current virtual machine infrastructure in the data center to accurately determine the sizing for new Amazon EC2 instances. Key metrics to be collected include CPU, memory, and disk usage, along with a record of running processes on each VM. Additionally, the company is interested in tracking network connections to understand the inter-server communication patterns.

The company is evaluating these methods to gather the required data in a cost-effective manner:

A: Utilize AWS Application Discovery Service and install its data collection agent on each virtual machine within the data center.

B: Install the Amazon CloudWatch agent on all local servers and configure it to send metrics to Amazon CloudWatch Logs.

C: Employ AWS Application Discovery Service and activate agentless discovery for the current virtualization setup.

D: Activate AWS Application Discovery Service via the AWS Management Console and adjust the corporate firewall to permit scanning over a VPN.

The most cost-effective approach is **A: Deploy AWS Application Discovery Service Data Collection Agent on Each VM.**

- **A: Deploy AWS Application Discovery Service Data Collection Agent on Each VM:** This solution is specifically designed for the scenario of assessing on-premises environments for cloud migration. The AWS Application Discovery Service's agent can gather detailed information about the configurations and usage of VMs, including CPU, memory, disk

utilization, network connections, and running processes. This comprehensive data collection facilitates informed decision-making about EC2 instance sizing and migration planning.

The reasons why the other options are less suitable:

- **B: Install CloudWatch Agent on Local Servers:** While the CloudWatch agent can collect metrics like CPU, memory, and disk usage, it's primarily geared towards monitoring resources already in AWS or extended cloud environments. It doesn't provide the same level of detailed discovery, particularly for process and network communication mapping, as the AWS Application Discovery Service.
- **C: Use Agentless Discovery in AWS Application Discovery Service:** Agentless discovery, while useful, might not provide as comprehensive a dataset as the agent-based method, especially regarding process inventory and certain detailed performance metrics.
- **D: Enable Discovery Service with Firewall Configuration for VPN Scans:** This approach does not accurately represent the functionality of AWS Application Discovery Service. The service does not scan over VPNs or require firewall configurations for scanning. This method would not effectively gather the detailed data required for the company's assessment.

Question 83

A company offering a SaaS application, hosted on AWS with Amazon EC2 instances behind a Network Load Balancer (NLB) and distributed across three Availability Zones in a single AWS Region, is expanding its application to multiple regions. To facilitate this expansion, the company requires a solution that provides static IP addresses for the application, which customers can use for their allow lists. Additionally, this solution should automatically direct customers to the geographically nearest region.

The company is exploring the following options to meet these needs:

A: Set up an Amazon CloudFront distribution and create a CloudFront origin group, adding the NLBs of each additional region to this group. Then, give customers the IP address ranges of CloudFront's edge locations.

B: Implement an AWS Global Accelerator standard accelerator, creating endpoints for the NLB in each additional region. Distribute the Global Accelerator's IP address to customers.

C: Create an Amazon CloudFront distribution with a custom origin for the NLB in each additional region. Provide the IP address ranges of the CloudFront distribution's edge locations to customers.

D: Establish an AWS Global Accelerator custom routing accelerator, add a listener to it, and include the IP addresses and ports for the NLB in each region. Share the Global Accelerator's IP address with customers.

The most suitable solution is **B: AWS Global Accelerator Standard Accelerator with NLB Endpoints in Additional Regions.**

- **B: AWS Global Accelerator Standard Accelerator with NLB Endpoints in Additional Regions:** This solution aligns well with the company's requirements. AWS Global Accelerator provides static IP addresses that can be used by customers for their allow lists. It also intelligently routes user traffic to the nearest regional endpoint, ensuring optimal performance. By creating a standard accelerator with endpoints for the NLB in each region, the company can effectively manage traffic distribution across multiple regions with minimal latency.

The reasons why the other options are less suitable:

- **A and C: Amazon CloudFront Distribution:** While CloudFront is an effective content delivery network, it does not provide static IP addresses for direct use by customers. CloudFront uses a network of edge locations, and its IP address ranges are not static, which would not meet the company's requirement for static IPs for allow lists.
- **D: AWS Global Accelerator Custom Routing Accelerator:** The custom routing accelerator is used for more specific, fine-grained routing needs and does not align with the company's requirement for straightforward regional traffic distribution. The standard accelerator is more suited for the company's needs of geographic routing and static IP provision.

Question 84

A company utilizing the AWS Cloud for multiple workloads has experienced an operational issue. The company, structured with distinct software development units, uses AWS Organizations for resource management and SAML-based federation for access permissions. Each development unit deploys production workloads into a shared production AWS account. A recent incident occurred where a member of one development unit accidentally terminated an EC2 instance that was under the purview of another unit. To prevent such incidents in the future, while still permitting developers to manage their respective instances, the company is looking for an effective solution.

The company is considering these options:

A: Set up individual Organizational Units (OUs) within AWS Organizations for each development unit and allocate these OUs to separate AWS accounts. Create Service Control Policies (SCPs) with deny actions and StringNotEquals conditions based on a

'DevelopmentUnit' resource tag that corresponds to each unit's name. Apply these SCPs to the relevant OUs.

B: During SAML federation, use AWS Security Token Service (AWS STS) to add a 'DevelopmentUnit' attribute as a session tag. Modify the IAM policies for the developers' assumed IAM roles to include deny actions and StringNotEquals conditions based on the 'DevelopmentUnit' resource tag and 'aws:PrincipalTag/DevelopmentUnit'.

C: Include a 'DevelopmentUnit' attribute as an AWS STS session tag during SAML federation. Create an SCP with allow actions and StringEquals conditions for the 'DevelopmentUnit' resource tag and 'aws:PrincipalTag/DevelopmentUnit', and assign this SCP to the root OU.

D: Develop separate IAM policies for each development unit, incorporating allow actions and StringEquals conditions for the 'DevelopmentUnit' resource tag corresponding to each unit. Assign these IAM policies to the assumed IAM roles during SAML federation, matching the development unit's name.

