Amazon

(AWS Certified Solutions Architect - Professional SAP-C02)

AWS Certified Solutions Architect - Professional SAP-C02

Total: **529 Questions**

Question: 1

A company needs to architect a hybrid DNS solution. This solution will use an Amazon Route 53 private hosted zone for the domain cloud.example.com for the resources stored within VPCs.
The company has the following DNS resolution requirements:
On-premises systems should be able to resolve and connect to cloud.example.com.
All VPCs should be able to resolve cloud.example.com.
There is already an AWS Direct Connect connection between the on-premises corporate network and AWS Transit Gateway.
Which architecture should the company use to meet these requirements with the HIGHEST performance?

A.Associate the private hosted zone to all the VPCs. Create a Route 53 inbound resolver in the shared services VPC. Attach all VPCs to the transit gateway and create forwarding rules in the on-premises DNS server for cloud.example.com that point to the inbound resolver.

B.Associate the private hosted zone to all the VPCs. Deploy an Amazon EC2 conditional forwarder in the shared services VPC. Attach all VPCs to the transit gateway and create forwarding rules in the on-premises DNS server for cloud.example.com that point to the conditional forwarder.

C.Associate the private hosted zone to the shared services VPCreate a Route 53 outbound resolver in the shared services VPAttach all VPCs to the transit gateway and create forwarding rules in the on-premises DNS server for cloud.example.com that point to the outbound resolver.

D.Associate the private hosted zone to the shared services VPC. Create a Route 53 inbound resolver in the shared services VPC. Attach the shared services VPC to the transit gateway and create forwarding rules in the on-premises DNS server for cloud.example.com that point to the inbound resolver.

Answer: A

Explanation:

The correct answer is A because it provides a highly performant and scalable hybrid DNS solution. Here's a detailed justification:

Private Hosted Zone Association: Associating the private hosted zone with all VPCs ensures that resources within each VPC can directly resolve the cloud.example.com domain names using Route 53's DNS servers. This is the fundamental requirement for internal DNS resolution within AWS.

Inbound Resolver for On-Premises Access: The Route 53 inbound resolver, placed in a shared services VPC, provides a dedicated endpoint for on-premises systems to forward DNS queries for cloud.example.com. Inbound resolvers allow on-premises networks to query the Route 53 private hosted zone in AWS.

Transit Gateway Integration: Attaching all VPCs to the Transit Gateway establishes connectivity between them and the on-premises network via the existing Direct Connect. This creates a consistent network path for DNS queries.

On-Premises DNS Forwarding: Creating forwarding rules on the on-premises DNS server that point to the inbound resolver ensures that any requests for cloud.example.com are directed to AWS for resolution within the private hosted zone.

Performance Advantages: Option A leverages Route 53's managed infrastructure, which is highly scalable and resilient, providing better performance compared to deploying and managing EC2-based conditional forwarders (Option B). It also doesn't limit the scope of association only to the shared service VPC (Options C and D).

Correctness of Options C & D: Options C and D limit the private hosted zone association only to the shared service VPC making it impossible for other VPC's resources to use DNS to resolve resources within AWS.

Authoritative Links:

Route 53 Private Hosted Zones:https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/private-hosted-zones.html

Route 53 Resolver:https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resolver.html **AWS Transit Gateway:**https://aws.amazon.com/transit-gateway/

Question: 2

A company is providing weather data over a REST-based API to several customers. The API is hosted by Amazon API Gateway and is integrated with different AWS Lambda functions for each API operation. The company uses Amazon Route 53 for DNS and has created a resource record of weather.example.com. The company stores data for the API in Amazon DynamoDB tables. The company needs a solution that will give the API the ability to fail over to a different AWS Region. Which solution will meet these requirements?

A.Deploy a new set of Lambda functions in a new Region. Update the API Gateway API to use an edge-optimized API endpoint with Lambda functions from both Regions as targets. Convert the DynamoDB tables to global tables.

B.Deploy a new API Gateway API and Lambda functions in another Region. Change the Route 53 DNS record to a multivalue answer. Add both API Gateway APIs to the answer. Enable target health monitoring. Convert the DynamoDB tables to global tables.

C.Deploy a new API Gateway API and Lambda functions in another Region. Change the Route 53 DNS record to a failover record. Enable target health monitoring. Convert the DynamoDB tables to global tables.

D.Deploy a new API Gateway API in a new Region. Change the Lambda functions to global functions. Change the Route 53 DNS record to a multivalue answer. Add both API Gateway APIs to the answer. Enable target health monitoring. Convert the DynamoDB tables to global tables.

Answer: C

Explanation:

Let's analyze why option C is the correct solution for creating a cross-region failover strategy for the weather data API.

The primary goal is to ensure the API remains available even if the primary AWS Region experiences an outage. This requires replicating the API functionality and data in another Region and configuring DNS to automatically switch to the secondary Region in case of a failure.

Option C leverages several key AWS features to achieve this. First, it replicates the entire API infrastructure by deploying a new API Gateway API and Lambda functions in a second Region. This ensures that a fully functional backup API is ready and waiting.

Crucially, it uses a Route 53 failover record. A failover record in Route 53 allows you to define a primary and a secondary record. Route 53 health checks monitor the primary endpoint (the API Gateway in the primary region). If the health check fails, Route 53 automatically begins routing traffic to the secondary endpoint (the API Gateway in the secondary region). This provides automatic and fast failover.
[https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/dns-failover.html]

Target health monitoring is enabled to ensure that the primary API Gateway endpoint is actively monitored for its health status. This is vital for timely failover. If the primary API Gateway or its underlying resources become unavailable, the health check will fail, triggering the failover.

Finally, the solution converts the DynamoDB tables to global tables. DynamoDB Global Tables provide multi-region, active-active database replication, enabling low-latency access to data from anywhere in the world and providing resilience in the face of regional outages.
[https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/GlobalTables.html] This ensures data consistency and availability across both Regions.

Now, let's examine why the other options are less suitable.

Option A uses an edge-optimized API endpoint and attempts to target Lambda functions in both Regions from the same API Gateway. While edge-optimized API Gateways offer lower latency, they are not designed for regional failover. Moreover, targeting Lambda functions from different regions from a single API Gateway can introduce significant latency and complexity. Edge-optimized API Gateways are about improving latency for clients; not about multi-region failover.

Option B employs a multivalue answer in Route 53. While multivalue answers can distribute traffic across multiple endpoints, they are not designed for failover. Route 53 will return multiple IP addresses, and clients will attempt to connect to them randomly. It doesn't guarantee that the healthy endpoint will be used during an outage.

Option D suggests using "global functions," which do not exist directly as a feature in AWS Lambda. Lambda functions are region-specific. Although Lambda functions can be invoked across Regions, it requires additional configuration and is not a built-in feature called a "global function." Similar to option B, using a multivalue answer also does not guarantee failover.

In summary, only option C provides a comprehensive solution that includes replicating the API and Lambda functions, configuring Route 53 for automatic failover with health checks, and ensuring data availability and consistency through DynamoDB Global Tables.

Question: 3

A company uses AWS Organizations with a single OU named Production to manage multiple accounts. All accounts are members of the Production OU. Administrators use deny list SCPs in the root of the organization to manage access to restricted services.
The company recently acquired a new business unit and invited the new unit's existing AWS account to the organization. Once onboarded, the administrators of the new business unit discovered that they are not able to update existing AWS Config rules to meet the company's policies.
Which option will allow administrators to make changes and continue to enforce the current policies without introducing additional long-term maintenance?

A.Remove the organization's root SCPs that limit access to AWS Config. Create AWS Service Catalog products for the company's standard AWS Config rules and deploy them throughout the organization, including the new account.

B.Create a temporary OU named Onboarding for the new account. Apply an SCP to the Onboarding OU to allow AWS Config actions. Move the new account to the Production OU when adjustments to AWS Config are complete.

C.Convert the organization's root SCPs from deny list SCPs to allow list SCPs to allow the required services only. Temporarily apply an SCP to the organization's root that allows AWS Config actions for principals only in the new account.

D.Create a temporary OU named Onboarding for the new account. Apply an SCP to the Onboarding OU to allow AWS Config actions. Move the organization's root SCP to the Production OU. Move the new account to the Production OU when adjustments to AWS Config are complete.

Answer: D

Explanation:

The correct answer is D. Here's a detailed justification:

The problem stems from deny list SCPs at the root of the organization preventing modifications to AWS Config rules in the new account. We need a solution that temporarily grants the necessary permissions to the new account's administrators to update the rules while still maintaining the organization's overall security posture.

Option D provides the best temporary and minimally disruptive solution. It creates an "Onboarding" OU and

places the new account there. A more permissive SCP allowing Config actions is applied to the "Onboarding" OU, effectively overriding the deny list SCP at the root only for accounts in that OU. The new account administrators can then make the necessary Config rule changes. Crucially, it also involves moving the root SCP to the Production OU. This action restricts the SCP's effect only to the Production OU, preventing it from impacting the Onboarding OU. Once the modifications are complete, the new account is moved to the Production OU, inheriting the standard restrictions. The temporary OU and SCP are then no longer needed and can be removed or kept for future onboarding.

Option A is undesirable because removing the root SCPs entirely would weaken the organization's security posture for all accounts, not just the new one. While Service Catalog could be a long-term solution, it doesn't immediately address the immediate need to modify existing rules in the new account.

Option B provides a similar temporary OU approach but doesn't address the placement of the original SCP. The root SCP would still impact accounts placed in the root, creating further issues and complications for other accounts managed in the root.

Option C converting to allow list SCPs is a significant undertaking with potentially wide-ranging impacts and isn't a suitable temporary solution. Temporarily allowing Config actions only for principals in the new account via SCP at root level is difficult to configure correctly and could unintentionally grant wider permissions than intended. It doesn't isolate the impact of the relaxed permissions, and maintains all other accounts still subject to the restrictive root SCP policies.

In summary, Option D provides a scoped, temporary, and reversible solution that balances the need to grant necessary permissions with the organization's security requirements.

Relevant AWS Documentation:

Service Control Policies (SCPs)
AWS Organizations

Question: 4

A company is running a two-tier web-based application in an on-premises data center. The application layer consists of a single server running a stateful application. The application connects to a PostgreSQL database running on a separate server. The application's user base is expected to grow significantly, so the company is migrating the application and database to AWS. The solution will use Amazon Aurora PostgreSQL, Amazon EC2 Auto Scaling, and Elastic Load Balancing.

Which solution will provide a consistent user experience that will allow the application and database tiers to scale?

A.Enable Aurora Auto Scaling for Aurora Replicas. Use a Network Load Balancer with the least outstanding requests routing algorithm and sticky sessions enabled.

B.Enable Aurora Auto Scaling for Aurora writers. Use an Application Load Balancer with the round robin routing algorithm and sticky sessions enabled.

C.Enable Aurora Auto Scaling for Aurora Replicas. Use an Application Load Balancer with the round robin routing and sticky sessions enabled.

D.Enable Aurora Scaling for Aurora writers. Use a Network Load Balancer with the least outstanding requests routing algorithm and sticky sessions enabled.

Answer: C

Explanation:

The correct answer is C. Here's a detailed justification:

The scenario requires scaling both the application and database tiers while maintaining a consistent user experience for a stateful application migrating to AWS.

Aurora Auto Scaling for Aurora Replicas: Aurora Auto Scaling dynamically adjusts the number of Aurora Replicas in response to changes in application demand. Aurora Replicas handle read traffic, offloading the writer node and improving read performance, which is crucial for scaling the database tier without impacting write operations or application availability.
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Managing.Scaling.html

Application Load Balancer (ALB) with Round Robin and Sticky Sessions: An ALB is best suited for routing HTTP/HTTPS traffic, which is typical for web applications. Using the round-robin routing algorithm distributes requests evenly across multiple EC2 instances within the Auto Scaling group. Sticky sessions (also known as session affinity) are critical for maintaining stateful application data. They ensure that all requests from a given user are consistently routed to the same EC2 instance, preserving the user's session data and ensuring a consistent experience.
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html

Why other options are incorrect:

Option A & D: Network Load Balancer (NLB): NLBs operate at Layer 4 (TCP/UDP), and while they provide high throughput and low latency, they are not ideal for HTTP/HTTPS traffic and do not inherently support sticky sessions based on application-layer cookies (necessary for stateful applications). NLBs support source IP stickiness, but this is generally less reliable than cookie-based stickiness.

Option B & D: Aurora Auto Scaling for Aurora Writers: While scaling the writer instance is possible, it's generally more disruptive. Scaling replicas handles read scaling far more effectively and is much less prone to causing issues. You scale the writers typically only when there are writer performance issues, not general scaling of the application load.

Option A & B: Least Outstanding Requests routing algorithm: While this algorithm can be beneficial in some scenarios, it's often less predictable than round robin in a well-managed Auto Scaling group. Also, for a stateful application, stickiness is far more important than balancing on number of requests.

In summary, Answer C offers the most comprehensive solution for scaling both application and database tiers of a stateful web application using Aurora PostgreSQL and EC2 Auto Scaling, while ensuring a consistent user experience.

Question: 5

A company uses a service to collect metadata from applications that the company hosts on premises. Consumer devices such as TVs and internet radios access the applications. Many older devices do not support certain HTTP headers and exhibit errors when these headers are present in responses. The company has configured an on-premises load balancer to remove the unsupported headers from responses sent to older devices, which the company identified by the User-Agent headers.
The company wants to migrate the service to AWS, adopt serverless technologies, and retain the ability to support the older devices. The company has already migrated the applications into a set of AWS Lambda functions. Which solution will meet these requirements?

A.Create an Amazon CloudFront distribution for the metadata service. Create an Application Load Balancer (ALB). Configure the CloudFront distribution to forward requests to the ALB. Configure the ALB to invoke the correct Lambda function for each type of request. Create a CloudFront function to remove the problematic headers based on the value of the User-Agent header.

B.Create an Amazon API Gateway REST API for the metadata service. Configure API Gateway to invoke the correct Lambda function for each type of request. Modify the default gateway responses to remove the problematic headers based on the value of the User-Agent header.

C.Create an Amazon API Gateway HTTP API for the metadata service. Configure API Gateway to invoke the correct Lambda function for each type of request. Create a response mapping template to remove the problematic headers based on the value of the User-Agent. Associate the response data mapping with the HTTP API.

D.Create an Amazon CloudFront distribution for the metadata service. Create an Application Load Balancer (ALB). Configure the CloudFront distribution to forward requests to the ALB. Configure the ALB to invoke the

correct Lambda function for each type of request. Create a function that will remove the problematic headers in response to viewer requests based on the value of the User-Agent header.

Answer: A

Explanation:

The correct answer is **A**. Here's why:

The primary requirement is to remove problematic HTTP headers from responses based on the User-Agent header to support older devices while migrating to serverless technologies in AWS.

Option A (CloudFront + ALB + Lambda + CloudFront Function): This solution perfectly aligns with the requirements. CloudFront, a CDN, is used for caching and edge processing. A CloudFront function allows you to run lightweight code at the edge, closest to the user. In this case, a CloudFront function is created to inspect the User-Agent header of incoming requests and selectively strip out problematic headers before the response is sent to the client. The ALB acts as a load balancer in front of the Lambda functions and distributes traffic. It integrates well with Lambda and provides necessary features such as routing. This approach is scalable, efficient, and avoids burdening the Lambda functions with header manipulation.

Option B (API Gateway REST API + Lambda + Gateway Response Modification): While API Gateway can invoke Lambda functions and modify responses, altering the default gateway responses based on the User-Agent is not a standard or recommended feature. This option would involve more complex configurations and could potentially affect all responses, not just those for older devices. API Gateway's transformation capabilities are less flexible than CloudFront functions for this specific header manipulation scenario.

Option C (API Gateway HTTP API + Lambda + Response Mapping Template): HTTP APIs are lightweight but typically offer less functionality than REST APIs. While response mapping templates can transform data, the ability to conditionally remove headers based on the User-Agent is more cumbersome and less efficient than using CloudFront Functions.

Option D (CloudFront + ALB + Lambda + function): This option is similar to option A, but it incorrectly uses Lambda@Edge. Lambda@Edge functions are designed for requests sent to CloudFront. The function needs to manipulate the response headers as they are returned to the consumer, a viewer response event. Also, Lambda@Edge functions have more stringent limitations on execution time and regions than CloudFront Functions. CloudFront Functions are designed specifically for lightweight modifications like header manipulation.

Justification:

1. **Edge Processing:** CloudFront allows for processing requests and responses at the edge, reducing latency and offloading processing from the origin (Lambda functions). This is beneficial for optimizing the user experience for older devices.
2. **Header Manipulation with CloudFront Functions:** CloudFront Functions provide a mechanism to execute JavaScript code directly in CloudFront locations. They're specifically designed for lightweight tasks such as modifying HTTP headers based on request attributes (like User-Agent). This enables precise targeting of older devices without affecting newer ones.
3. **Scalability and Serverless:** The solution leverages serverless technologies (Lambda, CloudFront) for automatic scaling and cost efficiency.
4. **ALB as a Router:** The ALB provides load balancing and routing to the appropriate Lambda functions based on the request. This allows the functions themselves to remain focused on the core application logic.
5. **Minimal Impact on Lambda Functions:** By handling the header removal in CloudFront, the Lambda functions don't need to be modified, reducing complexity and potential for errors.