The most effective strategy is **B: Add 'DevelopmentUnit' Session Tags During SAML Federation and Update IAM Policies.**

- **B: Add 'DevelopmentUnit' Session Tags During SAML Federation and Update IAM Policies:** This solution leverages session tags to securely identify resources associated with specific development units. By updating IAM policies to include conditional deny actions based on these tags, it ensures that developers can only manage EC2 instances belonging to their respective units. This approach provides fine-grained access control without restructuring the existing AWS account setup or adding complexity with multiple OUs and SCPs.

The reasons why the other options are less suitable:

- **A: Separate OUs and SCPs Based on Resource Tags:** While creating separate OUs for each development unit can offer organized structuring, the use of SCPs with deny actions based on resource tags could become complex and harder to manage. It also introduces an unnecessary restructuring of the AWS accounts.
- **C: Allow SCPs with 'DevelopmentUnit' Tags for Root OU:** Assigning allow SCPs to the root OU based on session tags is less effective in preventing cross-unit resource access. SCPs are more suitable for broad, organization-level permissions and not for fine-grained access control at the resource level.
- **D: Separate IAM Policies for Each Unit During SAML Federation:** Assigning different IAM policies to each development unit's IAM role is a viable approach but less streamlined compared to using session tags. It requires maintaining multiple IAM policies

and ensuring accurate assignment during federation, which can be more labor-intensive and error-prone.

Question 85

An enterprise-level company is in the process of developing a platform for its users to facilitate the deployment of AWS infrastructure services. The company's primary objectives include:

1. Ensuring that users have the least privilege access necessary for launching AWS infrastructure, preventing them from provisioning unapproved services.
2. Centralizing the management of infrastructure service creation.
3. Enabling the distribution of these infrastructure services across multiple accounts within AWS Organizations.
4. Mandating the application of specific tags on any infrastructure initiated by users.

To achieve these goals, the company is evaluating a mix of AWS services and strategies:

A: Create AWS CloudFormation templates for infrastructure services, store them in an Amazon S3 bucket, and modify the bucket policy to grant access to specified IAM roles or users.

B: Develop infrastructure services as AWS CloudFormation templates and upload them as products to AWS Service Catalog portfolios in a central account. Share these portfolios with accounts in the AWS Organizations structure.

C: Assign IAM roles to users with permissions for full AWS CloudFormation access and read-only access to Amazon S3. Implement an SCP at the AWS account root user level to restrict services to only AWS CloudFormation and Amazon S3.

D: Restrict user IAM roles to only have ServiceCatalogEndUserAccess permissions. Utilize an automation script to import central portfolios into local AWS accounts, replicate the TagOption, assign user access, and apply launch constraints.

E: Employ the AWS Service Catalog TagOption Library to manage a list of required tags. Attach these TagOptions to AWS Service Catalog products or portfolios.

F: Implement the AWS CloudFormation Resource Tags property to enforce tag application on CloudFormation templates designed for users.

The best combination of actions to fulfill the company's requirements is **B: Use AWS CloudFormation Templates in AWS Service Catalog**, **D: Restrict IAM Roles to ServiceCatalogEndUserAccess and Automate Portfolio Import**, and **E: Utilize AWS Service Catalog TagOption Library for Tag Management**.

- **B: Use AWS CloudFormation Templates in AWS Service Catalog:** By using AWS Service Catalog, the company can manage and distribute CloudFormation templates as standardized products, ensuring users can deploy only approved services. This approach centralizes the control and distribution of infrastructure services across multiple accounts.
- **D: Restrict IAM Roles to ServiceCatalogEndUserAccess and Automate Portfolio Import:** Assigning users only ServiceCatalogEndUserAccess permissions limits their ability to launch services outside of what's provided in the Service Catalog. Automating the portfolio import process into local accounts streamlines the distribution and access management of these services.
- **E: Utilize AWS Service Catalog TagOption Library for Tag Management:** The TagOption Library in AWS Service Catalog allows the company to define and enforce a standardized set of tags. These tags can be applied to Service Catalog products or portfolios, ensuring compliance with the company's tagging requirements.

The reasons why the other options are less suitable:

- **A: Store Templates in S3 and Modify Bucket Policy:** While storing CloudFormation templates in S3 is feasible, it doesn't provide the same level of service control and distribution management as Service Catalog. It lacks the ability to enforce standardized tagging and user permissions effectively.
- **C: Grant CloudFormation and S3 Access, Implement SCPs:** Providing broad CloudFormation and S3 access to users can contradict the principle of least privilege. SCPs can restrict service usage but don't offer the fine-grained control and tagging enforcement capabilities of Service Catalog.
- **F: Enforce Tags with CloudFormation Resource Tags Property:** While using the Resource Tags property in CloudFormation templates enforces tagging at the template level, it doesn't provide the centralized management and enforcement capabilities that Service Catalog offers.

Question 86

A company has recently launched a new web application and implemented AWS WAF for security, with its logs being stored in Amazon S3 via Amazon Kinesis Data Firehose. To analyze these logs, the company has been using Amazon Athena to execute a daily query retrieving data from the past 24 hours. Despite the consistent volume of logs each day, the time taken for the query to execute has been increasing progressively. The company is now looking for a solution to halt this increasing query time, prioritizing minimal operational effort.

The company is evaluating these potential solutions:

A: Implement an AWS Lambda function to amalgamate each day's AWS WAF logs into a single log file.

B: Decrease the volume of data scanned by directing AWS WAF to forward logs to a new S3 bucket daily.

C: Adjust the Kinesis Data Firehose setup to segment the data in Amazon S3 based on date and time, establish external tables in Amazon Redshift, and utilize Amazon Redshift Spectrum for querying the data.

D: Reconfigure Kinesis Data Firehose and the Athena table to partition data by date and time, and modify the Athena query to focus on relevant partitions.

The most suitable solution is **D: Adjust Kinesis Data Firehose and Athena Table for Data Partitioning and Refine Athena Query.**

- **D: Adjust Kinesis Data Firehose and Athena Table for Data Partitioning and Refine Athena Query:** This approach directly addresses the issue of increased query times. By partitioning the data by date and time, Athena can efficiently query only the relevant subset of data, significantly reducing the time taken for execution. This method is operationally efficient as it leverages Athena's ability to work with partitioned data, thus minimizing the need for additional resources or complex configurations.