Authoritative Links:

CloudFront Functions:https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/cloudfront-functions.html
Amazon CloudFront:https://aws.amazon.com/cloudfront/
Application Load Balancer:https://aws.amazon.com/elasticloadbalancing/application-load-balancer/
AWS Lambda:https://aws.amazon.com/lambda/

Question: 6

A retail company needs to provide a series of data files to another company, which is its business partner. These files are saved in an Amazon S3 bucket under Account A, which belongs to the retail company. The business partner company wants one of its IAM users, User_DataProcessor, to access the files from its own AWS account (Account B).
Which combination of steps must the companies take so that User_DataProcessor can access the S3 bucket successfully? (Choose two.)

A.Turn on the cross-origin resource sharing (CORS) feature for the S3 bucket in Account A.

B.In Account A, set the S3 bucket policy to the following:

```
{
    "Effect": "Allow",
    "Action": [
        "s3:GetObject",
        "s3:ListBucket"
    ],
    "Resource": "arn:aws:s3:::AccountABucketName/*"
}
```

C.In Account A, set the S3 bucket policy to the following:

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::AccountB:user/User_DataProcessor"
    },
    "Action": [
        "s3:GetObject",
        "s3:ListBucket"
    ],
    "Resource": [
        "arn:aws:s3:::AccountABucketName/*"
    ]
}
```

D.In Account B, set the permissions of User_DataProcessor to the following:

```
{
    "Effect": "Allow",
    "Action": [
        "s3:GetObject",
        "s3:ListBucket"
    ],
    "Resource": "arn:aws:s3:::AccountABucketName/*"
}
```

E.In Account B, set the permissions of User_DataProcessor to the following:

```
{
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::AccountB:user/User_DataProcessor"
    },
    "Action": [
        "s3:GetObject",
        "s3:ListBucket"
    ],
    "Resource": [
        "arn:aws:s3:::AccountABucketName/*"
    ]
}
```

Answer: C

Explanation:

Doesn't make sense for account B to control access to resources in account A. So D is NOT the answer.Account A owns the bucket and sets the bucket policy to allow access to a principal/user in Account B

C & D.But the ListBucket action won't work as you need to mention the arn of the bucket itself as well (without the /*)

Question: 7

A company is running a traditional web application on Amazon EC2 instances. The company needs to refactor the application as microservices that run on containers. Separate versions of the application exist in two distinct environments: production and testing. Load for the application is variable, but the minimum load and the maximum load are known. A solutions architect needs to design the updated application with a serverless architecture that minimizes operational complexity.
Which solution will meet these requirements MOST cost-effectively?

A.Upload the container images to AWS Lambda as functions. Configure a concurrency limit for the associated Lambda functions to handle the expected peak load. Configure two separate Lambda integrations within Amazon API Gateway: one for production and one for testing.

B.Upload the container images to Amazon Elastic Container Registry (Amazon ECR). Configure two auto scaled Amazon Elastic Container Service (Amazon ECS) clusters with the Fargate launch type to handle the expected load. Deploy tasks from the ECR images. Configure two separate Application Load Balancers to direct traffic to the ECS clusters.

C.Upload the container images to Amazon Elastic Container Registry (Amazon ECR). Configure two auto scaled

Amazon Elastic Kubernetes Service (Amazon EKS) clusters with the Fargate launch type to handle the expected load. Deploy tasks from the ECR images. Configure two separate Application Load Balancers to direct traffic to the EKS clusters.

D.Upload the container images to AWS Elastic Beanstalk. In Elastic Beanstalk, create separate environments and deployments for production and testing. Configure two separate Application Load Balancers to direct traffic to the Elastic Beanstalk deployments.

Answer: B

Explanation:

Option B is the most cost-effective and operationally simple serverless solution for containerized microservices with variable load, compared to the other options. Here's why:

Cost-Effectiveness: ECS Fargate provides a pay-as-you-go pricing model where you only pay for the compute and memory resources your containers consume. This aligns well with variable loads, ensuring minimal cost when the application is not heavily used. Lambda (Option A) while serverless, may become complex and expensive when dealing with entire containerized applications due to its invocation-based pricing and size limitations, which are not suitable for refactoring large traditional web applications. Option C is also not the most cost-effective, as EKS has more overhead and operational complexity compared to ECS Fargate. Elastic Beanstalk (Option D) abstracts away some infrastructure management but is not as cost-effective as Fargate for handling variable loads in the context of containerized microservices, due to EC2 instance usage even when the application is idle.

Operational Simplicity: ECS Fargate abstracts away the underlying infrastructure management (EC2 instances, cluster management), reducing operational overhead. The automatic scaling of ECS services easily manages the varying loads in production and testing environments. Two separate Application Load Balancers allow for independent routing and management of traffic to the respective environments. Lambda may introduce complexity if the application needs more than simple event-driven tasks. EKS (Option C) introduces additional operational complexity compared to ECS, as Kubernetes requires expertise and more configuration.

Elastic Beanstalk (Option D) offers managed services, but it is still less serverless than Fargate because you need to choose an EC2 instance type based on your maximum expected load and manually configure scaling policies.

Suitable for Microservices: ECS Fargate is designed for running containerized microservices. Option B allows easy scaling and management of microservices, making it suited to the requirements of a traditional web application being refactored into microservices.

Serverless: Both Lambda (Option A) and ECS Fargate (Option B and C) allow for serverless container deployments, but ECS Fargate is more appropriate when dealing with the entire application in containers.

Distinct Environments: Configuring two separate clusters in ECS (Option B) for the production and testing environments will provide isolation and the ability to run separate versions of the application without risk of impacting one another.

Therefore, the best option is B as it balances cost-effectiveness, operational simplicity, and the microservices architecture.

Here are some resources for further research:

AWS ECS Fargate: https://aws.amazon.com/fargate/
AWS Lambda: https://aws.amazon.com/lambda/
AWS EKS: https://aws.amazon.com/eks/
AWS Elastic Beanstalk: https://aws.amazon.com/elasticbeanstalk/

Question: 8

A company has a multi-tier web application that runs on a fleet of Amazon EC2 instances behind an Application Load Balancer (ALB). The instances are in an Auto Scaling group. The ALB and the Auto Scaling group are replicated in a backup AWS Region. The minimum value and the maximum value for the Auto Scaling group are set to zero. An Amazon RDS Multi-AZ DB instance stores the application's data. The DB instance has a read replica in the backup Region. The application presents an endpoint to end users by using an Amazon Route 53 record. The company needs to reduce its RTO to less than 15 minutes by giving the application the ability to automatically fail over to the backup Region. The company does not have a large enough budget for an active-active strategy. What should a solutions architect recommend to meet these requirements?

A.Reconfigure the application's Route 53 record with a latency-based routing policy that load balances traffic between the two ALBs. Create an AWS Lambda function in the backup Region to promote the read replica and modify the Auto Scaling group values. Create an Amazon CloudWatch alarm that is based on the HTTPCode_Target_5XX_Count metric for the ALB in the primary Region. Configure the CloudWatch alarm to invoke the Lambda function.

B.Create an AWS Lambda function in the backup Region to promote the read replica and modify the Auto Scaling group values. Configure Route 53 with a health check that monitors the web application and sends an Amazon Simple Notification Service (Amazon SNS) notification to the Lambda function when the health check status is unhealthy. Update the application's Route 53 record with a failover policy that routes traffic to the ALB in the backup Region when a health check failure occurs.

C.Configure the Auto Scaling group in the backup Region to have the same values as the Auto Scaling group in the primary Region. Reconfigure the application's Route 53 record with a latency-based routing policy that load balances traffic between the two ALBs. Remove the read replica. Replace the read replica with a standalone RDS DB instance. Configure Cross-Region Replication between the RDS DB instances by using snapshots and Amazon S3.

D.Configure an endpoint in AWS Global Accelerator with the two ALBs as equal weighted targets. Create an AWS Lambda function in the backup Region to promote the read replica and modify the Auto Scaling group values. Create an Amazon CloudWatch alarm that is based on the HTTPCode_Target_5XX_Count metric for the ALB in the primary Region. Configure the CloudWatch alarm to invoke the Lambda function.

Answer: B

Explanation:

The correct answer is B because it provides a cost-effective and automated failover solution that meets the RTO requirement.

Here's a breakdown of why option B is the best choice:

Route 53 Failover Policy: Using a failover policy in Route 53 ensures that traffic is automatically routed to the backup region when the primary region is deemed unhealthy. This aligns directly with the requirement for automated failover. https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/dns-failover.html

Health Checks: Route 53 health checks monitor the application's availability. This provides a robust and accurate way to detect failures in the primary region, triggering the failover process. https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/health-checks-creating-deleting.html

Lambda Function: A Lambda function in the backup region automates the promotion of the read replica to a standalone database instance and adjusts the Auto Scaling group's configuration to start the application servers. This minimizes manual intervention and reduces the overall failover time. https://aws.amazon.com/lambda/

SNS Notification: Amazon SNS acts as a reliable mechanism to notify the Lambda function when the health check fails. This ensures that the failover process is initiated promptly. https://aws.amazon.com/sns/

Why other options are less suitable:

Option A: Latency-based routing doesn't provide automatic failover. Traffic might still be directed to the unhealthy primary region. Triggering failover solely based on HTTP 5XX errors might be too reactive and miss

other types of failures.

Option C: Latency-based routing doesn't provide automatic failover. While Cross-Region Replication can work, setting it up using S3 snapshots and manual restoration would exceed the RTO of 15 minutes. Also,
configuring the Auto Scaling group in the backup Region to have the same values as in the primary Region defeats the "minimum cost" requirement.

Option D: AWS Global Accelerator is suitable for improving global application performance, but in this specific case, it is more expensive than the other options, and the question specifies cost constraints. Similarly to option A, triggering failover solely based on HTTP 5XX errors might be too reactive and miss other types of failures.

Question: 9

A company is hosting a critical application on a single Amazon EC2 instance. The application uses an Amazon ElastiCache for Redis single-node cluster for an in-memory data store. The application uses an Amazon RDS for MariaDB DB instance for a relational database. For the application to function, each piece of the infrastructure must be healthy and must be in an active state.
A solutions architect needs to improve the application's architecture so that the infrastructure can automatically recover from failure with the least possible downtime.
Which combination of steps will meet these requirements? (Choose three.)

A.Use an Elastic Load Balancer to distribute traffic across multiple EC2 instances. Ensure that the EC2 instances are part of an Auto Scaling group that has a minimum capacity of two instances.

B.Use an Elastic Load Balancer to distribute traffic across multiple EC2 instances. Ensure that the EC2 instances are configured in unlimited mode.

C.Modify the DB instance to create a read replica in the same Availability Zone. Promote the read replica to be the primary DB instance in failure scenarios.

D.Modify the DB instance to create a Multi-AZ deployment that extends across two Availability Zones.

E.Create a replication group for the ElastiCache for Redis cluster. Configure the cluster to use an Auto Scaling group that has a minimum capacity of two instances.

F.Create a replication group for the ElastiCache for Redis cluster. Enable Multi-AZ on the cluster.

Answer: ADF

Explanation:

Here's a detailed justification for the answer ADF:

A. Use an Elastic Load Balancer to distribute traffic across multiple EC2 instances. Ensure that the EC2 instances are part of an Auto Scaling group that has a minimum capacity of two instances.

Justification: This ensures high availability for the EC2 instances hosting the application. The Elastic Load Balancer (ELB) distributes traffic, preventing a single point of failure. The Auto Scaling group automatically replaces unhealthy EC2 instances, minimizing downtime. Keeping a minimum of two instances ensures redundancy.
Supporting Concepts: Load balancing, Auto Scaling, redundancy, high availability.
Authoritative Link:https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-benefits.html

D. Modify the DB instance to create a Multi-AZ deployment that extends across two Availability Zones.

Justification: Amazon RDS Multi-AZ deployments provide high availability and failover capabilities. If the primary DB instance fails, RDS automatically fails over to the standby instance in another Availability Zone, minimizing downtime. This ensures the database layer is resilient to failures.
Supporting Concepts: Database replication, failover, Availability Zones, high availability.

Authoritative Link:https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.MultiAZ.html

F. Create a replication group for the ElastiCache for Redis cluster. Enable Multi-AZ on the cluster.

Justification: Enabling Multi-AZ with ElastiCache for Redis creates a read replica in a different Availability Zone. In case of a failure in the primary node, ElastiCache automatically promotes the replica to be the primary node. This ensures minimal downtime and data loss.
Supporting Concepts: Redis replication, failover, Availability Zones, high availability.
Authoritative Link:https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/AutoFailover.html

Why other options are not optimal:

B: Configuring EC2 instances in unlimited mode affects CPU credit usage but doesn't directly address instance failure recovery.
C: While creating a read replica is good, manually promoting it during a failure is not automated and would lead to increased downtime compared to Multi-AZ.
E: Auto Scaling Group does not apply to ElastiCache Redis in this context. ElastiCache has its own mechanism (replication groups and Multi-AZ) for HA and failover.

Question: 10

A retail company is operating its ecommerce application on AWS. The application runs on Amazon EC2 instances behind an Application Load Balancer (ALB). The company uses an Amazon RDS DB instance as the database backend. Amazon CloudFront is configured with one origin that points to the ALB. Static content is cached. Amazon Route 53 is used to host all public zones.
After an update of the application, the ALB occasionally returns a 502 status code (Bad Gateway) error. The root cause is malformed HTTP headers that are returned to the ALB. The webpage returns successfully when a solutions architect reloads the webpage immediately after the error occurs.
While the company is working on the problem, the solutions architect needs to provide a custom error page instead of the standard ALB error page to visitors.
Which combination of steps will meet this requirement with the LEAST amount of operational overhead? (Choose two.)

A.Create an Amazon S3 bucket. Configure the S3 bucket to host a static webpage. Upload the custom error pages to Amazon S3.

B.Create an Amazon CloudWatch alarm to invoke an AWS Lambda function if the ALB health check response Target.FailedHealthChecks is greater than 0. Configure the Lambda function to modify the forwarding rule at the ALB to point to a publicly accessible web server.

C.Modify the existing Amazon Route 53 records by adding health checks. Configure a fallback target if the health check fails. Modify DNS records to point to a publicly accessible webpage.

D.Create an Amazon CloudWatch alarm to invoke an AWS Lambda function if the ALB health check response Elb.InternalError is greater than 0. Configure the Lambda function to modify the forwarding rule at the ALB to point to a public accessible web server.

E.Add a custom error response by configuring a CloudFront custom error page. Modify DNS records to point to a publicly accessible web page.

Answer: AE

Explanation:

The correct answer is AE. Here's why:

A. Create an Amazon S3 bucket... Upload the custom error pages to Amazon S3. This is necessary because you need a place to host the custom error page. Amazon S3 is ideal for serving static content like HTML error pages with high availability and scalability. S3 can be configured to serve static websites directly.

E. Add a custom error response by configuring a CloudFront custom error page. CloudFront is already being

used as a CDN in front of the ALB. CloudFront custom error pages allow you to intercept HTTP error codes (like 502) returned by the origin (ALB) and serve a custom page instead. This is the simplest and most direct way to replace the ALB's default error page without significantly altering the existing architecture or introducing complex routing logic. CloudFront is configured to deliver your website to the end users. By setting up custom error pages, you modify the content delivered by CloudFront when an origin error, such as a 502, occurs.

Why other options are not the best choices:

B & D: While a CloudWatch alarm triggering a Lambda function to modify the ALB routing could work, it is significantly more complex and adds operational overhead. Changing ALB rules dynamically based on health checks introduces potential race conditions and complexities in maintaining the application. Lambda is designed to execute code when triggered, but is not the most appropriate tool for this situation.

C: Modifying Route 53 records to point to a different web server is a global DNS change. This is a far more disruptive solution than simply configuring CloudFront's error pages. Additionally, DNS propagation times mean that not all users would see the custom error page immediately. DNS management is complex, so it's better to avoid changing it.

In summary: Hosting the error page on S3 and configuring CloudFront's custom error pages is the most efficient and least disruptive approach. It leverages existing infrastructure (CloudFront) and minimizes operational overhead.

Supporting Links:

Amazon S3 Static Website Hosting:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/WebsiteHosting.html
CloudFront Custom Error Pages:
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/custom-error-pages.html

Question: 11

A company has many AWS accounts and uses AWS Organizations to manage all of them. A solutions architect must implement a solution that the company can use to share a common network across multiple accounts. The company's infrastructure team has a dedicated infrastructure account that has a VPC. The infrastructure team must use this account to manage the network. Individual accounts cannot have the ability to manage their own networks. However, individual accounts must be able to create AWS resources within subnets.
Which combination of actions should the solutions architect perform to meet these requirements? (Choose two.)

A.Create a transit gateway in the infrastructure account.

B.Enable resource sharing from the AWS Organizations management account.

C.Create VPCs in each AWS account within the organization in AWS Organizations. Configure the VPCs to share the same CIDR range and subnets as the VPC in the infrastructure account. Peer the VPCs in each individual account with the VPC in the infrastructure account.

D.Create a resource share in AWS Resource Access Manager in the infrastructure account. Select the specific AWS Organizations OU that will use the shared network. Select each subnet to associate with the resource share.

E.Create a resource share in AWS Resource Access Manager in the infrastructure account. Select the specific AWS Organizations OU that will use the shared network. Select each prefix list to associate with the resource share.

Answer: BD

Explanation:

The correct answer is BD. Here's why:

D is correct: AWS Resource Access Manager (RAM) facilitates sharing AWS resources across accounts within an AWS Organization. By creating a resource share in the infrastructure account, the infrastructure team can specifically share the subnets of its managed VPC with other accounts or OUs. This aligns with the requirement that individual accounts can create resources within the defined subnets. Sharing specific subnets provides the necessary level of control and prevents individual accounts from managing the network infrastructure.
https://docs.aws.amazon.com/ram/latest/userguide/what-is.html

B is correct: To enable resource sharing across the organization, it's necessary to enable sharing from the AWS Organizations management account within RAM. This crucial step grants permissions for resource sharing to function across the organizational units. Without this step, the resource share created in the infrastructure account would be ineffective in granting access to other accounts.
https://docs.aws.amazon.com/ram/latest/userguide/getting-started-sharing.html

A is incorrect: While a Transit Gateway can centralize network traffic, it doesn't directly address the requirement of the infrastructure team managing the network and individual accounts using pre-existing subnets.

C is incorrect: Creating VPCs in each account with the same CIDR range and subnets as the infrastructure account's VPC is highly problematic. Overlapping CIDR ranges lead to routing conflicts and make interconnectivity exceedingly difficult. Furthermore, peering many VPCs directly is not scalable and introduces management overhead.

E is incorrect: Prefix lists are not directly used for sharing network access in the same way as subnets via AWS RAM. While prefix lists can be used in security groups and route tables, the primary requirement is to grant access to existing subnets for resource deployment, which RAM effectively handles when sharing subnets.

Therefore, enabling resource sharing from the management account and sharing subnets using RAM ensures central control of the network infrastructure within the infrastructure account while enabling individual accounts to deploy resources within those controlled subnets.

Question: 12

A company wants to use a third-party software-as-a-service (SaaS) application. The third-party SaaS application is consumed through several API calls. The third-party SaaS application also runs on AWS inside a VPC.
The company will consume the third-party SaaS application from inside a VPC. The company has internal security policies that mandate the use of private connectivity that does not traverse the internet. No resources that run in the company VPC are allowed to be accessed from outside the company's VPC. All permissions must conform to the principles of least privilege.
Which solution meets these requirements?

A.Create an AWS PrivateLink interface VPC endpoint. Connect this endpoint to the endpoint service that the third-party SaaS application provides. Create a security group to limit the access to the endpoint. Associate the security group with the endpoint.

B.Create an AWS Site-to-Site VPN connection between the third-party SaaS application and the company VPC. Configure network ACLs to limit access across the VPN tunnels.

C.Create a VPC peering connection between the third-party SaaS application and the company VPUpdate route tables by adding the needed routes for the peering connection.

D.Create an AWS PrivateLink endpoint service. Ask the third-party SaaS provider to create an interface VPC endpoint for this endpoint service. Grant permissions for the endpoint service to the specific account of the third-party SaaS provider.

Answer: A

Explanation:

The most suitable solution is **A. Create an AWS PrivateLink interface VPC endpoint. Connect this endpoint to the endpoint service that the third-party SaaS application provides. Create a security group to limit the access to the endpoint. Associate the security group with the endpoint.**

Here's a detailed justification:

Private Connectivity: AWS PrivateLink provides private connectivity between VPCs, AWS services, and on-premises networks without exposing traffic to the public internet. This directly addresses the requirement for private connectivity.

SaaS Integration: PrivateLink allows secure access to SaaS applications running in other AWS accounts or VPCs through endpoint services, enabling the company to consume the third-party SaaS application privately. **No External Access:** Since PrivateLink operates within the AWS network, it ensures that no resources within the company's VPC are accessible from outside, aligning with the security policy.

Least Privilege: Security groups attached to the interface VPC endpoint allow granular control over inbound and outbound traffic, limiting access to only the necessary ports and protocols. This enforces the principle of least privilege.

Endpoint Service Configuration: The third-party SaaS application creates an endpoint service. The company, as the consumer, then creates an interface VPC endpoint in their VPC and connects it to the provider's endpoint service.

Here's why the other options are less suitable:

B. AWS Site-to-Site VPN: While VPNs provide secure connectivity, they involve managing VPN tunnels, which adds operational overhead. Also, VPNs, while encrypted, still involve routing traffic over potentially shared networks, and may not be the most preferred solution when PrivateLink provides a more direct, private option. **C. VPC Peering:** VPC peering allows direct network connectivity between VPCs, but it does not offer the same level of control and security as PrivateLink. Peering requires managing overlapping IP address ranges and doesn't provide as much isolation. More importantly, VPC Peering is generally not suggested between VPCs managed by different organizations (like this scenario involving a third-party SaaS provider).

D. AWS PrivateLink Endpoint Service creation by the company: The company needs to consume the service, not provide it. Therefore, the third-party SaaS provider has to create and manage the Endpoint Service. The company can only create the Endpoint.

Authoritative Links:

AWS PrivateLink:https://aws.amazon.com/privatelink/
Interface VPC Endpoints (AWS PrivateLink):https://docs.aws.amazon.com/vpc/latest/privatelink/vpc-endpoints.html

Question: 13

A company needs to implement a patching process for its servers. The on-premises servers and Amazon EC2 instances use a variety of tools to perform patching. Management requires a single report showing the patch status of all the servers and instances.
Which set of actions should a solutions architect take to meet these requirements?