The reasons why the other options are less suitable:

- **A: Consolidate Logs into a Single File with Lambda:** While merging logs into a single file could streamline file management, it does not inherently address the issue of increasing query times. As data accumulates, the single file would become increasingly large, potentially exacerbating the issue.
- **B: Direct Logs to Different Buckets Daily:** Sending logs to a new bucket each day could complicate data management and does not leverage Athena's ability to efficiently query large datasets. This approach might also increase storage costs and complicate data retrieval.
- **C: Use Redshift Spectrum to Query Data:** Transitioning to Amazon Redshift Spectrum involves a significant change in the data analytics infrastructure, which might be more complex and costly than necessary. While Redshift Spectrum is powerful for large-scale data analysis, it adds complexity and may not be required for the company's current needs.

Question 87

A company is building a web application that will be hosted on Amazon EC2 instances, organized within an Auto Scaling group and accessible via a public Application Load Balancer

(ALB). The company wants to ensure that only users located in a specific country can access this application. Additionally, they require a solution to log any access attempts that are denied, preferring an option that involves minimal ongoing maintenance.

To achieve these objectives, the company is considering the following options:

A: Create a list (IPSet) of IP addresses corresponding to the specified country. Set up an AWS WAF web ACL and design a rule to reject requests not originating from the IP addresses in the IPSet. Link this rule with the web ACL and associate the web ACL with the ALB.

B: Establish an AWS WAF web ACL and formulate a rule to block access from locations outside the specified country. Integrate this rule with the web ACL, and then associate the web ACL with the ALB.

C: Implement AWS Shield with a configuration to block requests originating outside the specified country and associate AWS Shield with the ALB.

D: Develop a security group rule that permits traffic on ports 80 and 443 exclusively from IP ranges within the designated country. Apply this security group to the ALB.

The most appropriate solution is **B: Set Up AWS WAF Web ACL with Geographic Blocking Rule and Associate with ALB.**

- **B: Set Up AWS WAF Web ACL with Geographic Blocking Rule and Associate with ALB:** This option aligns well with the company's requirements. AWS WAF supports creating geographic-based blocking rules that can restrict access to the application based on the country of origin. Associating such a web ACL with the ALB ensures that only traffic from the specified country is allowed, while also providing the functionality to log blocked access attempts. This solution is straightforward to implement and requires minimal maintenance.

The reasons why the other options are less suitable:

- **A: Use AWS WAF with IPSet for Country-Specific IPs:** While this method could theoretically achieve the desired outcome, maintaining an IPSet with all IPs from a specific country is impractical due to the vast and dynamic nature of IP address allocations. It would also require continual updates, increasing maintenance efforts.
- **C: Configure AWS Shield for Geographic Blocking:** AWS Shield primarily provides protection against DDoS attacks and does not offer geographic blocking capabilities. It is not suitable for controlling application access based on user location.
- **D: Configure ALB Security Group with Country-Specific IP Ranges:** Similar to option A, maintaining a security group with all IPs from a specific country is unfeasible and would

require constant updating. Additionally, security groups do not inherently provide logging capabilities for blocked requests.

Question 88

A company is transitioning an application from on-premises to AWS, focusing on ensuring high availability and disaster recovery for its legacy Windows file server that stores sensitive, business-critical data. The data must not be transmitted over the public internet due to compliance mandates. The company prefers AWS managed services and plans to use Amazon FSx for Windows File Server. To meet disaster recovery objectives, the company needs to replicate the file server data to another AWS region.

The company is considering these solutions:

A: Set up an Amazon S3 bucket in the DR region. Connect the FSx for Windows File Server in the primary region to this S3 bucket using Amazon FSx File Gateway. Configure this S3 bucket for continuous backup in FSx File Gateway.

B: Create an FSx for Windows File Server in the DR region. Link the VPCs in the primary and DR regions using AWS Site-to-Site VPN. Utilize AWS DataSync for data transfer, operating over VPN endpoints.

C: Establish an FSx for Windows File Server in the DR region. Connect the VPCs in the primary and DR regions via VPC peering. Set up AWS DataSync for data transfer, employing interface VPC endpoints with AWS PrivateLink.

D: Implement an FSx for Windows File Server in the DR region. Connect the VPCs in both regions through AWS Transit Gateway. Employ AWS Transfer Family to transfer files between the FSx file systems in the primary and DR regions over the AWS private network.

The most suitable solution is **C: FSx for Windows File Server in DR Region, VPC Peering, and AWS DataSync with PrivateLink.**

- **C: FSx for Windows File Server in DR Region, VPC Peering, and AWS DataSync with PrivateLink:** This option effectively addresses the requirement for secure and private data replication without using the public internet. Establishing an FSx for Windows File Server in the DR region and connecting the primary and DR region VPCs through VPC peering provides a direct network path. Using AWS DataSync with interface VPC endpoints (AWS PrivateLink) ensures that data replication occurs over the AWS network, adhering to compliance needs and ensuring efficient data transfer.

The reasons why the other options are less suitable:

- **A: FSx File Gateway with S3 Bucket:** FSx File Gateway is not a service offered by AWS. This option incorrectly assumes the existence of such a service and its capability to replicate FSx data to an S3 bucket in another region.
- **B: Site-to-Site VPN with AWS DataSync:** While using AWS DataSync over a Site-to-Site VPN could replicate the data, this method may not be as efficient as VPC peering with PrivateLink. VPN connections might introduce additional latency and complexity compared to VPC peering.
- **D: AWS Transit Gateway with AWS Transfer Family:** Employing AWS Transit Gateway and AWS Transfer Family for file transfers is a more complex solution than necessary. AWS Transfer Family is typically used for transferring files over protocols like SFTP, FTPS, and FTP, and is not the ideal choice for replicating FSx for Windows File Server data.

Question 89

A company is in the planning stages of developing an application that necessitates a Recovery Point Objective (RPO) of less than 5 minutes and a Recovery Time Objective (RTO) of less than 10 minutes. The estimated database size for the application is around 10 TB. An essential aspect of their design involves selecting a database solution that enables rapid failover to a backup system in a secondary AWS Region.