A.Use AWS Systems Manager to manage patches on the on-premises servers and EC2 instances. Use Systems Manager to generate patch compliance reports.

B.Use AWS OpsWorks to manage patches on the on-premises servers and EC2 instances. Use Amazon QuickSight integration with OpsWorks to generate patch compliance reports.

C.Use an Amazon EventBridge rule to apply patches by scheduling an AWS Systems Manager patch remediation job. Use Amazon Inspector to generate patch compliance reports.

D.Use AWS OpsWorks to manage patches on the on-premises servers and EC2 instances. Use AWS X-Ray to post the patch status to AWS Systems Manager OpsCenter to generate patch compliance reports.

Answer: A

Explanation:

The best approach to achieve a unified patching report across on-premises servers and EC2 instances is to leverage AWS Systems Manager. Systems Manager is designed for centralized management of hybrid environments, encompassing both AWS and on-premises resources. Option A directly uses Systems Manager for patch management on both on-premises servers and EC2 instances, providing a single pane of glass for controlling the patching process.

Crucially, Systems Manager offers built-in reporting capabilities through its Patch Manager feature. These capabilities include the generation of patch compliance reports, satisfying the requirement of a single report showing patch statuses for all servers and instances.

Options B, C, and D are less suitable. OpsWorks (B and D) is primarily focused on application management and might not be the best choice for a comprehensive patching solution across both environments. Furthermore, its integration with QuickSight or X-Ray for reporting, while possible, is not as straightforward or natively integrated as Systems Manager's reporting features. EventBridge and Inspector (C) could be parts of a patching strategy, but EventBridge is primarily for event-driven automation, and Inspector focuses on security vulnerabilities, not necessarily patching. The primary focus should be on centralized management and reporting of patch status. Relying on Inspector for the patch compliance reports isn't its core functionality.

By centralizing patch management and leveraging its reporting capabilities, Systems Manager streamlines the process of creating and maintaining a single, comprehensive patching report. This reduces complexity and simplifies the monitoring of patching efforts across the entire infrastructure.

Further reading:

AWS Systems Manager Patch Manager
AWS Systems Manager Compliance

Question: 14

A company is running an application on several Amazon EC2 instances in an Auto Scaling group behind an Application Load Balancer. The load on the application varies throughout the day, and EC2 instances are scaled in and out on a regular basis. Log files from the EC2 instances are copied to a central Amazon S3 bucket every 15 minutes. The security team discovers that log files are missing from some of the terminated EC2 instances.
Which set of actions will ensure that log files are copied to the central S3 bucket from the terminated EC2 instances?

A.Create a script to copy log files to Amazon S3, and store the script in a file on the EC2 instance. Create an Auto Scaling lifecycle hook and an Amazon EventBridge rule to detect lifecycle events from the Auto Scaling group. Invoke an AWS Lambda function on the autoscaling:EC2_INSTANCE_TERMINATING transition to send ABANDON to the Auto Scaling group to prevent termination, run the script to copy the log files, and terminate the instance using the AWS SDK.

B.Create an AWS Systems Manager document with a script to copy log files to Amazon S3. Create an Auto Scaling lifecycle hook and an Amazon EventBridge rule to detect lifecycle events from the Auto Scaling group. Invoke an AWS Lambda function on the autoscaling:EC2_INSTANCE_TERMINATING transition to call the AWS Systems Manager API SendCommand operation to run the document to copy the log files and send CONTINUE to the Auto Scaling group to terminate the instance.

C.Change the log delivery rate to every 5 minutes. Create a script to copy log files to Amazon S3, and add the script to EC2 instance user data. Create an Amazon EventBridge rule to detect EC2 instance termination. Invoke an AWS Lambda function from the EventBridge rule that uses the AWS CLI to run the user-data script to copy the log files and terminate the instance.

D.Create an AWS Systems Manager document with a script to copy log files to Amazon S3. Create an Auto Scaling lifecycle hook that publishes a message to an Amazon Simple Notification Service (Amazon SNS) topic. From the SNS notification, call the AWS Systems Manager API SendCommand operation to run the document to copy the log files and send ABANDON to the Auto Scaling group to terminate the instance.

Answer: B

Explanation:

The correct answer is **B**. Here's why:

The Problem: EC2 instances in an Auto Scaling group are being terminated before their log files are copied to S3, resulting in data loss. We need a mechanism to reliably copy logs during the termination process.

Why B is the Best Solution:

AWS Systems Manager (SSM) Documents: SSM Documents allow you to define automated tasks to be executed on EC2 instances. This is ideal for the log copying task. Storing the script as an SSM document ensures consistent execution and easier management compared to embedding it directly within instance configuration or Lambda.

Auto Scaling Lifecycle Hooks: Lifecycle hooks provide a pause in the instance termination process, allowing us to execute tasks before the instance is fully terminated. This is crucial to ensure log files are copied before the instance disappears.

EventBridge Rule & Lambda: EventBridge detects the autoscaling:EC2_INSTANCE_TERMINATING event, triggering a Lambda function. This function serves as the orchestrator, connecting the lifecycle hook with the SSM document execution.

Lambda's Role: The Lambda function calls the SSM SendCommand API operation to execute the pre-defined SSM document on the terminating instance.

CONTINUE Action: After the SSM document (log copy script) completes, the Lambda function sends a CONTINUE signal to the Auto Scaling group. This allows the instance termination process to proceed gracefully after the logs have been successfully copied.

Why other options are incorrect:

A: Holding the instance termination with ABANDON and manually terminating with the SDK is risky. If the Lambda fails, the instance could remain running indefinitely, incurring costs. Furthermore, manually terminating the instance bypasses the Auto Scaling group's management, potentially leading to inconsistencies.

C: User data scripts run during instance launch, not termination. Trying to trigger user data scripts during termination via EventBridge and Lambda is ineffective and not the intended use case. Also, shortening the log delivery interval is a workaround, not a guaranteed solution.

D: Using SNS for this specific task introduces unnecessary complexity. SNS is primarily for fan-out notifications. Directly calling the SSM API from the Lambda function triggered by EventBridge is more efficient and reliable. Furthermore, using ABANDON as in option A has the same issues.

In summary, Option B provides a reliable and automated solution by leveraging SSM Documents, Auto Scaling lifecycle hooks, EventBridge, and Lambda to guarantee log files are copied before instance termination.

Authoritative Links:

AWS Systems Manager: https://aws.amazon.com/systems-manager/
Auto Scaling Lifecycle Hooks: https://docs.aws.amazon.com/autoscaling/ec2/userguide/lifecycle-hooks.html Amazon EventBridge: https://aws.amazon.com/eventbridge/
AWS Lambda: https://aws.amazon.com/lambda/

Question: 15

A company is using multiple AWS accounts. The DNS records are stored in a private hosted zone for Amazon Route

53 in Account A. The company's applications and databases are running in Account B.
A solutions architect will deploy a two-tier application in a new VPC. To simplify the configuration, the db.example.com CNAME record set for the Amazon RDS endpoint was created in a private hosted zone for Amazon Route 53. During deployment, the application failed to start. Troubleshooting revealed that db.example.com is not resolvable on the Amazon EC2 instance. The solutions architect confirmed that the record set was created correctly in Route 53.
Which combination of steps should the solutions architect take to resolve this issue? (Choose two.)

A.Deploy the database on a separate EC2 instance in the new VPC. Create a record set for the instance's private IP in the private hosted zone.

B.Use SSH to connect to the application tier EC2 instance. Add an RDS endpoint IP address to the /etc/resolv.conf file.

C.Create an authorization to associate the private hosted zone in Account A with the new VPC in Account B.

D.Create a private hosted zone for the example com domain in Account B. Configure Route 53 replication between AWS accounts.

E.Associate a new VPC in Account B with a hosted zone in Account A. Delete the association authorization in Account A.

Answer: CE

Explanation:

Let's analyze why options C and E are the correct solutions to resolve the DNS resolution issue, and why the other options are not ideal.

The core problem is that the EC2 instance in Account B cannot resolve the db.example.com CNAME record, even though it exists in the Route 53 private hosted zone in Account A. This means the VPC in Account B is not authorized to query the private hosted zone in Account A.

Option C: Create an authorization to associate the private hosted zone in Account A with the new VPC in Account B. This is a necessary step to enable cross-account DNS resolution. When you create a private hosted zone, it's initially only associated with the VPCs in the same account. To allow VPCs in other accounts to resolve records within the hosted zone, you need to create an authorization. The account that owns the VPC (Account B) can then associate its VPC with the hosted zone using the authorization ID provided by the hosted zone owner (Account A).

Option E: Associate the new VPC in Account B with the hosted zone in Account A. This is the action taken by the account that owns the VPC (Account B) after an authorization has been created in Account A. This step effectively grants the VPC in Account B permission to query the private hosted zone in Account A. The prompt specifies to create an authorization with a new VPC, so we can assume there is not an authorization already made. The second part, "Delete the association authorization in Account A," is incorrect. Deleting the authorization would immediately break the ability of any VPC using the authorization to resolve DNS through this private hosted zone. Since it's necessary to create a new authorization, you are not deleting an existing authorization. Thus, associating the VPC after the authorization is created is the correct answer.

Now, let's analyze why the other options are incorrect:

Option A: Deploy the database on a separate EC2 instance in the new VPC. Create a record set for the instance's private IP in the private hosted zone. While this would solve the immediate problem by hosting the database in the same VPC, it's not a practical long-term solution. It involves significant infrastructure changes and potentially refactoring the application to interact with the new database instance. It doesn't address the underlying issue of cross-account DNS resolution. It bypasses the centralized DNS management in Route 53, complicating future management.

Option B: Use SSH to connect to the application tier EC2 instance. Add an RDS endpoint IP address to the /etc/resolv.conf file. This is a fragile and non-scalable solution. Manually adding the IP address of the RDS

endpoint to /etc/resolv.conf breaks the principle of using DNS for service discovery and will lead to problems if the RDS instance is ever replaced or fails over (its IP address might change). It is a bad practice to manually modify /etc/resolv.conf on EC2 instances, and updates to the underlying operating system can potentially overwrite the manual changes.

Option D: Create a private hosted zone for the example com domain in Account B. Configure Route 53 replication between AWS accounts. Creating another private hosted zone is not the solution, because it introduces further complexity in DNS management across accounts. Route 53 replication is not the same as Private Hosted Zone association; the first is generally used for public DNS. This adds complexity and is likely to introduce DNS inconsistencies. The goal is to use the existing DNS records defined in Account A.

In summary, creating an authorization in Account A (Option C) and associating the VPC in Account B with the hosted zone using this authorization (Option E) is the correct approach to enable cross-account DNS resolution using Route 53 private hosted zones, without impacting existing infrastructure.

Supporting links:

Using Private Hosted Zones
Associating a VPC with a Private Hosted Zone

Question: 16

A company used Amazon EC2 instances to deploy a web fleet to host a blog site. The EC2 instances are behind an Application Load Balancer (ALB) and are configured in an Auto Scaling group. The web application stores all blog content on an Amazon EFS volume.
The company recently added a feature for bloggers to add video to their posts, attracting 10 times the previous user traffic. At peak times of day, users report buffering and timeout issues while attempting to reach the site or watch videos.
Which is the MOST cost-efficient and scalable deployment that will resolve the issues for users?

A.Reconfigure Amazon EFS to enable maximum I/O.

B.Update the blog site to use instance store volumes for storage. Copy the site contents to the volumes at launch and to Amazon S3 at shutdown.

C.Configure an Amazon CloudFront distribution. Point the distribution to an S3 bucket, and migrate the videos from EFS to Amazon S3.

D.Set up an Amazon CloudFront distribution for all site contents, and point the distribution at the ALB.

Answer: C

Explanation:

The best solution is to leverage Amazon CloudFront to distribute the video content stored in Amazon S3. The problem is that the current architecture, relying on EFS and EC2 instances behind an ALB, is struggling to handle the increased traffic from video content, causing buffering and timeouts.

Option C addresses this directly by offloading the video delivery to CloudFront. CloudFront is a CDN (Content Delivery Network) specifically designed for low-latency, high-throughput content delivery. By storing the videos in S3 and serving them through CloudFront, the load on the EC2 instances, EFS, and ALB is
significantly reduced. S3 provides scalable and cost-effective storage for the videos, and CloudFront caches the video content at edge locations closer to the users, drastically improving the user experience and reducing latency. This also means your origin server (ALB/EC2) don't need to serve the videos, thus reducing the overall CPU/Memory usage on the EC2 instances.

Option A, reconfiguring EFS for maximum I/O, may improve performance, but it won't scale as effectively as a CDN and will be more expensive. EFS is not designed for serving static content directly to a large number of users.

Option B, using instance store volumes, is not a persistent storage solution and would require complex data synchronization mechanisms between instances and S3. This adds unnecessary operational overhead and is not a cost-effective or scalable solution. Additionally, instance store volumes are ephemeral, meaning data is lost when the instance is stopped or terminated.

Option D, using CloudFront for all site content, while potentially beneficial, is not the most cost-efficient way to solve the current issue. The primary bottleneck is the video content, so focusing on distributing that through CloudFront offers the best balance of cost and performance improvement. Migrating all site content would require a more complex setup with Origin Access Identity to protect the S3 bucket, adding
unnecessary complexity given the primary issue is video delivery.

Therefore, option C is the most cost-efficient and scalable deployment for resolving the user experience issues. It leverages the strengths of S3 for storage and CloudFront for content delivery to offload the video delivery from the web servers, improving performance and scalability while minimizing cost.

Further Research:

Amazon CloudFront:https://aws.amazon.com/cloudfront/
Amazon S3:https://aws.amazon.com/s3/
Amazon EFS:https://aws.amazon.com/efs/

Question: 17

A company with global offices has a single 1 Gbps AWS Direct Connect connection to a single AWS Region. The company's on-premises network uses the connection to communicate with the company's resources in the AWS Cloud. The connection has a single private virtual interface that connects to a single VPC.
A solutions architect must implement a solution that adds a redundant Direct Connect connection in the same Region. The solution also must provide connectivity to other Regions through the same pair of Direct Connect connections as the company expands into other Regions.
Which solution meets these requirements?

A.Provision a Direct Connect gateway. Delete the existing private virtual interface from the existing connection. Create the second Direct Connect connection. Create a new private virtual interface on each connection, and connect both private virtual interfaces to the Direct Connect gateway. Connect the Direct Connect gateway to the single VPC.

B.Keep the existing private virtual interface. Create the second Direct Connect connection. Create a new private virtual interface on the new connection, and connect the new private virtual interface to the single VPC.

C.Keep the existing private virtual interface. Create the second Direct Connect connection. Create a new public virtual interface on the new connection, and connect the new public virtual interface to the single VPC.

D.Provision a transit gateway. Delete the existing private virtual interface from the existing connection. Create the second Direct Connect connection. Create a new private virtual interface on each connection, and connect both private virtual interfaces to the transit gateway. Associate the transit gateway with the single VPC.

Answer: A

Explanation:

The correct answer is A because it leverages the AWS Direct Connect Gateway (DXGW) to achieve both redundancy and connectivity to multiple regions. Here's a breakdown:

Redundancy: Creating a second Direct Connect connection and connecting both to the DXGW ensures that if one connection fails, traffic can failover to the other, providing high availability. Each connection has its own private virtual interface (VIF).

Multi-Region Connectivity: The DXGW acts as a central hub, allowing connections from multiple Direct Connect locations (on-premises) to reach multiple AWS Regions. This fulfills the requirement of extending connectivity to other Regions as the company expands.

Option B is incorrect because connecting each Direct Connect connection directly to the same VPC does not inherently allow cross-region connectivity. While redundant, it's limited to the single VPC and Region.

Option C is incorrect. Using a public virtual interface is not the appropriate way to connect to resources within a VPC. Public VIFs are used to access public AWS services and are not intended for private network connectivity to a VPC.

Option D suggests using a Transit Gateway, which is a valid option for connecting multiple VPCs and on-premises networks. However, in this scenario, a Direct Connect Gateway is a more suitable choice since it is specifically designed to handle Direct Connect connections and their integration with AWS networks, including cross-region connectivity. Transit Gateway adds unnecessary complexity since the requirement is about Direct Connect redundancy and multi-region Direct Connect connectivity.

In summary, option A provides the most cost-effective and appropriate solution by utilizing the Direct Connect Gateway to establish redundant Direct Connect connections and enabling future expansion to other AWS Regions through the same connections, all while maintaining private connectivity to VPCs.

Supporting Links:

AWS Direct Connect Gateway: https://docs.aws.amazon.com/directconnect/latest/UserGuide/direct-connect-gateways-intro.html
AWS Direct Connect Resiliency Recommendations: https://aws.amazon.com/premiumsupport/knowledge-center/direct-connect-resiliency/

Question: 18

A company has a web application that allows users to upload short videos. The videos are stored on Amazon EBS volumes and analyzed by custom recognition software for categorization.
The website contains static content that has variable traffic with peaks in certain months. The architecture consists of Amazon EC2 instances running in an Auto Scaling group for the web application and EC2 instances running in an Auto Scaling group to process an Amazon SQS queue. The company wants to re-architect the application to reduce operational overhead using AWS managed services where possible and remove dependencies on third-party software.
Which solution meets these requirements?

A.Use Amazon ECS containers for the web application and Spot instances for the Auto Scaling group that processes the SQS queue. Replace the custom software with Amazon Rekognition to categorize the videos.

B.Store the uploaded videos in Amazon EFS and mount the file system to the EC2 instances for the web application. Process the SQS queue with an AWS Lambda function that calls the Amazon Rekognition API to categorize the videos.

C.Host the web application in Amazon S3. Store the uploaded videos in Amazon S3. Use S3 event notification to publish events to the SQS queue. Process the SQS queue with an AWS Lambda function that calls the Amazon Rekognition API to categorize the videos.

D.Use AWS Elastic Beanstalk to launch EC2 instances in an Auto Scaling group for the web application and launch a worker environment to process the SQS queue. Replace the custom software with Amazon Rekognition to categorize the videos.

Answer: C

Explanation:

The correct answer is C because it leverages AWS managed services to minimize operational overhead and removes dependencies on EBS and third-party software, aligning perfectly with the stated requirements. Hosting the static web application on Amazon S3 utilizes its highly scalable and cost-effective object storage capabilities, eliminating the need for EC2 instances and Auto Scaling for static content. S3's event
notifications trigger the SQS queue when new videos are uploaded, creating an event-driven architecture.
Processing the SQS queue with AWS Lambda functions, which invoke the Amazon Rekognition API,

completely replaces the EC2-based processing with serverless, managed compute. Lambda eliminates the operational burden of managing EC2 instances for queue processing. Amazon Rekognition provides managed video analysis capabilities, replacing the custom software and related operational overhead. This serverless approach optimizes cost and reduces management overhead significantly. Using S3, SQS, Lambda, and Rekognition leads to a fully managed, scalable, and cost-effective solution that meets all specified needs.

Alternatives A, B, and D do not fully utilize AWS managed services and maintain dependencies on EC2 instances and Auto Scaling for static content serving or video storage, which are not ideal for reducing operational overhead as much as option C.

Further reading:

Amazon S3:https://aws.amazon.com/s3/
Amazon SQS:https://aws.amazon.com/sqs/
AWS Lambda:https://aws.amazon.com/lambda/
Amazon Rekognition:https://aws.amazon.com/rekognition/

Question: 19

A company has a serverless application comprised of Amazon CloudFront, Amazon API Gateway, and AWS Lambda functions. The current deployment process of the application code is to create a new version number of the Lambda function and run an AWS CLI script to update. If the new function version has errors, another CLI script reverts by deploying the previous working version of the function. The company would like to decrease the time to deploy new versions of the application logic provided by the Lambda functions, and also reduce the time to detect and revert when errors are identified.
How can this be accomplished?

A.Create and deploy nested AWS CloudFormation stacks with the parent stack consisting of the AWS CloudFront distribution and API Gateway, and the child stack containing the Lambda function. For changes to Lambda, create an AWS CloudFormation change set and deploy; if errors are triggered, revert the AWS CloudFormation change set to the previous version.

B.Use AWS SAM and built-in AWS CodeDeploy to deploy the new Lambda version, gradually shift traffic to the new version, and use pre-traffic and post-traffic test functions to verify code. Rollback if Amazon CloudWatch alarms are triggered.

C.Refactor the AWS CLI scripts into a single script that deploys the new Lambda version. When deployment is completed, the script tests execute. If errors are detected, revert to the previous Lambda version.

D.Create and deploy an AWS CloudFormation stack that consists of a new API Gateway endpoint that references the new Lambda version. Change the CloudFront origin to the new API Gateway endpoint, monitor errors and if detected, change the AWS CloudFront origin to the previous API Gateway endpoint.

Answer: B

Explanation:

The correct answer is **B. Use AWS SAM and built-in AWS CodeDeploy to deploy the new Lambda version, gradually shift traffic to the new version, and use pre-traffic and post-traffic test functions to verify code. Rollback if Amazon CloudWatch alarms are triggered.**

Here's why:

AWS SAM (Serverless Application Model) simplifies building and deploying serverless applications. It provides a higher-level abstraction over CloudFormation, making it easier to define serverless resources like Lambda functions, API Gateway, and CloudFront. Crucially, SAM integrates seamlessly with AWS CodeDeploy for safe and controlled deployments of Lambda functions.

CodeDeploy allows for gradual traffic shifting to new Lambda versions, which is essential for minimizing the impact of errors during deployments. This can be done using deployment strategies like Canary or Linear deployments. Canary deployments release the new version to a small subset of users, while Linear

deployments increase traffic gradually over time. This controlled release allows you to identify issues before they affect the entire user base.