The company is contemplating the following database solutions to fulfill these business needs in a cost-effective manner:

A: Utilize Amazon Aurora to create a DB cluster, capturing snapshots every 5 minutes. After each snapshot is completed, copy it to a secondary region for backup purposes.

B: Implement Amazon RDS with a cross-Region read replica in a secondary region. Should a failure occur, the read replica would be promoted to serve as the primary database.

C: Set up an Amazon Aurora DB cluster in the primary region and another in a secondary region, utilizing AWS Database Migration Service (DMS) to maintain synchronization between the two.

D: Deploy an Amazon RDS instance with a read replica within the same region. In case of a failure, promote the read replica to become the primary database.

The most suitable solution is **B: Amazon RDS with Cross-Region Read Replica.**

- **B: Amazon RDS with Cross-Region Read Replica:** This option effectively meets the RPO and RTO requirements. Having a cross-Region read replica ensures that the secondary database is constantly updated and can be quickly promoted to a primary database in the event of a regional failure. This approach provides a balance between cost and the ability to quickly failover to a secondary region.

The reasons why the other options are less suitable:

- **A: Frequent Aurora Snapshots with Cross-Region Copy:** While snapshotting an Aurora DB cluster at regular intervals ensures data backup, the process of copying snapshots to a secondary region every 5 minutes can be costly due to the volume of data involved (10 TB). Additionally, this approach might not meet the RPO and RTO requirements due to the time it takes to create and transfer snapshots.
- **C: Aurora Clusters in Two Regions with AWS DMS:** Utilizing two Aurora clusters in different regions with synchronization via AWS DMS can be more expensive and complex than necessary. While it would provide data redundancy, the cost and operational overhead of maintaining two clusters and using DMS might not be justifiable.
- **D: RDS with In-Region Read Replica:** This option does not address the requirement of failing over to a secondary region, as it only involves a read replica in the same region. It is suitable for intra-regional high availability but not for inter-regional disaster recovery.

Question 90

A financial institution is in the process of establishing a new AWS account specifically for a digital wallet application. The company currently utilizes AWS Organizations for account management. A solutions architect has created a new member account within AWS Organizations using the IAM user "Support1" from the management account and the email address finance1@example.com.

The task at hand is to determine the correct approach for creating IAM users in this newly established member account. The options being considered are:

A: Log into the AWS Management Console with the root user credentials of the new AWS account, using the 64-character password provided in the initial email sent to finance1@example.com, and then proceed to create the necessary IAM users.

B: Use the management account to assume the 'OrganizationAccountAccessRole' role in the new member account using its account ID, then set up the IAM users as needed.

C: Navigate to the AWS Management Console sign-in page, select "Sign in using root account credentials," log in using the email address finance1@example.com and the root password of the management account, and then create the required IAM users.

D: Visit the AWS Management Console sign-in page, sign in using the account ID of the new member account alongside the credentials of the "Support1" IAM user, and proceed to create the IAM users.

The most appropriate solution is **B: Assume the 'OrganizationAccountAccessRole' Role from the Management Account.**

- **B: Assume the 'OrganizationAccountAccessRole' Role from the Management Account:** This approach is correct and aligns with AWS best practices for managing multiple accounts within AWS Organizations. When a new account is created under AWS Organizations, it automatically has a role named 'OrganizationAccountAccessRole' that grants full administrative access to the account. By assuming this role from the management account, the solutions architect can access the new member account and set up the required IAM users without needing to log in with the root user credentials.

The reasons why the other options are less suitable:

- **A: Use Root User Credentials from Initial Email:** While it's possible to sign in as the root user using the credentials provided in the initial setup email, it's not the recommended approach due to security concerns. AWS best practices suggest minimizing the use of root account credentials and instead using IAM roles for account access.
- **C: Sign in as Root User with Management Account's Root Password:** This option is incorrect because the root password of the management account cannot be used to access the new member account. Each AWS account, including the root account, has its own set of credentials.
- **D: Sign in with 'Support1' IAM User Credentials:** The 'Support1' IAM user from the management account cannot directly access the new member account without assuming a role specifically designed for cross-account access, such as 'OrganizationAccountAccessRole'.

Question 91

A car rental company has developed a serverless REST API for its mobile application, which includes an Amazon API Gateway with a Regional endpoint, AWS Lambda functions, and an Amazon Aurora MySQL Serverless DB cluster. Recently, they expanded API access to their partners' mobile apps, leading to a substantial increase in request volume. This surge in traffic has occasionally triggered database memory errors. Traffic analysis shows that clients frequently make repeated HTTP GET requests for the same data in quick succession, especially during business hours and around certain events like holidays.

The company is now seeking a solution to effectively handle the increased load without significantly raising costs. The following strategies are being considered:

A: Change the API Gateway from a Regional to an edge-optimized endpoint and enable caching on the production stage.

B: Introduce an Amazon ElastiCache for Redis cache to hold results from database queries, and update the Lambda functions to utilize this cache.

C: Adjust the Aurora Serverless DB cluster to increase its maximum memory capacity.

D: Implement throttling on the API Gateway production stage, setting specific rate and burst limits to control incoming call volumes.

The most suitable strategy is A: Switch API Gateway to Edge-Optimized and Enable Caching.

- A: Switch API Gateway to Edge-Optimized and Enable Caching: This solution addresses the issue of repeated requests effectively. By converting to an edge-optimized endpoint, API Gateway can handle requests more efficiently, especially for geographically dispersed clients. Enabling caching reduces the number of direct calls to the backend services (Lambda and Aurora), thus decreasing the load on the database and mitigating memory errors.

The reasons why the other options are less suitable:

- B: Implement Amazon ElastiCache for Redis Cache: While introducing ElastiCache for Redis would help reduce database load by caching query results, it adds complexity to the architecture and operational overhead. Additionally, it may lead to increased costs due to the additional service.
- C: Increase Aurora Serverless Maximum Memory: Increasing the memory of the Aurora Serverless DB cluster could solve the memory error issue but doesn't address the root cause of repeated requests. It could also lead to higher costs without improving overall efficiency.
- D: Enable Throttling on API Gateway: Throttling limits the number of incoming requests but does not reduce the load caused by legitimate repeated requests. It might lead to a degraded user experience, especially during peak times, without addressing the underlying issue of handling repeated queries efficiently.

Question 92

A company is transitioning an application and its associated MySQL database, which processes highly confidential data, from an on-premises setup to AWS. The database, currently 5 TB in size, is continuously updated with new data. For security purposes, the data must be transmitted without passing over the public internet and requires encryption both in transit and at rest. The company has already prepared the database schema on an Amazon RDS for MySQL instance. They have established a 1 Gbps AWS Direct Connect connection, including both a public and a private Virtual Interface (VIF). The goal is to migrate the database to AWS while minimizing downtime.

The company is considering these solutions:

A: Perform a database backup, transfer it to an AWS Snowball Edge Storage Optimized device, and then import the backup to Amazon S3. Utilize server-side encryption with S3-

managed encryption keys (SSE-S3) for at-rest encryption and TLS for in-transit encryption. Finally, import the data from S3 to the RDS instance.

B: Use AWS Database Migration Service (DMS) for the migration. Set up a DMS replication instance in a private subnet and create VPC endpoints for DMS. Configure a DMS task for data copying from the on-premises database to the RDS instance using full load plus change data capture (CDC). Encrypt data at rest using AWS Key Management Service (KMS) default key and ensure in-transit encryption with TLS.

C: Backup the database and use AWS DataSync to transfer the backup files to Amazon S3. Secure data at rest with SSE-S3 and in transit with TLS. Then, import this data from S3 to the RDS instance.

D: Implement Amazon S3 File Gateway and establish a private connection to S3 using AWS PrivateLink. Backup the database and copy the files to S3. Use SSE-S3 for at-rest encryption and TLS for in-transit encryption. Import the data from S3 to the RDS instance.

The most suitable solution is **B: Migrate using AWS Database Migration Service with DMS Replication Instance, VPC Endpoints, and KMS Encryption.**

- **B: Migrate using AWS Database Migration Service with DMS Replication Instance, VPC Endpoints, and KMS Encryption:** This approach leverages AWS DMS, which is specifically designed for efficient database migrations with minimal downtime. Setting up the replication instance in a private subnet and using VPC endpoints ensures that data does not travel over the public internet, addressing the security concern. DMS's ability to perform full load migrations combined with ongoing change data capture (CDC) minimizes downtime and keeps the target database synchronized during the migration process. Encryption at rest is handled through AWS KMS, and TLS secures the data in transit.

The reasons why the other options are less suitable:

- **A: Using AWS Snowball Edge for Backup Import:** While Snowball Edge is an effective way to transfer large amounts of data, it involves physical shipment of the device, potentially leading to longer downtime than a direct migration solution like DMS.
- **C: DataSync for Backup File Transfer to S3:** AWS DataSync can transfer data efficiently to S3, but it does not offer the same real-time data replication capabilities as DMS, which could result in longer downtime during the migration process.
- **D: S3 File Gateway with PrivateLink:** Using S3 File Gateway with PrivateLink for secure transfer to S3 is a viable approach, but like option C, it lacks the real-time replication and minimal downtime offered by AWS DMS.

Question 93

A company is setting up a new big data analytics cluster on AWS, which will operate over numerous Linux Amazon EC2 instances dispersed across various Availability Zones. A critical requirement for this cluster is that all nodes must be able to concurrently read from and write to a shared file storage system. This storage system needs to be highly available, resilient, compatible with the POSIX standard, and capable of handling high throughput.

To meet these requirements, the company is evaluating these storage solutions:

A: Implement AWS Storage Gateway with a file gateway NFS file share linked to an Amazon S3 bucket, and mount this NFS file share on each EC2 instance in the cluster.

B: Set up an Amazon Elastic File System (EFS) with General Purpose performance mode, and mount the EFS file system on each EC2 instance in the cluster.

C: Create a new Amazon Elastic Block Store (EBS) volume using the io2 volume type, and attach this EBS volume to all EC2 instances in the cluster.

D: Establish a new Amazon Elastic File System (EFS) using Max I/O performance mode, and mount the EFS file system on each EC2 instance in the cluster.

The most appropriate solution is **D: Amazon EFS with Max I/O Performance Mode.**

- **D: Amazon EFS with Max I/O Performance Mode:** This option is ideally suited for the company's requirements. EFS is a fully managed service that provides scalable file storage accessible to multiple EC2 instances across different Availability Zones. It is POSIX-compliant, ensuring compatibility with the company's Linux-based environment. EFS with Max I/O performance mode is specifically designed for high-throughput and high-operations scenarios, such as big data and analytics applications, making it the optimal choice for the company's use case.

The reasons why the other options are less suitable:

- **A: AWS Storage Gateway with NFS File Share:** While AWS Storage Gateway provides NFS connectivity, it's not designed for high-performance computing scenarios like big data analytics. Storage Gateway serves different use cases, such as hybrid cloud storage, and may not provide the required throughput for this scenario.
- **B: Amazon EFS with General Purpose Performance Mode:** EFS in General Purpose mode is suitable for a broad range of use cases, but for high-throughput requirements of a big data analytics cluster, Max I/O mode is more appropriate as it's optimized for high levels of I/O operations.
- **C: Amazon EBS with io2 Volume Type:** EBS volumes are block-level storage that can only be attached to a single EC2 instance at a time. This limitation makes EBS unsuitable for scenarios where shared file storage accessible by multiple instances is required.

Question 94

A company offers a SaaS solution on AWS, featuring an HTTPS endpoint through Amazon API Gateway and using AWS Lambda for computing. Data storage is handled by an Amazon Aurora Serverless v1 database. The entire solution, deployed via AWS Serverless Application Model (AWS SAM), operates across multiple Availability Zones. Currently, the solution lacks a disaster recovery (DR) plan. The company aims to establish a DR strategy that allows for recovery in an alternate AWS Region, targeting a Recovery Time Objective (RTO) of 5 minutes and a Recovery Point Objective (RPO) of 1 minute.