Pre-traffic and post-traffic test functions further enhance the safety of deployments. These functions automatically run before and after traffic is shifted to the new version, respectively. They can execute automated tests to verify that the new version is functioning correctly and meeting performance requirements. If the tests fail, CodeDeploy can automatically roll back the deployment to the previous working version.

Moreover, the integration with Amazon CloudWatch alarms adds another layer of safety. CloudWatch alarms can monitor metrics like error rates, latency, or resource utilization. If any of these metrics exceed predefined thresholds after the new version is deployed, the alarms can trigger a rollback, ensuring that the application remains stable.

Option A is less ideal because while CloudFormation provides infrastructure-as-code capabilities, using change sets for reverting doesn't provide the granularity and automated rollback mechanisms that CodeDeploy offers. Also managing nested stacks and manually reverting can be time-consuming.

Option C proposes a single CLI script, which, while simpler, lacks the safety features of gradual traffic shifting and automated rollback provided by CodeDeploy. It increases the risk of impacting the entire user base if an error is introduced.

Option D introduces new API Gateway endpoints and CloudFront origins, which adds unnecessary complexity to the deployment process. This also potentially leads to inconsistencies and higher operational overhead. SAM and CodeDeploy provide a more streamlined and automated solution.

In summary, SAM and CodeDeploy offer the best combination of rapid deployment, error detection, and automated rollback for serverless applications, addressing the company's requirements effectively.

Supporting Links:

AWS SAM:https://aws.amazon.com/serverless/sam/
AWS CodeDeploy:https://aws.amazon.com/codedeploy/
Lambda Deployment using CodeDeploy:
https://docs.aws.amazon.com/codedeploy/latest/userguide/tutorials-lambda.html

Question: 20

A company is planning to store a large number of archived documents and make the documents available to employees through the corporate intranet. Employees will access the system by connecting through a client VPN service that is attached to a VPC. The data must not be accessible to the public.
The documents that the company is storing are copies of data that is held on physical media elsewhere. The number of requests will be low. Availability and speed of retrieval are not concerns of the company.
Which solution will meet these requirements at the LOWEST cost?

A.Create an Amazon S3 bucket. Configure the S3 bucket to use the S3 One Zone-Infrequent Access (S3 One Zone-IA) storage class as default. Configure the S3 bucket for website hosting. Create an S3 interface endpoint. Configure the S3 bucket to allow access only through that endpoint.

B.Launch an Amazon EC2 instance that runs a web server. Attach an Amazon Elastic File System (Amazon EFS) file system to store the archived data in the EFS One Zone-Infrequent Access (EFS One Zone-IA) storage class Configure the instance security groups to allow access only from private networks.

C.Launch an Amazon EC2 instance that runs a web server Attach an Amazon Elastic Block Store (Amazon EBS) volume to store the archived data. Use the Cold HDD (sc1) volume type. Configure the instance security groups to allow access only from private networks.

D.Create an Amazon S3 bucket. Configure the S3 bucket to use the S3 Glacier Deep Archive storage class as default. Configure the S3 bucket for website hosting. Create an S3 interface endpoint. Configure the S3 bucket to allow access only through that endpoint.

Answer: A

Explanation:

The best solution is A because it leverages S3 One Zone-IA, which is designed for infrequently accessed data that doesn't require the high availability of standard S3. Since the requirements explicitly state that availability and speed are not concerns, and the data is backed up elsewhere, the lower cost of S3 One Zone-IA makes it ideal. Configuring an S3 interface endpoint ensures that traffic to S3 stays within the VPC, meeting the requirement that data is not publicly accessible. While website hosting is mentioned, it's not actually enabled for public access in this scenario because access is controlled via the interface endpoint and VPC routing. This approach efficiently balances cost and security.

Option D, while using Glacier Deep Archive (the cheapest S3 storage), is incorrect because Glacier Deep Archive is designed for extremely infrequent access and retrieval times of hours, which is unlikely to meet the need for employee access, however infrequent. Although not explicitly ruled out, Option A is cheaper and suitable given all the other requirements.

Options B and C are less suitable because they involve running EC2 instances, which incur compute costs in addition to storage costs. They also require more management overhead compared to S3. EFS and EBS are more suited for active file systems and block storage needs, respectively, not cost-effectively archiving infrequently accessed data. Even using the One Zone-IA tier of EFS, it is still more expensive than S3 One Zone-IA. Also, the maintenance overhead of EBS and EFS will be a burden. S3, being a fully managed service, is the simplest and likely lowest cost.

Here are some resources for further reading:

Amazon S3 Storage Classes:https://aws.amazon.com/s3/storage-classes/
Amazon S3 Glacier Deep Archive:https://aws.amazon.com/s3/storage-classes/glacier/ **VPC Endpoints:**https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html **Amazon EFS Storage Classes:**https://aws.amazon.com/efs/pricing/

Question: 21

A company is using an on-premises Active Directory service for user authentication. The company wants to use the same authentication service to sign in to the company's AWS accounts, which are using AWS Organizations. AWS Site-to-Site VPN connectivity already exists between the on-premises environment and all the company's AWS accounts.
The company's security policy requires conditional access to the accounts based on user groups and roles. User identities must be managed in a single location.
Which solution will meet these requirements?

A.Configure AWS IAM Identity Center (AWS Single Sign-On) to connect to Active Directory by using SAML 2.0. Enable automatic provisioning by using the System for Cross-domain Identity Management (SCIM) v2.0 protocol. Grant access to the AWS accounts by using attribute-based access controls (ABACs).

B.Configure AWS IAM Identity Center (AWS Single Sign-On) by using IAM Identity Center as an identity source. Enable automatic provisioning by using the System for Cross-domain Identity Management (SCIM) v2.0 protocol. Grant access to the AWS accounts by using IAM Identity Center permission sets.

C.In one of the company's AWS accounts, configure AWS Identity and Access Management (IAM) to use a SAML 2.0 identity provider. Provision IAM users that are mapped to the federated users. Grant access that corresponds to appropriate groups in Active Directory. Grant access to the required AWS accounts by using cross-account IAM users.

D.In one of the company's AWS accounts, configure AWS Identity and Access Management (IAM) to use an OpenID Connect (OIDC) identity provider. Provision IAM roles that grant access to the AWS account for the federated users that correspond to appropriate groups in Active Directory. Grant access to the required AWS accounts by using cross-account IAM roles.

Answer: A

Explanation:

The correct answer is A because it utilizes AWS IAM Identity Center (successor to AWS Single Sign-On) configured with Active Directory for centralized user authentication. This leverages the existing VPN connection. SAML 2.0 is the industry standard for federated identity, allowing Active Directory to assert user identities to AWS. SCIM v2.0 automates the provisioning and deprovisioning of users and groups from Active Directory to IAM Identity Center, keeping identity information synchronized. ABAC uses attributes (like user groups and roles from Active Directory) to define permissions, fulfilling the requirement for conditional access based on user groups and roles, without the complexity of managing individual IAM roles per user. IAM Identity Center integrates with AWS Organizations, enabling single sign-on access to multiple AWS accounts.

Option B is incorrect because using IAM Identity Center as an identity source doesn't leverage the existing on-premises Active Directory, thus failing the requirement for using the existing authentication service.

Option C is incorrect because manually configuring IAM with SAML in a single account and then using cross-account IAM users is a complex and less scalable solution compared to using IAM Identity Center across the organization. User management becomes cumbersome as roles and users need to be maintained across multiple accounts.

Option D is incorrect because while OIDC can also be used for federation, SAML is generally preferred for integrating with Active Directory. Furthermore, relying on cross-account IAM roles still presents the same scalability challenges as option C and does not provide centralized user management.

Here are some useful links:

AWS IAM Identity Center:https://aws.amazon.com/iam/identity-center/
SAML 2.0:https://en.wikipedia.org/wiki/Security_Assertion_Markup_Language
SCIM:https://en.wikipedia.org/wiki/System_for_Cross-domain_Identity_Management
Attribute-Based Access Control (ABAC):https://docs.aws.amazon.com/IAM/latest/UserGuide/security-abac.html

Question: 22

A software company has deployed an application that consumes a REST API by using Amazon API Gateway, AWS Lambda functions, and an Amazon DynamoDB table. The application is showing an increase in the number of errors during PUT requests. Most of the PUT calls come from a small number of clients that are authenticated with specific API keys.
A solutions architect has identified that a large number of the PUT requests originate from one client. The API is noncritical, and clients can tolerate retries of unsuccessful calls. However, the errors are displayed to customers and are causing damage to the API's reputation.
What should the solutions architect recommend to improve the customer experience?

A.Implement retry logic with exponential backoff and irregular variation in the client application. Ensure that the errors are caught and handled with descriptive error messages.

B.Implement API throttling through a usage plan at the API Gateway level. Ensure that the client application handles code 429 replies without error.

C.Turn on API caching to enhance responsiveness for the production stage. Run 10-minute load tests. Verify that the cache capacity is appropriate for the workload.

D.Implement reserved concurrency at the Lambda function level to provide the resources that are needed during sudden increases in traffic.

Answer: B

Explanation:

The best approach is **B. Implement API throttling through a usage plan at the API Gateway level. Ensure that the client application handles code 429 replies without error.**

Here's why:

The problem is that a small number of clients, particularly one, are overwhelming the API with PUT requests, leading to errors. Throttling directly addresses this by limiting the number of requests a specific client can make within a given timeframe. API Gateway usage plans are designed for this purpose. By setting a limit on the offending client's API key through a usage plan tied to a specific API Gateway stage, you control the rate at which they can make requests.

Crucially, the application already tolerates retries. When a client exceeds its request limit, API Gateway returns a 429 "Too Many Requests" error. The problem states the client's errors are visible and damaging the API's reputation. If the client application is properly designed to handle 429 errors gracefully (by implementing retry logic with backoff or displaying a user-friendly message), the user experience is improved.

Why other options are not as effective:

A (Retry logic with exponential backoff): While retry logic is good practice, it doesn't solve the root cause of the problem. If the client continues to send excessive requests, retries will only exacerbate the situation by placing even more load on the API, potentially leading to even more errors.

C (API caching): Caching enhances read performance (GET requests), not write performance (PUT requests). Caching is irrelevant when dealing with an overload of PUT requests causing errors.

D (Reserved concurrency): Reserved concurrency for Lambda could help avoid throttling at the Lambda function level. However, the problem states the excessive traffic originates from a single client. Reserved concurrency would not prevent that client from overwhelming the API, and potentially still exhausting downstream resources like DynamoDB. Throttling at the API Gateway before reaching the Lambda functions is the more targeted and efficient solution. It prevents the excessive requests from even reaching the backend.

In summary, throttling at the API Gateway with usage plans is the most direct and effective way to control the rate of requests from a specific client, thereby mitigating the overload and improving the overall customer experience, especially given that retries are acceptable.

Authoritative Links:

Amazon API Gateway Throttling:https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-request-throttling.html
API Gateway Usage Plans:https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-usage-plans.html

Question: 23

A company is running a data-intensive application on AWS. The application runs on a cluster of hundreds of Amazon EC2 instances. A shared file system also runs on several EC2 instances that store 200 TB of data. The application reads and modifies the data on the shared file system and generates a report. The job runs once monthly, reads a subset of the files from the shared file system, and takes about 72 hours to complete. The compute instances scale in an Auto Scaling group, but the instances that host the shared file system run continuously. The compute and storage instances are all in the same AWS Region.
A solutions architect needs to reduce costs by replacing the shared file system instances. The file system must provide high performance access to the needed data for the duration of the 72-hour run.
Which solution will provide the LARGEST overall cost reduction while meeting these requirements?

A. Migrate the data from the existing shared file system to an Amazon S3 bucket that uses the S3 Intelligent-Tiering storage class. Before the job runs each month, use Amazon FSx for Lustre to create a new file system

with the data from Amazon S3 by using lazy loading. Use the new file system as the shared storage for the duration of the job. Delete the file system when the job is complete.

B.Migrate the data from the existing shared file system to a large Amazon Elastic Block Store (Amazon EBS) volume with Multi-Attach enabled. Attach the EBS volume to each of the instances by using a user data script in the Auto Scaling group launch template. Use the EBS volume as the shared storage for the duration of the job. Detach the EBS volume when the job is complete

C.Migrate the data from the existing shared file system to an Amazon S3 bucket that uses the S3 Standard storage class. Before the job runs each month, use Amazon FSx for Lustre to create a new file system with the data from Amazon S3 by using batch loading. Use the new file system as the shared storage for the duration of the job. Delete the file system when the job is complete.

D.Migrate the data from the existing shared file system to an Amazon S3 bucket. Before the job runs each month, use AWS Storage Gateway to create a file gateway with the data from Amazon S3. Use the file gateway as the shared storage for the job. Delete the file gateway when the job is complete.

Answer: A

Explanation:

The most cost-effective solution is **A. Migrate the data from the existing shared file system to an Amazon S3 bucket that uses the S3 Intelligent-Tiering storage class. Before the job runs each month, use Amazon FSx for Lustre to create a new file system with the data from Amazon S3 by using lazy loading. Use the new file system as the shared storage for the duration of the job. Delete the file system when the job is complete.**

Here's a detailed justification:

Cost Reduction Focus: The core requirement is the largest overall cost reduction. Maintaining a dedicated shared file system on EC2 instances continuously is expensive due to the ongoing compute and storage costs.

S3 for Data Storage: S3 is a highly scalable, durable, and cost-effective object storage service. Storing the 200 TB of data in S3 is significantly cheaper than storing it on continuously running EC2-backed file systems.

S3 Intelligent-Tiering: Using S3 Intelligent-Tiering further optimizes cost by automatically moving data between frequent and infrequent access tiers based on access patterns. This is ideal because the data is only accessed once a month.

FSx for Lustre for Performance: Amazon FSx for Lustre provides a high-performance, parallel file system optimized for data-intensive workloads. It can be easily integrated with S3, allowing you to import data from S3 when needed and export results back to S3.

Lazy Loading with FSx for Lustre: Lazy loading is a crucial feature of FSx for Lustre. With lazy loading, only the data that is actually accessed by the application is copied from S3 to FSx for Lustre. This significantly reduces the data transfer costs and the amount of storage needed on the FSx for Lustre file system. **Ephemeral FSx for Lustre:** Creating the FSx for Lustre file system only when needed for the 72-hour job and deleting it afterward ensures you only pay for the file system resources when they are actively being used. This avoids the cost of maintaining a persistent file system.

Why other options are less cost-effective:

Option B (EBS Multi-Attach): While EBS Multi-Attach allows multiple EC2 instances to access the same volume, it does not address the underlying problem of storing 200 TB of data on an EBS volume, which is comparatively expensive. Also, EBS Multi-Attach has limitations and might not scale well for hundreds of instances.

Option C (S3 Standard with Batch Loading): Batch loading to FSx for Lustre means copying all 200 TB from S3 to FSx for Lustre regardless of how much data is accessed in a month. Since only a subset is accessed, this wastes cost.

Option D (AWS Storage Gateway): AWS Storage Gateway, specifically the File Gateway, provides a local cache for frequently accessed data. However, for a monthly job accessing a subset of a large dataset, the cache benefits are minimal. File Gateway also doesn't offer the same level of performance as FSx for Lustre for data-intensive applications.

Authoritative Links:

Amazon S3 Storage Classes:https://aws.amazon.com/s3/storage-classes/
Amazon FSx for Lustre:https://aws.amazon.com/fsx/lustre/
AWS Storage Gateway:https://aws.amazon.com/storagegateway/

Question: 24

A company is developing a new service that will be accessed using TCP on a static port. A solutions architect must ensure that the service is highly available, has redundancy across Availability Zones, and is accessible using the DNS name my.service.com, which is publicly accessible. The service must use fixed address assignments so other companies can add the addresses to their allow lists.
Assuming that resources are deployed in multiple Availability Zones in a single Region, which solution will meet these requirements?

A.Create Amazon EC2 instances with an Elastic IP address for each instance. Create a Network Load Balancer (NLB) and expose the static TCP port. Register EC2 instances with the NLB. Create a new name server record set named my.service.com, and assign the Elastic IP addresses of the EC2 instances to the record set. Provide the Elastic IP addresses of the EC2 instances to the other companies to add to their allow lists.

B.Create an Amazon ECS cluster and a service definition for the application. Create and assign public IP addresses for the ECS cluster. Create a Network Load Balancer (NLB) and expose the TCP port. Create a target group and assign the ECS cluster name to the NLCreate a new A record set named my.service.com, and assign the public IP addresses of the ECS cluster to the record set. Provide the public IP addresses of the ECS cluster to the other companies to add to their allow lists.

C.Create Amazon EC2 instances for the service. Create one Elastic IP address for each Availability Zone. Create a Network Load Balancer (NLB) and expose the assigned TCP port. Assign the Elastic IP addresses to the NLB for each Availability Zone. Create a target group and register the EC2 instances with the NLB. Create a new A (alias) record set named my.service.com, and assign the NLB DNS name to the record set.

D.Create an Amazon ECS cluster and a service definition for the application. Create and assign public IP address for each host in the cluster. Create an Application Load Balancer (ALB) and expose the static TCP port. Create a target group and assign the ECS service definition name to the ALB. Create a new CNAME record set and associate the public IP addresses to the record set. Provide the Elastic IP addresses of the Amazon EC2 instances to the other companies to add to their allow lists.

Answer: C

Explanation:

The correct answer is C because it leverages a Network Load Balancer (NLB) with Elastic IPs in each Availability Zone to provide a stable, publicly accessible endpoint for the service. The NLB is designed for high availability and fault tolerance across multiple Availability Zones, and the Elastic IPs ensure the service has static IP addresses that external companies can add to their allow lists. By using an A (alias) record pointing to the NLB's DNS name, the DNS resolution is handled dynamically, allowing the NLB to manage the underlying instances without requiring updates to the DNS records when instances change.

Option A is incorrect because directly assigning Elastic IPs to EC2 instances does not provide true high availability and load balancing across Availability Zones. While instances are accessible individually, there's no mechanism for automatic failover or traffic distribution.

Option B incorrectly suggests creating and assigning public IP addresses to the ECS cluster itself. ECS clusters do not inherently have public IPs assigned to them. ECS tasks running on EC2 launch types inherit the public IP addresses of the underlying instances. Assigning addresses to the cluster isn't a standard practice. Moreover, while NLBs can work with ECS, assigning ECS cluster names to target groups is not how NLBs handle traffic distribution to containers.

Option D is incorrect because Application Load Balancers (ALBs) primarily operate at Layer 7 (HTTP/HTTPS) and are less suitable for raw TCP traffic. While ECS can be integrated with ALBs, this option doesn't

emphasize the static IP requirement and suggests creating CNAME records with IP addresses, which is not best practice. CNAMES should point to DNS names, not IP addresses.

In summary, option C fulfills all the requirements: high availability through an NLB, redundancy across Availability Zones, accessibility via a public DNS name, and the use of static IP addresses provided by the NLB associated with each AZ through its Elastic IPs.

Relevant Documentation:

Network Load Balancer:https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html
Elastic IP Addresses:https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html
Route 53 Alias Records:https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-choosing-alias-non-alias.html

Question: 25

A company uses an on-premises data analytics platform. The system is highly available in a fully redundant configuration across 12 servers in the company's data center.
The system runs scheduled jobs, both hourly and daily, in addition to one-time requests from users. Scheduled jobs can take between 20 minutes and 2 hours to finish running and have tight SLAs. The scheduled jobs account for 65% of the system usage. User jobs typically finish running in less than 5 minutes and have no SLA. The user jobs account for 35% of system usage. During system failures, scheduled jobs must continue to meet SLAs. However, user jobs can be delayed.
A solutions architect needs to move the system to Amazon EC2 instances and adopt a consumption-based model to reduce costs with no long-term commitments. The solution must maintain high availability and must not affect the SLAs.
Which solution will meet these requirements MOST cost-effectively?

A.Split the 12 instances across two Availability Zones in the chosen AWS Region. Run two instances in each Availability Zone as On-Demand Instances with Capacity Reservations. Run four instances in each Availability Zone as Spot Instances.

B.Split the 12 instances across three Availability Zones in the chosen AWS Region. In one of the Availability Zones, run all four instances as On-Demand Instances with Capacity Reservations. Run the remaining instances as Spot Instances.

C.Split the 12 instances across three Availability Zones in the chosen AWS Region. Run two instances in each Availability Zone as On-Demand Instances with a Savings Plan. Run two instances in each Availability Zone as Spot Instances.

D.Split the 12 instances across three Availability Zones in the chosen AWS Region. Run three instances in each Availability Zone as On-Demand Instances with Capacity Reservations. Run one instance in each Availability Zone as a Spot Instance.

Answer: D

Explanation:

Here's a detailed justification for why option D is the most cost-effective solution while meeting the requirements:

The primary goal is to migrate the on-premises data analytics platform to EC2, reduce costs, and maintain high availability, especially for scheduled jobs with tight SLAs. The solution must be consumption-based without long-term commitments. User jobs can tolerate delays during failures.

Option D proposes splitting the 12 instances across three Availability Zones (AZs) for high availability. This strategy protects against single AZ failures. By using three On-Demand Instances with Capacity Reservations per AZ (9 total), the solution ensures that a core capacity is always available to handle the critical scheduled jobs. Capacity Reservations guarantee that the instances will be available when needed, thus upholding the SLAs. The Capacity Reservation ensures predictable performance and minimizes the risk of instance unavailability, crucial for scheduled jobs.