The company is considering the following strategies to meet these DR requirements:

A: Set up a read replica of the Aurora Serverless v1 database in the secondary Region. Utilize AWS SAM to develop a runbook for deploying the solution in the target Region. In the event of a disaster, promote the read replica to be the primary database.

B: Convert the Aurora Serverless v1 database to a standard Aurora MySQL global database covering both the primary and secondary Regions. Use AWS SAM to create a runbook for solution deployment in the secondary Region.

C: Deploy an Aurora Serverless v1 DB cluster with multiple writer instances in the secondary Region and launch the solution there. Arrange the solutions in both Regions in an active-passive configuration.

D: Switch the Aurora Serverless v1 database to a standard Aurora MySQL global database spanning the primary and secondary Regions. Deploy the solution in the secondary Region as well, setting up both Regional solutions in an active-passive configuration.

The most appropriate solution is **D: Convert to Aurora MySQL Global Database and Deploy in Both Regions with Active-Passive Configuration.**

- **D: Convert to Aurora MySQL Global Database and Deploy in Both Regions with Active-Passive Configuration:** This option effectively meets the RTO and RPO requirements. By transitioning to a standard Aurora MySQL global database, the company can leverage the database's replication capabilities across regions. Deploying the solution in both the primary and secondary Regions, with an active-passive configuration, ensures quick failover and minimal data loss in the event of a disaster. This setup provides robust DR capabilities while maintaining the required RTO and RPO.

The reasons why the other options are less suitable:

- **A: Read Replica with AWS SAM Runbook:** While using a read replica and AWS SAM runbook is a viable approach, Aurora Serverless v1 does not support cross-region read

replicas, which limits its effectiveness for the desired DR strategy.

- **B: Aurora Global Database with AWS SAM Runbook:** Transitioning to a global database is a good step, but this option does not mention deploying the solution in the secondary Region, which is crucial for meeting the RTO and RPO objectives.
- **C: Multi-Writer Aurora Serverless v1 in Target Region:** Aurora Serverless v1 does not support multiple writer instances, making this option infeasible. Additionally, it does not provide the same level of DR readiness as a global database configuration.

Question 95

A travel agency company operates an AWS Cloud-based application used by its employees to find travel destination information. The content for destinations is refreshed quarterly. The application infrastructure includes two static Amazon EC2 instances, addressed through Amazon Route 53 with a multivalue record pointing to their Elastic IP addresses. Amazon DynamoDB serves as the primary database, while a self-hosted Redis instance is used for caching. During content updates, the system experiences a significant load increase, occasionally resulting in downtime. The company requires a solution to enhance the application's availability and manage the load effectively during content updates.

The company is evaluating these solutions:

A: Implement DynamoDB Accelerator (DAX) for caching. Modify the application to use DAX. Set up an Auto Scaling group for the EC2 instances and an Application Load Balancer (ALB). Assign the Auto Scaling group as the ALB's target. Change the Route 53 record to a simple routing policy aimed at the ALB's DNS alias. Implement scheduled scaling for the EC2 instances ahead of content updates.

B: Transition to Amazon ElastiCache for Redis for caching. Update the application to use ElastiCache. Create an Auto Scaling group for the EC2 instances. Set up an Amazon CloudFront distribution with the Auto Scaling group as its origin. Modify the Route 53 record to a simple routing policy targeting the CloudFront distribution's DNS alias. Manually increase the number of EC2 instances before content updates.

C: Switch to Amazon ElastiCache for Memcached for caching. Adjust the application to use ElastiCache. Establish an Auto Scaling group for the EC2 instances and an Application Load Balancer (ALB). Designate the Auto Scaling group as the target for the ALB. Update the Route 53 record to a simple routing policy targeting the ALB's DNS alias. Arrange for scheduled scaling of the application prior to content updates.

D: Set up DAX for caching. Revise the application to utilize DAX. Develop an Auto Scaling group for the EC2 instances and an Amazon CloudFront distribution, setting the Auto Scaling group as the origin. Modify the Route 53 record to target the CloudFront distribution's DNS

alias with a simple routing policy. Manually scale up the EC2 instances before updating the content.

The most effective solution is **A: Implement DAX, Auto Scaling, ALB, and Scheduled Scaling.**

- **A: Implement DAX, Auto Scaling, ALB, and Scheduled Scaling:** This option optimally addresses the company's needs. Using DynamoDB Accelerator (DAX) provides a high-performance, in-memory cache for DynamoDB, reducing the load on the database during content updates. An Auto Scaling group for EC2 instances ensures that compute resources can scale based on demand. The Application Load Balancer (ALB) helps distribute incoming traffic evenly across the scaled instances. Updating the Route 53 record to point to the ALB allows for efficient traffic management, and scheduled scaling prepares the system for anticipated high loads during content updates.

The reasons why the other options are less suitable:

- **B: ElastiCache for Redis, CloudFront, Manual Scaling:** While ElastiCache for Redis is a valid caching solution, the use of CloudFront and manual scaling of EC2 instances adds complexity and operational overhead. CloudFront is more suited for content delivery rather than load balancing for compute resources.
- **C: ElastiCache for Memcached, ALB, Scheduled Scaling:** Switching to ElastiCache for Memcached could work for caching purposes, but DAX is more directly integrated with DynamoDB, offering better performance for this specific use case.
- **D: DAX, CloudFront, Manual Scaling:** Similar to option B, incorporating CloudFront and manual scaling introduces unnecessary complexity. CloudFront's primary use case as a content delivery network does not align well with the requirements for load balancing application traffic.

Question 96

A company has developed a custom mobile application for uploading image data, which experiences peak usage during weekdays from 8 AM to 5 PM, with thousands of images uploaded every minute. Outside of these hours, the app sees minimal use. Users receive notifications when their image processing is complete. The company is seeking a scalable solution for processing these images efficiently and notifying users upon completion.

The company is considering the following strategies to ensure scalable image processing:

A: Directly upload images from the mobile app to Amazon S3 and configure S3 event notifications to send messages to an Amazon MQ queue.

B: Directly upload images from the mobile app to Amazon S3 and set up S3 event notifications to enqueue messages in an Amazon Simple Queue Service (Amazon SQS) standard queue.

C: Trigger an AWS Lambda function to process the images as soon as messages are available in the queue.

D: Use an S3 Batch Operations job to process images when messages are available in the queue.

E: Utilize Amazon Simple Notification Service (Amazon SNS) to send push notifications to the mobile app once image processing is complete.

F: Use Amazon Simple Email Service (Amazon SES) to send push notifications to the mobile app after processing is finished.

The most effective combination of actions is **B: S3 to SQS for Image Uploads, C: AWS Lambda for Image Processing, E: SNS for User Notifications.**

- **B: S3 to SQS for Image Uploads:** This approach is efficient for handling high-volume uploads. Uploading images directly to Amazon S3 and using S3 event notifications to send messages to an SQS queue allows for decoupling the upload and processing stages. The SQS standard queue is well-suited for managing a high number of messages and ensuring that each image gets processed.
- **C: AWS Lambda for Image Processing:** Triggering an AWS Lambda function from SQS enables scalable, on-demand image processing. Lambda can automatically scale to match the queue size, ensuring efficient processing during peak hours and cost-effectiveness during off-peak times.
- **E: SNS for User Notifications:** Using Amazon SNS to send push notifications to the mobile app is an effective way to notify users once image processing is complete. SNS is designed for high-throughput, push-based messaging, making it suitable for real-time user notifications.

The reasons why the other options are less suitable:

- **A: Amazon MQ for Messaging:** Amazon MQ is typically used for migrating existing message brokers or for applications requiring a traditional message broker. It's more complex and less scalable for this use case compared to SQS.
- **D: S3 Batch Operations for Image Processing:** S3 Batch Operations is more suited for bulk processing of S3 objects and does not provide the real-time, event-driven processing capabilities offered by AWS Lambda.

- F: SES for User Notifications: Amazon SES is an email service and is not designed for push notifications like Amazon SNS. It would be less effective for real-time, app-based user notifications.

Question 97

A company is transitioning its website from an on-premises data center to AWS, intending to adopt a containerized, microservice-based architecture. This move aims to enhance availability and cost-efficiency. The company's security policy mandates adherence to best practices, specifically the principle of least privilege for configuring privileges and network permissions. The application has been deployed to an Amazon Elastic Container Service (ECS) cluster, and the company now needs to finalize the containerized architecture in a way that aligns with these security requirements.

The company is considering the following actions to complete this setup:

A: Configure tasks to use the bridge network mode.

B: Configure tasks to use the awsvpc network mode.

C: Implement security groups for Amazon EC2 instances and utilize IAM roles for EC2 instances to interact with other resources.

D: Attach security groups to the tasks and inject IAM credentials into the containers at launch for accessing other resources.

E: Assign security groups to the tasks and employ IAM roles for tasks to access other AWS resources.

The correct steps to fulfill these requirements are B: Use awsvpc Network Mode for Tasks and E: Apply Security Groups to Tasks and Use IAM Roles for Task Access to Resources.

- B: Use awsvpc Network Mode for Tasks: This network mode allows each ECS task to have its own elastic network interface (ENI), private IP address, and security group, providing a higher level of network isolation and security. This setup aligns with the principle of least privilege by enabling fine-grained network controls at the task level.
- E: Apply Security Groups to Tasks and Use IAM Roles for Tasks: Assigning security groups directly to ECS tasks (in awsvpc network mode) allows for precise control over inbound and outbound traffic at the task level, enhancing security. Using IAM roles for tasks to access AWS resources ensures that each task has only the necessary permissions, adhering to the least privilege principle without embedding credentials into the containers.

The reasons why the other options are less suitable:

- A: Bridge Network Mode for Tasks: The bridge network mode is less secure than awsvpc because it does not provide the same level of network isolation. It is more suitable for simple or legacy applications where fine-grained network control is not a primary concern.
- C: Security Groups for EC2 Instances and IAM Roles for EC2 Instances: While applying security groups to EC2 instances is important, this does not provide the same task-level control as applying them directly to ECS tasks. Additionally, using IAM roles for EC2 instances does not offer the same granularity as IAM roles for individual tasks.
- D: Security Groups for Tasks and Passing IAM Credentials into Containers: Manually passing IAM credentials into containers is not a best practice and can lead to security risks. It is preferable to use IAM roles for tasks, which automatically provide credentials to the containers in a secure manner.

Question 98

A company operates a serverless application utilizing AWS Lambda functions and Amazon DynamoDB. They've recently added new features requiring these Lambda functions to access an Amazon Neptune DB cluster situated within three subnets of a Virtual Private Cloud (VPC). The company needs to configure its infrastructure so that the Lambda functions can simultaneously reach both the Neptune DB cluster and the DynamoDB tables.

To achieve this, the company is considering the following approaches:

A: Set up three public subnets in the VPC housing the Neptune cluster, and connect them to the internet via an internet gateway. Place the Lambda functions within these public subnets.

B: Establish three private subnets in the Neptune VPC and configure internet traffic to pass through a NAT gateway. Position the Lambda functions in these new private subnets.

C: Keep the Lambda functions outside the VPC and adjust the Neptune security group to allow access from the IP ranges used by the Lambda functions.

D: Maintain the Lambda functions outside the VPC. Set up a VPC endpoint for the Neptune database, allowing Lambda functions to access Neptune via this endpoint.

E: Create three private subnets in the Neptune VPC. Deploy the Lambda functions within these new isolated subnets. Establish a VPC endpoint for DynamoDB and route traffic to DynamoDB through this endpoint.

The optimal solutions are B: Private Subnets in Neptune VPC for Lambda with NAT Gateway and E: Isolated Subnets for Lambda in Neptune VPC with DynamoDB VPC Endpoint.

- **B: Private Subnets in Neptune VPC for Lambda with NAT Gateway:** This configuration provides secure and private connectivity for the Lambda functions to access the Neptune DB cluster within the VPC. The NAT gateway enables the Lambda functions to access the internet, which is necessary for them to interact with AWS services like DynamoDB, while maintaining the private nature of their networking environment.
- **E: Isolated Subnets for Lambda in Neptune VPC with DynamoDB VPC Endpoint:** Hosting the Lambda functions in private subnets within the Neptune VPC allows direct access to the Neptune cluster. Setting up a VPC endpoint for DynamoDB ensures that the Lambda functions can securely connect to DynamoDB without the need to traverse the public internet.

The reasons why the other options are less suitable:

- **A: Public Subnets in Neptune VPC for Lambda:** While hosting Lambda functions in public subnets would provide access to Neptune, it's generally not a best practice for security reasons. It exposes the Lambda functions more than necessary, especially when private subnets and VPC endpoints offer a more secure alternative.
- **C: Lambda Outside VPC with Security Group Adjustments:** Hosting Lambda functions outside the VPC and modifying security groups to allow access based on IP ranges can introduce security risks and is less reliable, as Lambda functions may use a range of IPs that can change over time.
- **D: Lambda Outside VPC with Neptune Endpoint:** Currently, AWS does not support VPC endpoints for Amazon Neptune. Therefore, this option is not feasible.

Question 99

A company operates an on-premises application that interacts with two SMB file shares, also located in their data center. The application works with two kinds of files - metadata files and image files - and creates copies of these files. The company is looking to develop a disaster recovery (DR) solution where this duplicated data can be stored on AWS. The solution needs to enable SMB access to the data from either the data center or AWS in case of a disaster. Additionally, although the copied data is infrequently accessed, it must be retrievable within 5 minutes when needed.

The company is considering these options for their DR solution:

A: Implement AWS Outposts with Amazon S3 storage and configure a Windows Amazon EC2 instance on Outposts to function as a file server.

B: Set up an Amazon FSx File Gateway and configure an Amazon FSx for Windows File Server Multi-AZ file system with SSD storage.

C: Deploy an Amazon S3 File Gateway and set it up to store metadata files in Amazon S3 Standard-Infrequent Access (S3 Standard-IA) and image files in S3 Glacier Deep Archive.

D: Deploy an Amazon S3 File Gateway and configure it to store both metadata files and image files in Amazon S3 Standard-Infrequent Access (S3 Standard-IA).

The most suitable solution is **D: Amazon S3 File Gateway with S3 Standard-IA for Metadata and Image Files.**

- **D: Amazon S3 File Gateway with S3 Standard-IA for Metadata and Image Files:** This option effectively meets the company's requirements. The Amazon S3 File Gateway allows the company to maintain SMB access to files, which is compatible with their current setup. By using S3 Standard-Infrequent Access for both metadata and image files, the company ensures that the data is readily available within 5 minutes, as required, while also benefiting from the cost-efficiency of S3 Standard-IA for infrequently accessed data.

The reasons why the other options are less suitable:

- **A: AWS Outposts with EC2 as File Server:** While AWS Outposts extends AWS services to on-premises environments, this approach might be more complex and costly than necessary for the company's DR needs. Additionally, managing an EC2 instance as a file server introduces additional operational overhead.
- **B: FSx File Gateway and FSx for Windows File Server:** Amazon FSx for Windows File Server provides fully managed Microsoft Windows file servers, but this solution is more suited for workloads that require the full features and compatibility of Windows file servers. It might be overkill for the company's DR requirements.
- **C: S3 File Gateway with S3 Standard-IA and Glacier Deep Archive:** Using S3 Glacier Deep Archive for image files does not align with the requirement for data to be available within 5 minutes, as the retrieval times from Glacier Deep Archive are typically much longer.

Question 100

A company is developing a contingency plan to enable 400 employees, who use a combination of Windows and Linux desktops equipped with various software like web browsers and mail clients, to work remotely in case of an unforeseen disaster. The plan necessitates a solution that integrates with the company's existing on-premises Active Directory, allowing employees to utilize their current identity credentials. Additionally, the solution must support multifactor authentication (MFA) and replicate the current desktop user experience.

The company is exploring these options:

A: Implement Amazon WorkSpaces, a managed cloud desktop service. Establish a VPN connection to the on-premises network, set up an AD Connector to link with the on-premises Active Directory, and enable MFA for Amazon WorkSpaces via the AWS Management Console.

B: Adopt Amazon AppStream 2.0 for application streaming, configure Desktop View for employees, and create a VPN connection to the on-premises network. Set up Active Directory Federation Services (AD FS) on-premises and connect the VPC network to AD FS through the VPN.

C: Utilize Amazon WorkSpaces for the cloud desktop service. Create a VPN connection to the on-premises network, establish an AD Connector for integration with the on-premises Active Directory, and set up a RADIUS server for MFA.

D: Choose Amazon AppStream 2.0 as the application streaming service. Set up Active Directory Federation Services on-premises and configure MFA for user access on AppStream 2.0.

The most suitable solution is **C: Amazon WorkSpaces with AD Connector and RADIUS Server for MFA.**

- **C: Amazon WorkSpaces with AD Connector and RADIUS Server for MFA:** This approach effectively meets all the company's requirements. Amazon WorkSpaces provides a full cloud-based desktop experience for both Windows and Linux, closely replicating the existing desktop environment. Integration with on-premises Active Directory via AD Connector maintains current user credentials. The use of a RADIUS server for MFA ensures an added layer of security, complying with the company's multifactor authentication requirement.

The reasons why the other options are less suitable:

- **A: MFA for WorkSpaces via AWS Management Console:** While Amazon WorkSpaces is a fitting solution, enabling MFA through the AWS Management Console does not offer the same level of integration with the company's existing systems as a RADIUS server setup would.
- **B: Amazon AppStream 2.0 with AD FS and VPN:** AppStream 2.0 is more focused on application streaming rather than providing a full desktop experience. While it could be used for remote work, it might not replicate the user experience of the existing desktops as closely as WorkSpaces.
- **D: AppStream 2.0 with AD FS and MFA:** Similar to option B, AppStream 2.0 primarily streams applications and might not fully match the desktop experience provided by

WorkSpaces. Moreover, setting up MFA directly for AppStream 2.0 access might not integrate as seamlessly with the company's on-premises Active Directory.