The remaining instance per AZ (3 total) is run as a Spot Instance. Spot Instances can be used for user jobs, since these are more fault-tolerant and can be delayed. Spot Instances provide significant cost savings when available.

Options A and B are less ideal because they rely more heavily on Spot Instances, creating more risk for scheduled jobs, or use only 2 AZs, reducing availability.

Option C uses Savings Plans and Spot Instances. Savings Plans, while providing discounts, involve a commitment (1 or 3 years), which contradicts the "no long-term commitments" requirement. Capacity Reservations ensure capacity is always available while allowing consumption-based pricing.

Therefore, Option D strikes the best balance between cost savings (using Spot Instances for fault-tolerant user jobs) and guaranteed capacity (using On-Demand Instances with Capacity Reservations for SLA-driven scheduled jobs), distributed across three AZs for high availability and cost-effectiveness.

Relevant Links:

Amazon EC2 Capacity Reservations:https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-capacity-reservations.html
Amazon EC2 Spot Instances:https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html
AWS Availability Zones:https://aws.amazon.com/about-aws/global-infrastructure/regions_availability_zones/

Question: 26

A security engineer determined that an existing application retrieves credentials to an Amazon RDS for MySQL database from an encrypted file in Amazon S3. For the next version of the application, the security engineer wants to implement the following application design changes to improve security:
The database must use strong, randomly generated passwords stored in a secure AWS managed service. The application resources must be deployed through AWS CloudFormation.
The application must rotate credentials for the database every 90 days.
A solutions architect will generate a CloudFormation template to deploy the application.
Which resources specified in the CloudFormation template will meet the security engineer’s requirements with the LEAST amount of operational overhead?

A.Generate the database password as a secret resource using AWS Secrets Manager. Create an AWS Lambda function resource to rotate the database password. Specify a Secrets Manager RotationSchedule resource to rotate the database password every 90 days.

B.Generate the database password as a SecureString parameter type using AWS Systems Manager Parameter Store. Create an AWS Lambda function resource to rotate the database password. Specify a Parameter Store RotationSchedule resource to rotate the database password every 90 days.

C.Generate the database password as a secret resource using AWS Secrets Manager. Create an AWS Lambda function resource to rotate the database password. Create an Amazon EventBridge scheduled rule resource to trigger the Lambda function password rotation every 90 days.

D.Generate the database password as a SecureString parameter type using AWS Systems Manager Parameter Store. Specify an AWS AppSync DataSource resource to automatically rotate the database password every 90 days.

Answer: A

Explanation:

The correct answer is A because it leverages AWS Secrets Manager for secure password generation, storage, and automated rotation, aligning with the security engineer's requirements with minimal operational overhead.

Here's a detailed justification:

AWS Secrets Manager: This service is designed for securely managing secrets like database passwords. It offers built-in rotation capabilities, reducing the operational burden of manual rotation.

https://aws.amazon.com/secrets-manager/

Random Password Generation: Secrets Manager can generate strong, random passwords, satisfying the requirement for strong credentials.

Secure Storage: Secrets Manager encrypts secrets at rest and in transit, enhancing security.

Automated Rotation: The RotationSchedule resource in Secrets Manager automates password rotation according to the specified frequency (90 days in this case), eliminating the need for manual intervention or custom scheduling mechanisms, thus minimizing operational overhead.

Lambda Function: A Lambda function acts as the rotation function, executing the necessary steps to update the database password and update the secret in Secrets Manager. Secrets Manager natively integrates with Lambda for this purpose.

CloudFormation Integration: Secrets Manager resources can be defined and deployed directly through CloudFormation, enabling infrastructure-as-code and streamlining deployment.

Option B is incorrect because AWS Systems Manager Parameter Store, while capable of storing secrets, does not have a built-in automated rotation mechanism like Secrets Manager. Parameter Store does not have a RotationSchedule resource.

Option C is incorrect because using Amazon EventBridge to trigger a Lambda function for rotation requires more configuration and management than the built-in RotationSchedule feature of Secrets Manager, leading to increased operational overhead. Also, while EventBridge would work, Secrets Manager is the designed and recommended approach.

Option D is incorrect because AWS AppSync is not intended for automatic password rotation of RDS databases. AppSync is a GraphQL service. Systems Manager Parameter Store lacks the native rotation features of Secrets Manager.

Question: 27

A company is storing data in several Amazon DynamoDB tables. A solutions architect must use a serverless architecture to make the data accessible publicly through a simple API over HTTPS. The solution must scale automatically in response to demand.
Which solutions meet these requirements? (Choose two.)

A.Create an Amazon API Gateway REST API. Configure this API with direct integrations to DynamoDB by using API Gateway's AWS integration type.

B.Create an Amazon API Gateway HTTP API. Configure this API with direct integrations to Dynamo DB by using API Gateway's AWS integration type.

C.Create an Amazon API Gateway HTTP API. Configure this API with integrations to AWS Lambda functions that return data from the DynamoDB tables.

D.Create an accelerator in AWS Global Accelerator. Configure this accelerator with AWS Lambda@Edge function integrations that return data from the DynamoDB tables.

E.Create a Network Load Balancer. Configure listener rules to forward requests to the appropriate AWS Lambda functions.

Answer: AC

Explanation:

Here's a detailed justification for why options A and C are the correct choices, and why the others are not:

Why A is correct:

Amazon API Gateway REST API: API Gateway is a fully managed service that makes it easy to create, publish, maintain, monitor, and secure APIs at any scale. It's designed for creating robust, scalable APIs for backend services. A REST API allows you to expose your DynamoDB data through standard HTTP methods (GET, POST, PUT, DELETE).

Direct Integrations (AWS Integration Type): API Gateway offers "AWS Service Proxy" integrations, which allow you to directly integrate with AWS services like DynamoDB without needing to write any code. You can map the incoming HTTP request parameters to DynamoDB operations (e.g., GetItem, Query, Scan). This greatly simplifies the architecture and reduces latency.

Why C is correct:

Amazon API Gateway HTTP API: Similar to REST APIs, HTTP APIs also offer a way to expose backend functionality via HTTP. They are often more cost-effective and have lower latency than REST APIs for simpler use cases.

Lambda Function Integration: This involves using Lambda functions as an intermediary layer between the API Gateway and DynamoDB. The API Gateway triggers the Lambda function upon receiving a request. The Lambda function then interacts with DynamoDB (e.g., querying the database), processes the retrieved data, and returns the response back to the API Gateway, which in turn sends it to the client. This offers flexibility in data transformation and business logic implementation.

Why B is incorrect:

While HTTP APIs are a viable option for creating APIs, direct integration with DynamoDB is not a supported feature for HTTP APIs. They are better suited for proxying requests to backend computes such as Lambda functions, HTTP endpoints or Application Load Balancers.

Why D is incorrect:

AWS Global Accelerator and Lambda@Edge: While Global Accelerator improves application availability and performance for globally distributed users, it is not primarily designed for API functionality as described in the question. While Lambda@Edge, which operates as part of CloudFront, could interact with DynamoDB, it's more commonly used for edge processing and caching closer to users. Using it as the primary API layer introduces unnecessary complexity compared to API Gateway.

Why E is incorrect:

Network Load Balancer (NLB) with Lambda: While NLBs can front Lambda functions, they are not typically used to expose API endpoints directly to the public. An NLB is a Layer 4 load balancer, which means it operates on the transport layer. It primarily handles TCP, TLS, and UDP traffic. API Gateway provides better features to define API endpoints and ensure security with authorization.

In summary: Options A and C provide the best serverless architectures for exposing DynamoDB data via a public API, offering scalability, security, and ease of management. Option A simplifies the setup with direct integrations, while Option C provides more flexibility and customization.

Authoritative Links:

Amazon API Gateway:https://aws.amazon.com/api-gateway/
AWS Lambda:https://aws.amazon.com/lambda/
Amazon DynamoDB:https://aws.amazon.com/dynamodb/
API Gateway AWS Integrations:https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-api-integration-types.html

Question: 28

A company has registered 10 new domain names. The company uses the domains for online marketing. The company needs a solution that will redirect online visitors to a specific URL for each domain. All domains and target URLs are defined in a JSON document. All DNS records are managed by Amazon Route 53.
A solutions architect must implement a redirect service that accepts HTTP and HTTPS requests.
Which combination of steps should the solutions architect take to meet these requirements with the LEAST amount of operational effort? (Choose three.)

A.Create a dynamic webpage that runs on an Amazon EC2 instance. Configure the webpage to use the JSON document in combination with the event message to look up and respond with a redirect URL.

B.Create an Application Load Balancer that includes HTTP and HTTPS listeners.

C.Create an AWS Lambda function that uses the JSON document in combination with the event message to look up and respond with a redirect URL.

D.Use an Amazon API Gateway API with a custom domain to publish an AWS Lambda function.

E.Create an Amazon CloudFront distribution. Deploy a function.

F.Create an SSL certificate by using AWS Certificate Manager (ACM). Include the domains as Subject Alternative Names.

Answer: CEF

Explanation:

Here's a detailed justification for choosing options C, E, and F for the redirect service implementation, along with supporting concepts and links:

The most efficient approach leverages serverless architecture and content delivery networks to minimize operational overhead. Option C, using an AWS Lambda function, is key. Lambda allows you to execute code without provisioning or managing servers. The Lambda function will parse the JSON document containing domain-to-URL mappings and use the incoming request's domain name to determine the appropriate redirect URL.

Option E, creating an Amazon CloudFront distribution and using a Lambda@Edge function, is crucial for performance and cost-effectiveness. CloudFront is a content delivery network (CDN) that caches content closer to users, reducing latency and improving the user experience. Lambda@Edge enables you to execute Lambda functions at CloudFront edge locations. In this scenario, the Lambda@Edge function intercepts the request before it reaches your origin (Lambda function in Option C). It checks the host header, consults your domain-to-URL mapping, and generates an HTTP redirect response directly from the edge location. This avoids invoking the origin Lambda function for every redirect, saving costs and further reducing latency.

Option F, creating an SSL certificate with AWS Certificate Manager (ACM) and including all domain names, is essential for secure HTTPS redirects. ACM provides free SSL/TLS certificates for use with AWS services like CloudFront. Using a wildcard certificate (if appropriate and cost-effective based on the number of domains and your security posture) might be suitable, but including all domains as Subject Alternative Names (SANs) is generally preferred for clarity and manageability. This ensures that the certificate is valid for all the domains, allowing CloudFront to serve HTTPS traffic securely.

Options A, B, and D introduce unnecessary complexity and operational overhead. Option A (EC2 instance) requires managing servers, patching, and scaling. Option B (Application Load Balancer) is overkill; a CDN is more suitable for simple redirects. Option D (API Gateway) is unnecessary because CloudFront can directly serve the redirects via Lambda@Edge, bypassing the need for an API gateway.

Therefore, CEF is the optimal combination. It offers a scalable, performant, and cost-effective solution with minimal operational burden. The other options would result in managing infrastructure or unnecessary layers of complexity.

Supporting Links:

AWS Lambda:https://aws.amazon.com/lambda/
Amazon CloudFront:https://aws.amazon.com/cloudfront/
Lambda@Edge:https://aws.amazon.com/lambda/edge/
AWS Certificate Manager (ACM):https://aws.amazon.com/certificate-manager/

Question: 29

A company that has multiple AWS accounts is using AWS Organizations. The company's AWS accounts host VPCs, Amazon EC2 instances, and containers.
The company's compliance team has deployed a security tool in each VPC where the company has deployments.
The security tools run on EC2 instances and send information to the AWS account that is dedicated for the compliance team.
The company has tagged all the compliance-related resources with a key of "costCenter" and a value or "compliance".
The company wants to identify the cost of the security tools that are running on the EC2 instances so that the company can charge the compliance team's AWS account. The cost calculation must be as accurate as possible. What should a solutions architect do to meet these requirements?

A.In the management account of the organization, activate the costCenter user-defined tag. Configure monthly AWS Cost and Usage Reports to save to an Amazon S3 bucket in the management account. Use the tag breakdown in the report to obtain the total cost for the costCenter tagged resources.

B.In the member accounts of the organization, activate the costCenter user-defined tag. Configure monthly AWS Cost and Usage Reports to save to an Amazon S3 bucket in the management account. Schedule a monthly AWS Lambda function to retrieve the reports and calculate the total cost for the costCenter tagged resources.

C.In the member accounts of the organization activate the costCenter user-defined tag. From the management account, schedule a monthly AWS Cost and Usage Report. Use the tag breakdown in the report to calculate the total cost for the costCenter tagged resources.

D.Create a custom report in the organization view in AWS Trusted Advisor. Configure the report to generate a monthly billing summary for the costCenter tagged resources in the compliance team's AWS account.

Answer: A

Explanation:

The correct answer is A because it provides the most accurate and centralized approach to cost allocation across multiple AWS accounts within an organization.

Here's a detailed justification:

Tag Activation: Activating the costCenter user-defined tag in the management account ensures that this tag is tracked across all member accounts for cost allocation purposes. Without activating the tag, the Cost and Usage Report will not be able to break down costs based on this tag.

Centralized Cost and Usage Reports: Configuring monthly AWS Cost and Usage Reports (CUR) to be saved to an Amazon S3 bucket in the management account provides a centralized repository for cost data across the entire organization. This eliminates the need to gather reports from individual member accounts, simplifying the analysis process. CUR provide the most granular and detailed billing information available in AWS.

Tag-Based Cost Breakdown: The CUR includes a tag breakdown, which allows you to filter and analyze costs based on specific tags, such as the costCenter tag in this case. This enables you to accurately determine the total cost associated with the compliance-related resources, specifically the EC2 instances running the security tools.

Accuracy: By using the CUR with tag breakdown, you achieve the most accurate cost calculation. The CUR includes details on all charges, including compute, storage, data transfer, and other AWS services utilized by the tagged resources.

Centralized Management: Managing the cost reporting and analysis from the management account provides a centralized view of costs and simplifies the process of allocating costs to the compliance team's AWS account.

Options B and C are less efficient because they involve activating the tag in each member account. While it works technically, it doesn't leverage the power of consolidated billing. Option B also involves Lambda function, which introduces unnecessary complexity as the CUR already provides the needed breakdown. Option D is incorrect because AWS Trusted Advisor does not directly provide custom billing summaries based on tags.

Refer to the AWS documentation on Cost and Usage Reports and Tagging for more information:

AWS Cost and Usage Reports:https://docs.aws.amazon.com/cur/latest/userguide/what-is-cur.html **Tagging AWS Resources:**https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/cost-alloc-tags.html
AWS Organizations Consolidated Billing:
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_getting-started_concepts.html#account

Question: 30

A company has 50 AWS accounts that are members of an organization in AWS Organizations. Each account contains multiple VPCs. The company wants to use AWS Transit Gateway to establish connectivity between the VPCs in each member account. Each time a new member account is created, the company wants to automate the process of creating a new VPC and a transit gateway attachment.
Which combination of steps will meet these requirements? (Choose two.)

A.From the management account, share the transit gateway with member accounts by using AWS Resource Access Manager.

B.From the management account, share the transit gateway with member accounts by using an AWS Organizations SCP.

C.Launch an AWS CloudFormation stack set from the management account that automatically creates a new VPC and a VPC transit gateway attachment in a member account. Associate the attachment with the transit gateway in the management account by using the transit gateway ID.

D.Launch an AWS CloudFormation stack set from the management account that automatically creates a new VPC and a peering transit gateway attachment in a member account. Share the attachment with the transit gateway in the management account by using a transit gateway service-linked role.

E.From the management account, share the transit gateway with member accounts by using AWS Service Catalog.

Answer: AC

Explanation:

The correct answer is AC. Here's why:

A. From the management account, share the transit gateway with member accounts by using AWS Resource Access Manager.

AWS Resource Access Manager (RAM) is the recommended way to share Transit Gateways across accounts within an AWS Organization. This allows the central Transit Gateway, which manages the network connections, to be used by VPCs in different accounts. Without sharing, each account would need its own Transit Gateway, increasing complexity and cost. RAM simplifies the process of granting access and maintains centralized control.
https://docs.aws.amazon.com/ram/latest/userguide/what-is.html

C. Launch an AWS CloudFormation stack set from the management account that automatically creates a new VPC and a VPC transit gateway attachment in a member account. Associate the attachment with the

transit gateway in the management account by using the transit gateway ID.

CloudFormation StackSets enable you to deploy CloudFormation stacks across multiple AWS accounts and regions from a single management account. This automates the creation of new VPCs and Transit Gateway attachments in each newly created member account. Specifying the Transit Gateway ID in the CloudFormation template ensures the new attachments connect to the central Transit Gateway instance. This automates the process of establishing connectivity when a new account is created, fulfilling the requirements of the question. https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/what-is-stacksets.html

Why other options are incorrect:

B: AWS Organizations SCPs (Service Control Policies) primarily manage permissions and guardrails within an organization. They do not share Transit Gateways.
D: Peering transit gateway attachment is not a valid resource and the approach of sharing via service-linked role isn't applicable.
E: AWS Service Catalog is used to create and manage catalogs of IT services approved for use in AWS. It's not designed for sharing Transit Gateways.

Question: 31

An enterprise company wants to allow its developers to purchase third-party software through AWS Marketplace. The company uses an AWS Organizations account structure with full features enabled, and has a shared services account in each organizational unit (OU) that will be used by procurement managers. The procurement team's policy indicates that developers should be able to obtain third-party software from an approved list only and use Private Marketplace in AWS Marketplace to achieve this requirement. The procurement team wants administration of Private Marketplace to be restricted to a role named procurement-manager-role, which could be assumed by procurement managers. Other IAM users, groups, roles, and account administrators in the company should be denied Private Marketplace administrative access.
What is the MOST efficient way to design an architecture to meet these requirements?

A.Create an IAM role named procurement-manager-role in all AWS accounts in the organization. Add the PowerUserAccess managed policy to the role. Apply an inline policy to all IAM users and roles in every AWS account to deny permissions on the AWSPrivateMarketplaceAdminFullAccess managed policy.

B.Create an IAM role named procurement-manager-role in all AWS accounts in the organization. Add the AdministratorAccess managed policy to the role. Define a permissions boundary with the AWSPrivateMarketplaceAdminFullAccess managed policy and attach it to all the developer roles.

C.Create an IAM role named procurement-manager-role in all the shared services accounts in the organization. Add the AWSPrivateMarketplaceAdminFullAccess managed policy to the role. Create an organization root-level SCP to deny permissions to administer Private Marketplace to everyone except the role named procurement-manager-role. Create another organization root-level SCP to deny permissions to create an IAM role named procurement-manager-role to everyone in the organization.

D.Create an IAM role named procurement-manager-role in all AWS accounts that will be used by developers. Add the AWSPrivateMarketplaceAdminFullAccess managed policy to the role. Create an SCP in Organizations to deny permissions to administer Private Marketplace to everyone except the role named procurement-manager-role. Apply the SCP to all the shared services accounts in the organization.

Answer: C

Explanation:

The most efficient solution is **C**. Here's why:

Centralized Control: AWS Organizations and Service Control Policies (SCPs) offer centralized governance across all accounts within the organization. This aligns with the enterprise's requirement for restricted administration.

Role-Based Access Control: Creating the procurement-manager-role in the shared services accounts and

assigning AWSPrivateMarketplaceAdminFullAccess ensures only procurement managers in designated accounts can administer the Private Marketplace.

Explicit Deny with SCPs: SCPs with explicit deny statements are crucial. The first SCP denies Private Marketplace administration to everyone at the root level of the organization. This provides a broad restriction, which is then selectively allowed through the procurement-manager-role. The second SCP restricts the ability to create the procurement-manager-role itself, mitigating the risk of unauthorized users assuming the role by creating it themselves.

Least Privilege: The AWSPrivateMarketplaceAdminFullAccess policy grants the minimum required permissions to manage the Private Marketplace, adhering to the principle of least privilege.

Why other options are incorrect:

A: Adding PowerUserAccess is overly permissive. Applying inline deny policies to all IAM users and roles is cumbersome and difficult to maintain across a large organization.

B:AdministratorAccess is far too broad and violates the principle of least privilege. Permissions boundaries restrict what roles can do, but don't prevent users from taking actions outside those roles. This doesn't effectively prevent unauthorized administration of the Private Marketplace.

D: Applying the SCP to shared services accounts is ineffective because the role granting permissions is also in the shared services accounts. The SCP needs to restrict access everywhere else. Furthermore, the procurement-manager-role is only needed in shared services accounts, not all developer accounts.

Authoritative Links:

AWS Organizations: https://aws.amazon.com/organizations/
Service Control Policies (SCPs): https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps.html
AWS Marketplace Private Marketplace: https://docs.aws.amazon.com/marketplace/latest/userguide/private-marketplace.html
IAM Roles: https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html

Question: 32

A company is in the process of implementing AWS Organizations to constrain its developers to use only Amazon EC2, Amazon S3, and Amazon DynamoDB. The developers account resides in a dedicated organizational unit (OU). The solutions architect has implemented the following SCP on the developers account:

```
{
        "Version": "2012-10-17",
        "Statement": [
                {
                        "Sid": "AllowEC2",
                        "Effect": "Allow",
                        "Action": "ec2:*",
                        "Resource": "*"
                },
                {
                        "Sid": "AllowDynamoDB",
                        "Effect": "Allow",
                        "Action": "dynamodb:*",
                        "Resource": "*"
                },
                {
                        "Sid": "AllowS3",
                        "Effect": "Allow",
                        "Action": "s3:*",
                        "Resource": "*"
                }
        ]
}
```

When this policy is deployed, IAM users in the developers account are still able to use AWS services that are not listed in the policy.
What should the solutions architect do to eliminate the developers' ability to use services outside the scope of this policy?

A.Create an explicit deny statement for each AWS service that should be constrained.

B.Remove the FullAWSAccess SCP from the developers account's OU.

C.Modify the FullAWSAccess SCP to explicitly deny all services.

D.Add an explicit deny statement using a wildcard to the end of the SCP.

Answer: B

Explanation:

Initially I voted for A but then I saw the following statement : "AWS services that aren't explicitly allowed by the SCPs associated with an AWS account or its parent OUs are denied access to the AWS accounts or OUs

associated with the SCP. SCPs associated to an OU are inherited by all AWS accounts in that OU"

B' is the BEST answer, but not the only correct one. 'D' is also technically correct, because adding a wildcard DENY statement would override the FullAWSAccess SCP attached by default to the OU and it would have the same final result. However 'B' is more appropriate here, the so called best practice. This is what 'Professional' exam certs are all about.

Question: 33

A company is hosting a monolithic REST-based API for a mobile app on five Amazon EC2 instances in public subnets of a VPC. Mobile clients connect to the API by using a domain name that is hosted on Amazon Route 53.
The company has created a Route 53 multivalue answer routing policy with the IP addresses of all the EC2 instances. Recently, the app has been overwhelmed by large and sudden increases to traffic. The app has not been able to keep up with the traffic.
A solutions architect needs to implement a solution so that the app can handle the new and varying load. Which solution will meet these requirements with the LEAST operational overhead?

A.Separate the API into individual AWS Lambda functions. Configure an Amazon API Gateway REST API with Lambda integration for the backend. Update the Route 53 record to point to the API Gateway API.

B.Containerize the API logic. Create an Amazon Elastic Kubernetes Service (Amazon EKS) cluster. Run the containers in the cluster by using Amazon EC2. Create a Kubernetes ingress. Update the Route 53 record to point to the Kubernetes ingress.

C.Create an Auto Scaling group. Place all the EC2 instances in the Auto Scaling group. Configure the Auto Scaling group to perform scaling actions that are based on CPU utilization. Create an AWS Lambda function that reacts to Auto Scaling group changes and updates the Route 53 record.

D.Create an Application Load Balancer (ALB) in front of the API. Move the EC2 instances to private subnets in the VPC. Add the EC2 instances as targets for the ALB. Update the Route 53 record to point to the ALB.

Answer: A

Explanation:

The best solution is **A. Separate the API into individual AWS Lambda functions. Configure an Amazon API Gateway REST API with Lambda integration for the backend. Update the Route 53 record to point to the API Gateway API.**

Here's why:

Scalability: Lambda functions scale automatically and independently based on the number of incoming requests. This provides the best elasticity for handling sudden and large traffic spikes. API Gateway can also handle a massive amount of concurrent API calls.

Operational Overhead: Lambda and API Gateway are serverless services. This significantly reduces operational overhead, as you don't need to manage servers, operating systems, or scaling infrastructure. The overhead of managing EC2 instances, EKS clusters, or even Auto Scaling groups is considerably higher.

Cost Efficiency: You only pay for the compute time consumed by Lambda functions when they are running.

API Gateway also has a pay-per-use pricing model. This can be more cost-effective than running dedicated EC2 instances, especially during periods of low traffic.

Integration: API Gateway integrates seamlessly with Lambda functions, making it easy to create and manage RESTful APIs. It also provides features like request validation, authentication, and authorization.

Route 53 integration: Updating the Route 53 record to point to the API Gateway endpoint is a straightforward process.

Here's why the other options are less suitable:

B (EKS): EKS is a powerful container orchestration platform, but it adds significant operational overhead. Managing a Kubernetes cluster requires expertise and effort, and it is not the least operationally heavy way to

handle the scalability requirements.

C (Auto Scaling Group): While Auto Scaling helps with scaling, it requires you to manage EC2 instances, which still involves operational overhead like patching, security updates, and instance sizing. Furthermore, the Lambda function needed to update Route 53 records when scaling events happen increases the complexity.

D (ALB): An ALB would improve availability and load distribution, but it still relies on EC2 instances that require ongoing management. It also might take time for the ALB to react to traffic changes and scale instances which will also require more operational overhead.

Authoritative Links:

AWS Lambda:https://aws.amazon.com/lambda/
Amazon API Gateway:https://aws.amazon.com/api-gateway/
Amazon Route 53:https://aws.amazon.com/route53/

Question: 34

A company has created an OU in AWS Organizations for each of its engineering teams. Each OU owns multiple AWS accounts. The organization has hundreds of AWS accounts.
A solutions architect must design a solution so that each OU can view a breakdown of usage costs across its AWS accounts.
Which solution meets these requirements?

A.Create an AWS Cost and Usage Report (CUR) for each OU by using AWS Resource Access Manager. Allow each team to visualize the CUR through an Amazon QuickSight dashboard.

B.Create an AWS Cost and Usage Report (CUR) from the AWS Organizations management account. Allow each team to visualize the CUR through an Amazon QuickSight dashboard.

C.Create an AWS Cost and Usage Report (CUR) in each AWS Organizations member account. Allow each team to visualize the CUR through an Amazon QuickSight dashboard.

D.Create an AWS Cost and Usage Report (CUR) by using AWS Systems Manager. Allow each team to visualize the CUR through Systems Manager OpsCenter dashboards.

Answer: B

Explanation:

The correct answer is B because it leverages the central management capabilities of AWS Organizations for cost reporting. Creating a single Cost and Usage Report (CUR) from the management account allows for consolidated billing and cost tracking across the entire organization, including all OUs and their associated accounts. This centralized approach simplifies administration and avoids the complexity of managing multiple CURs, as would be the case in options A and C. The CUR generated from the management account contains detailed cost information, which can then be visualized using Amazon QuickSight dashboards. These dashboards can be customized to provide each OU with a breakdown of its usage costs, meeting the requirement of the question.

Option A is incorrect because AWS Resource Access Manager (RAM) is designed for sharing AWS resources between accounts, not creating or managing cost reports. While RAM can share access to the CUR data, it does not solve the problem of generating a consolidated report for each OU. Option C is inefficient and difficult to manage because it would require creating and maintaining a CUR in each member account. This approach would create significant overhead and complexity. Option D is incorrect because AWS Systems Manager is designed for operations management and automation, not cost reporting or visualization. Systems Manager OpsCenter dashboards are not designed for cost analysis. Therefore, option B offers the most efficient and scalable solution to the problem by leveraging the centralized cost management capabilities of AWS Organizations and the visualization capabilities of Amazon QuickSight.

Further Reading:

AWS Cost and Usage Reports:https://docs.aws.amazon.com/cur/latest/userguide/what-is-cur.html **AWS Organizations:**https://aws.amazon.com/organizations/
Amazon QuickSight:https://aws.amazon.com/quicksight/

Question: 35

A company is storing data on premises on a Windows file server. The company produces 5 GB of new data daily. The company migrated part of its Windows-based workload to AWS and needs the data to be available on a file system in the cloud. The company already has established an AWS Direct Connect connection between the on-premises network and AWS.
Which data migration strategy should the company use?

A.Use the file gateway option in AWS Storage Gateway to replace the existing Windows file server, and point the existing file share to the new file gateway.

B.Use AWS DataSync to schedule a daily task to replicate data between the on-premises Windows file server and Amazon FSx.

C.Use AWS Data Pipeline to schedule a daily task to replicate data between the on-premises Windows file server and Amazon Elastic File System (Amazon EFS).

D.Use AWS DataSync to schedule a daily task to replicate data between the on-premises Windows file server and Amazon Elastic File System (Amazon EFS).

Answer: B

Explanation:

The most suitable data migration strategy is option B: using AWS DataSync to schedule a daily task to replicate data between the on-premises Windows file server and Amazon FSx. Here's why:

DataSync's suitability for the task: AWS DataSync is designed specifically for efficient and automated data transfer between on-premises storage and AWS storage services. It can handle the daily replication requirement effectively. (https://aws.amazon.com/datasync/)

FSx's appropriateness: Amazon FSx provides fully managed file systems that are compatible with Windows file servers, making it a natural fit for the on-premises Windows workload. (https://aws.amazon.com/fsx/)

Direct Connect Integration: The existing AWS Direct Connect connection provides the necessary bandwidth and dedicated network connectivity for reliable data transfer. DataSync is optimized to use the Direct Connect connection.

Incremental Replication: DataSync supports incremental replication, meaning it only transfers changed data, which is ideal for the daily 5GB of new data and minimizes bandwidth consumption.

File Gateway Limitation: Option A, using File Gateway, is more suitable when you want to replace your existing file server entirely. The scenario requires keeping the on-premises server and replicating data to AWS.

Data Pipeline is Not Optimal: Option C, using AWS Data Pipeline, is designed for more complex data workflows and transformations, which are not needed in this simple file replication scenario. It's an overkill for simple data migration. Additionally, it's now superseded by AWS Glue.

EFS Incompatibility: Option D uses DataSync with EFS. While technically possible, EFS isn't designed for native compatibility with Windows-based file servers and SMB protocol which the company presumably uses on-premise. FSx is a better fit for the on-premises Windows server.

Therefore, using AWS DataSync to schedule a daily task to replicate data between the on-premises Windows file server and Amazon FSx provides an efficient, automated, and compatible solution for the company's data migration needs, leveraging the existing Direct Connect connection.

Question: 36

A company's solutions architect is reviewing a web application that runs on AWS. The application references static assets in an Amazon S3 bucket in the us-east-1 Region. The company needs resiliency across multiple AWS Regions. The company already has created an S3 bucket in a second Region.
Which solution will meet these requirements with the LEAST operational overhead?

A.Configure the application to write each object to both S3 buckets. Set up an Amazon Route 53 public hosted zone with a record set by using a weighted routing policy for each S3 bucket. Configure the application to reference the objects by using the Route 53 DNS name.

B.Create an AWS Lambda function to copy objects from the S3 bucket in us-east-1 to the S3 bucket in the second Region. Invoke the Lambda function each time an object is written to the S3 bucket in us-east-1. Set up an Amazon CloudFront distribution with an origin group that contains the two S3 buckets as origins.

C.Configure replication on the S3 bucket in us-east-1 to replicate objects to the S3 bucket in the second Region. Set up an Amazon CloudFront distribution with an origin group that contains the two S3 buckets as origins.

D.Configure replication on the S3 bucket in us-east-1 to replicate objects to the S3 bucket in the second Region. If failover is required, update the application code to load S3 objects from the S3 bucket in the second Region.

Answer: C

Explanation:

The correct answer is C because it offers a highly resilient and automated solution with minimal operational overhead, leveraging native AWS services for replication and content delivery.

Here's a detailed justification:

S3 Replication: S3 Cross-Region Replication (CRR) automatically and asynchronously copies objects between S3 buckets in different AWS Regions. This ensures data redundancy and availability in the second Region without requiring custom code or manual intervention. This satisfies the resiliency requirement efficiently.
https://docs.aws.amazon.com/AmazonS3/latest/userguide/replication.html

CloudFront Origin Groups: Amazon CloudFront origin groups provide a mechanism for failover between origins. By configuring both S3 buckets as origins within an origin group, CloudFront can automatically switch to the secondary bucket if the primary bucket becomes unavailable. This ensures that the application continues to serve static assets even in the event of a regional outage.
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/origin-groups.html

Least Operational Overhead: This solution is highly automated. S3 replication manages data synchronization, and CloudFront manages failover. No custom code (like Lambda functions) or manual intervention is required, minimizing operational overhead.

Let's examine why the other options are less desirable:

Option A: Writing to both S3 buckets from the application requires code changes and adds latency to object uploads. Route 53 weighted routing can distribute traffic, but it doesn't guarantee immediate failover and may result in inconsistent data if writes aren't perfectly synchronized.

Option B: Using a Lambda function to copy objects adds complexity and potential points of failure. Lambda invocation for every object write can incur costs and latency. While CloudFront is helpful, the replication method is less efficient than S3's built-in replication.

Option D: Manually updating application code for failover is time-consuming and error-prone, increasing

operational burden. The recovery time objective (RTO) would be much higher compared to using CloudFront origin groups.

In conclusion, option C provides the most efficient and resilient solution by utilizing S3 replication for automatic data synchronization and CloudFront origin groups for seamless failover, minimizing operational overhead and ensuring high availability of static assets.

Question: 37

A company is hosting a three-tier web application in an on-premises environment. Due to a recent surge in traffic that resulted in downtime and a significant financial impact, company management has ordered that the application be moved to AWS. The application is written in .NET and has a dependency on a MySQL database. A solutions architect must design a scalable and highly available solution to meet the demand of 200,000 daily users.
Which steps should the solutions architect take to design an appropriate solution?

A.Use AWS Elastic Beanstalk to create a new application with a web server environment and an Amazon RDS MySQL Multi-AZ DB instance. The environment should launch a Network Load Balancer (NLB) in front of an Amazon EC2 Auto Scaling group in multiple Availability Zones. Use an Amazon Route 53 alias record to route traffic from the company's domain to the NLB.

B.Use AWS CloudFormation to launch a stack containing an Application Load Balancer (ALB) in front of an Amazon EC2 Auto Scaling group spanning three Availability Zones. The stack should launch a Multi-AZ deployment of an Amazon Aurora MySQL DB cluster with a Retain deletion policy. Use an Amazon Route 53 alias record to route traffic from the company's domain to the ALB.

C.Use AWS Elastic Beanstalk to create an automatically scaling web server environment that spans two separate Regions with an Application Load Balancer (ALB) in each Region. Create a Multi-AZ deployment of an Amazon Aurora MySQL DB cluster with a cross-Region read replica. Use Amazon Route 53 with a geoproximity routing policy to route traffic between the two Regions.

D.Use AWS CloudFormation to launch a stack containing an Application Load Balancer (ALB) in front of an Amazon ECS cluster of Spot instances spanning three Availability Zones. The stack should launch an Amazon RDS MySQL DB instance with a Snapshot deletion policy. Use an Amazon Route 53 alias record to route traffic from the company's domain to the ALB.

Answer: B

Explanation:

The correct answer is B because it provides a scalable, highly available, and cost-effective solution for migrating the .NET application and its MySQL database to AWS. Let's break down why:

AWS CloudFormation: Using CloudFormation allows for infrastructure-as-code, enabling repeatable and consistent deployments. This is crucial for managing complex environments and future updates.

Application Load Balancer (ALB): ALBs distribute incoming application traffic across multiple targets, such as EC2 instances or containers, in multiple Availability Zones. This ensures high availability and fault tolerance. https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html

Amazon EC2 Auto Scaling Group: An Auto Scaling group ensures that the number of EC2 instances matches the demand. It automatically adjusts capacity to maintain performance and availability, scaling out during peak times and scaling in during periods of low traffic. Spanning three Availability Zones increases resilience. https://docs.aws.amazon.com/autoscaling/ec2/userguide/AutoScalingGroup.html

Amazon Aurora MySQL Multi-AZ DB Cluster: Aurora MySQL offers significantly better performance than standard MySQL and is designed for high availability. The Multi-AZ deployment automatically replicates data to standby instances in other Availability Zones for failover protection. https://aws.amazon.com/rds/aurora/

Retain Deletion Policy: This policy prevents the database from being accidentally deleted when the

CloudFormation stack is deleted, ensuring data preservation.

Amazon Route 53 Alias Record: Using an alias record in Route 53 directly links the company's domain name to the ALB, providing a simplified and efficient way to route traffic.

https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/resource-record-sets-choosing-alias-non-alias.html

Option A is less ideal because Elastic Beanstalk, while simplifying deployment, offers less granular control compared to CloudFormation. An NLB is optimized for TCP traffic, whereas an ALB is more suitable for web applications due to its Layer 7 capabilities.

Option C introduces unnecessary complexity by using multi-region deployment with geoproximity routing and can increase costs. While redundancy is good, a properly configured Multi-AZ deployment is usually sufficient for most high availability requirements and is simpler to manage and more cost-effective initially.

Option D uses Spot Instances for the EC2 instances, which can be interrupted and are not suitable for a mission-critical application needing sustained availability. Furthermore, RDS MySQL with a Snapshot deletion policy poses a data loss risk upon stack deletion.

Therefore, option B provides the most appropriate balance of scalability, high availability, manageability, and cost-effectiveness for the specified requirements.

Question: 38

A company is using AWS Organizations to manage multiple AWS accounts. For security purposes, the company requires the creation of an Amazon Simple Notification Service (Amazon SNS) topic that enables integration with a third-party alerting system in all the Organizations member accounts.
A solutions architect used an AWS CloudFormation template to create the SNS topic and stack sets to automate the deployment of CloudFormation stacks. Trusted access has been enabled in Organizations.
What should the solutions architect do to deploy the CloudFormation StackSets in all AWS accounts?

A.Create a stack set in the Organizations member accounts. Use service-managed permissions. Set deployment options to deploy to an organization. Use CloudFormation StackSets drift detection.

B.Create stacks in the Organizations member accounts. Use self-service permissions. Set deployment options to deploy to an organization. Enable the CloudFormation StackSets automatic deployment.

C.Create a stack set in the Organizations management account. Use service-managed permissions. Set deployment options to deploy to the organization. Enable CloudFormation StackSets automatic deployment.

D.Create stacks in the Organizations management account. Use service-managed permissions. Set deployment options to deploy to the organization. Enable CloudFormation StackSets drift detection.

Answer: C

Explanation:

The correct answer is C. Here's a detailed justification:

To achieve automated deployment of the SNS topic across all AWS accounts within an AWS Organization, CloudFormation StackSets with service-managed permissions is the ideal approach. StackSets allows you to manage CloudFormation stacks across multiple AWS accounts and regions from a central management account.

Option A is incorrect because StackSets should be managed from the organization's management account, not from individual member accounts.

Option B is incorrect because StackSets should be managed from the organization's management account, not from individual member accounts. Also, when integrating with AWS Organizations, service-managed

permissions are the recommended approach for StackSets. Self-managed permissions require more manual configuration and delegation of roles within each member account, complicating management. CloudFormation StackSets automatic deployment feature is the correct approach here for automatic deployment of updates.

Option D is incorrect because StackSets are preferred over individual stacks for managing deployments across multiple accounts. Also, CloudFormation StackSets automatic deployment is the correct approach here for automatic deployment of updates. While drift detection is valuable for monitoring changes, it doesn't automate the initial deployment as effectively as the automatic deployment feature.

The correct approach, option C, involves creating the StackSet in the Organizations management account. Using service-managed permissions simplifies management, allowing StackSets to assume necessary roles in member accounts without manual role creation in each account. Setting deployment options to deploy to the organization ensures the SNS topic is deployed across all current and future member accounts. Enabling automatic deployment ensures that any updates to the StackSet are automatically propagated to all target accounts, maintaining consistency and reducing manual intervention. This solution adheres to AWS best practices for multi-account management using Organizations and CloudFormation.

Here are some authoritative links for further research:

AWS CloudFormation StackSets:
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/what-is-stacksets.html **Using AWS CloudFormation StackSets with Organizations:**
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacksets-orgs.html **Permissions models for StackSets:**
https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/stacksets-permissions.html

Question: 39

A company wants to migrate its workloads from on premises to AWS. The workloads run on Linux and Windows. The company has a large on-premises infrastructure that consists of physical machines and VMs that host numerous applications.

The company must capture details about the system configuration, system performance, running processes, and network connections of its on-premises workloads. The company also must divide the on-premises applications into groups for AWS migrations. The company needs recommendations for Amazon EC2 instance types so that the company can run its workloads on AWS in the most cost-effective manner.

Which combination of steps should a solutions architect take to meet these requirements? (Choose three.)

A.Assess the existing applications by installing AWS Application Discovery Agent on the physical machines and VMs.

B.Assess the existing applications by installing AWS Systems Manager Agent on the physical machines and VMs.

C.Group servers into applications for migration by using AWS Systems Manager Application Manager.

D.Group servers into applications for migration by using AWS Migration Hub.

E.Generate recommended instance types and associated costs by using AWS Migration Hub.

F.Import data about server sizes into AWS Trusted Advisor. Follow the recommendations for cost optimization.

Answer: ADE

Explanation:

The correct answer is ADE. Here's why:

A. Assess the existing applications by installing AWS Application Discovery Agent on the physical

machines and VMs. AWS Application Discovery Service (ADS) is the primary tool for gathering detailed information about on-premises workloads, including system configuration, performance metrics, running processes, and network dependencies. The agentless discovery collector (or the agent based AWS Agent) can be used for more in-depth details. This aligns perfectly with the requirement to capture comprehensive details about the on-premises environment.

AWS Application Discovery Service

D. Group servers into applications for migration by using AWS Migration Hub. AWS Migration Hub is designed to track the progress of application migrations to AWS. It allows you to group servers into applications to simplify the migration process and track the overall status of each application. This satisfies the requirement to divide on-premises applications into groups.

AWS Migration Hub

E. Generate recommended instance types and associated costs by using AWS Migration Hub. After discovering the on-premises environment and grouping the applications, AWS Migration Hub can provide recommendations for EC2 instance types and associated costs based on the collected data. It helps to ensure that the migrated workloads run on AWS in a cost-effective manner, as the requirement specifies.

AWS Migration Hub Instance Recommendations

Why the other options are less suitable:

B. Assess the existing applications by installing AWS Systems Manager Agent on the physical machines and VMs. While AWS Systems Manager Agent can gather information about instances, it's not primarily designed for discovering and profiling on-premises environments for migration purposes. Application Discovery Service is the more appropriate tool.

C. Group servers into applications for migration by using AWS Systems Manager Application Manager. AWS Systems Manager Application Manager focuses on operational management and visibility of applications after they've been migrated to AWS, not the initial grouping and discovery for migration planning.

F. Import data about server sizes into AWS Trusted Advisor. Follow the recommendations for cost optimization. AWS Trusted Advisor is a tool for cost optimization, security, fault tolerance and performance improvement on AWS resources. It does not help with the initial discovery and gathering of information from on-premises environments required for planning a migration.

Question: 40

A company is hosting an image-processing service on AWS in a VPC. The VPC extends across two Availability Zones. Each Availability Zone contains one public subnet and one private subnet.

The service runs on Amazon EC2 instances in the private subnets. An Application Load Balancer in the public subnets is in front of the service. The service needs to communicate with the internet and does so through two NAT gateways. The service uses Amazon S3 for image storage. The EC2 instances retrieve approximately 1 TB of data from an S3 bucket each day.

The company has promoted the service as highly secure. A solutions architect must reduce cloud expenditures as much as possible without compromising the service's security posture or increasing the time spent on ongoing operations.

Which solution will meet these requirements?

A.Replace the NAT gateways with NAT instances. In the VPC route table, create a route from the private subnets to the NAT instances.

B.Move the EC2 instances to the public subnets. Remove the NAT gateways.

C.Set up an S3 gateway VPC endpoint in the VPAttach an endpoint policy to the endpoint to allow the required actions on the S3 bucket.

D.Attach an Amazon Elastic File System (Amazon EFS) volume to the EC2 instances. Host the images on the

EFS volume.

Answer: C

Explanation:

The most cost-effective and secure solution is to implement an S3 gateway VPC endpoint (Option C). Here's why:

Cost Reduction: NAT Gateways charge for data processed. Since the EC2 instances transfer 1 TB of data daily from S3, this generates significant NAT gateway costs. S3 gateway endpoints provide free connectivity to S3 within the VPC.

Security: S3 gateway endpoints allow EC2 instances to access S3 without traversing the public internet, enhancing security. Traffic stays within the AWS network. An endpoint policy can restrict access to specific S3 buckets and actions, further bolstering security.

Performance: S3 gateway endpoints are highly available and scalable. By using a gateway endpoint, you eliminate network hops and bottlenecks associated with NAT gateways, potentially improving data transfer speeds.

Simplicity: Once configured, the gateway endpoint requires minimal ongoing operational effort. The route table entries are managed automatically.

Option A (NAT Instances) is less desirable because NAT instances require manual management, patching, and scaling. They introduce a single point of failure, and they can be more expensive than S3 gateway endpoints, particularly at higher data transfer volumes.

Option B (moving EC2 instances to public subnets) significantly compromises security. It exposes the EC2 instances directly to the internet, which is generally undesirable for security best practices. Removing NAT gateways is not an option in this scenario given the original configuration requirements.

Option D (Amazon EFS) is not suitable because it would require migrating the existing data from S3 to EFS, which is time-consuming. Moreover, EFS is typically more expensive than S3 for large-scale object storage, and might not be designed for image processing at the scale provided.

Therefore, leveraging an S3 gateway VPC endpoint addresses the problem statement's constraints by providing a secure, cost-effective, and operationally simple solution for accessing S3 from within the VPC.

Authoritative Links:

VPC Endpoints:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html
S3 Gateway Endpoints:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints-s3.html **VPC Pricing:**https://aws.amazon.com/vpc/pricing/

Question: 41

A company recently deployed an application on AWS. The application uses Amazon DynamoDB. The company measured the application load and configured the RCUs and WCUs on the DynamoDB table to match the expected peak load. The peak load occurs once a week for a 4-hour period and is double the average load. The application load is close to the average load for the rest of the week. The access pattern includes many more writes to the table than reads of the table.

A solutions architect needs to implement a solution to minimize the cost of the table.

Which solution will meet these requirements?

A.Use AWS Application Auto Scaling to increase capacity during the peak period. Purchase reserved RCUs and

WCUs to match the average load.

B.Configure on-demand capacity mode for the table.

C.Configure DynamoDB Accelerator (DAX) in front of the table. Reduce the provisioned read capacity to match the new peak load on the table.

D.Configure DynamoDB Accelerator (DAX) in front of the table. Configure on-demand capacity mode for the table.

Answer: A

Explanation:

The correct answer is A. Here's why:

The problem focuses on cost optimization for a DynamoDB table with a predictable weekly peak load that significantly exceeds the average load, especially concerning writes. The access pattern is write-heavy, making write capacity units (WCUs) a primary cost driver.

Option A (Use AWS Application Auto Scaling and reserved capacity) directly addresses this scenario. AWS Application Auto Scaling allows DynamoDB to automatically adjust the provisioned RCUs and WCUs based on real-time demand. During the 4-hour peak, Auto Scaling will increase capacity to handle the increased load. Purchasing reserved capacity for the average load provides a cost-effective baseline for the remaining time, avoiding the higher on-demand costs. Reserved capacity offers significant discounts compared to on-demand pricing when you have predictable, consistent usage.

Option B (Configure on-demand capacity mode) isn't optimal. On-demand capacity mode eliminates the need for capacity planning but charges for actual reads and writes. While it handles fluctuating workloads well, it's generally more expensive than provisioned capacity with Auto Scaling when you have predictable periods of high usage, as stated in the problem. Since the peak is predictable, you can utilize provisioned capacity with auto scaling to manage the traffic during peak times and not use on-demand, which charges every time you have an event.

Option C and D (Configure DynamoDB Accelerator (DAX)) is incorrect and DAX is suitable to improve the read performance with cached data and not optimal for optimizing costs in write heavy application. It can also adds complexity and cost. Further, the peak is related to the writes, therefore DAX is not applicable in this situation. Reducing provisioned read capacity is impractical for the peak periods as it may result in throttling.

In summary, combining reserved capacity for the average load with Auto Scaling to handle the peak load provides the most cost-effective solution for a predictable, fluctuating workload in DynamoDB. This minimizes wasted capacity during off-peak hours while ensuring sufficient resources during the peak period.

Supporting Documentation:

AWS DynamoDB Auto Scaling:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/AutoScaling.html
DynamoDB Reserved Capacity:https://aws.amazon.com/dynamodb/pricing/provisioned/
DynamoDB On-Demand Capacity:https://aws.amazon.com/dynamodb/pricing/on-demand/

Question: 42

A solutions architect needs to advise a company on how to migrate its on-premises data processing application to the AWS Cloud. Currently, users upload input files through a web portal. The web server then stores the uploaded files on NAS and messages the processing server over a message queue. Each media file can take up to 1 hour to process. The company has determined that the number of media files awaiting processing is significantly higher during business hours, with the number of files rapidly declining after business hours.

What is the MOST cost-effective migration recommendation?

A.Create a queue using Amazon SQS. Configure the existing web server to publish to the new queue. When there are messages in the queue, invoke an AWS Lambda function to pull requests from the queue and process the files. Store the processed files in an Amazon S3 bucket.

B.Create a queue using Amazon MQ. Configure the existing web server to publish to the new queue. When there are messages in the queue, create a new Amazon EC2 instance to pull requests from the queue and process the files. Store the processed files in Amazon EFS. Shut down the EC2 instance after the task is complete.

C.Create a queue using Amazon MQ. Configure the existing web server to publish to the new queue. When there are messages in the queue, invoke an AWS Lambda function to pull requests from the queue and process the files. Store the processed files in Amazon EFS.

D.Create a queue using Amazon SQS. Configure the existing web server to publish to the new queue. Use Amazon EC2 instances in an EC2 Auto Scaling group to pull requests from the queue and process the files. Scale the EC2 instances based on the SQS queue length. Store the processed files in an Amazon S3 bucket.

Answer: D

Explanation:

The correct answer is D because it provides the most cost-effective and scalable solution for migrating the data processing application to AWS. Here's a detailed justification:

SQS for Messaging: Amazon SQS (Simple Queue Service) is a fully managed message queuing service, ideal for decoupling the web server from the processing servers. This offers reliability and scalability.

https://aws.amazon.com/sqs/

Auto Scaling Group: Utilizing an EC2 Auto Scaling group enables dynamic scaling of processing capacity based on the number of messages in the SQS queue. During peak business hours when the queue length increases, the Auto Scaling group automatically launches more EC2 instances to handle the workload. Conversely, during off-peak hours, instances are terminated to reduce costs.

Cost Efficiency: The dynamic scaling inherent in using EC2 Auto Scaling based on queue length offers optimal cost efficiency. You only pay for the processing capacity needed at any given time.

EC2 for Processing: EC2 instances provide the computational power necessary to process the media files, which can take up to an hour each.

S3 for Storage: Amazon S3 (Simple Storage Service) provides durable, scalable, and cost-effective object storage for the processed files. https://aws.amazon.com/s3/

Why other options are less ideal:

Option A (Lambda): While Lambda is serverless, it has execution time limits (currently 15 minutes). Media files can take up to 1 hour to process which exceeds Lambda's limitation.

Option B (Amazon MQ, EC2, EFS): Amazon MQ is a managed message broker service for ActiveMQ and RabbitMQ, which is more suited for migrating existing applications that rely on these brokers. SQS is a simpler, fully managed queueing solution. EFS (Elastic File System) is suitable for shared file storage, but S3 is more cost-effective for storing processed files. Shutting down the EC2 instance after each task requires more management than auto-scaling.

Option C (Amazon MQ, Lambda, EFS): Similar to option A, Lambda's execution time limit makes it unsuitable for processing media files that take up to an hour. Using EFS instead of S3 is less cost-effective for storing the processed files.

In summary, Option D provides the best balance of scalability, cost-effectiveness, and manageability for migrating the data processing application to AWS by utilizing SQS for decoupling, EC2 Auto Scaling for dynamic processing capacity, and S3 for durable storage.

Question: 43

A company is using Amazon OpenSearch Service to analyze data. The company loads data into an OpenSearch

Service cluster with 10 data nodes from an Amazon S3 bucket that uses S3 Standard storage. The data resides in the cluster for 1 month for read-only analysis. After 1 month, the company deletes the index that contains the data from the cluster. For compliance purposes, the company must retain a copy of all input data.

The company is concerned about ongoing costs and asks a solutions architect to recommend a new solution.

Which solution will meet these requirements MOST cost-effectively?

A.Replace all the data nodes with UltraWarm nodes to handle the expected capacity. Transition the input data from S3 Standard to S3 Glacier Deep Archive when the company loads the data into the cluster.

B.Reduce the number of data nodes in the cluster to 2 Add UltraWarm nodes to handle the expected capacity. Configure the indexes to transition to UltraWarm when OpenSearch Service ingests the data. Transition the input data to S3 Glacier Deep Archive after 1 month by using an S3 Lifecycle policy.

C.Reduce the number of data nodes in the cluster to 2. Add UltraWarm nodes to handle the expected capacity. Configure the indexes to transition to UltraWarm when OpenSearch Service ingests the data. Add cold storage nodes to the cluster Transition the indexes from UltraWarm to cold storage. Delete the input data from the S3 bucket after 1 month by using an S3 Lifecycle policy.

D.Reduce the number of data nodes in the cluster to 2. Add instance-backed data nodes to handle the expected capacity. Transition the input data from S3 Standard to S3 Glacier Deep Archive when the company loads the data into the cluster.

Answer: B

Explanation:

The correct answer is B because it offers the most cost-effective solution for the given requirements of data analysis, retention, and compliance.

Option B optimizes cost in several ways:

1. **Reduces Hot Storage:** By reducing the number of data nodes (which are expensive "hot" storage), the upfront cost of the OpenSearch cluster is lowered.
2. **Utilizes UltraWarm:** Adding UltraWarm nodes provides a cost-effective way to store the data for analysis, as UltraWarm uses S3 as its backing store, offering significantly cheaper storage than dedicated data nodes. UltraWarm is designed for read-only analysis, fulfilling the requirement.

https://aws.amazon.com/blogs/database/amazon-opensearch-service-unveils-ultrawarm-a-new-warm-storage-tier/

3. **S3 Glacier Deep Archive for Compliance:** Transitioning the input data to S3 Glacier Deep Archive after one month using an S3 Lifecycle policy ensures the data is retained for compliance purposes at the lowest possible storage cost. S3 Glacier Deep Archive is designed for long-term archival.

https://aws.amazon.com/glacier/deep-archive/https://docs.aws.amazon.com/AmazonS3/latest/userguide/lifecycle-configuration-examples.html

4. **Automated Transition:** The S3 Lifecycle policy automates the transition to Glacier Deep Archive, minimizing operational overhead.

Option A is less cost-effective because transitioning the input data to S3 Glacier Deep Archive immediately means the OpenSearch cluster needs to access Glacier Deep Archive to ingest data which is much slower and potentially more expensive than initially using S3 Standard. This is less suitable for the initial month of analysis.

Option C introduces Cold storage which, while cost-effective, necessitates another layer of complexity and data movement.

Option D uses instance-backed storage which is generally less cost-effective for long term storage compared to UltraWarm or S3 Glacier. Using S3 Glacier Deep Archive upfront also shares same issue with option A.

In summary, Option B strikes the best balance between performance during the active analysis period and cost-effective storage for long-term compliance, by reducing the hot storage requirements, leveraging UltraWarm for analysis, and then transitioning data to S3 Glacier Deep Archive for long-term, low-cost storage after analysis.

Question: 44

A company has 10 accounts that are part of an organization in AWS Organizations. AWS Config is configured in each account. All accounts belong to either the Prod OU or the NonProd OU.

The company has set up an Amazon EventBridge rule in each AWS account to notify an Amazon Simple Notification Service (Amazon SNS) topic when an Amazon EC2 security group inbound rule is created with 0.0.0.0/0 as the source. The company's security team is subscribed to the SNS topic.

For all accounts in the NonProd OU, the security team needs to remove the ability to create a security group inbound rule that includes 0.0.0.0/0 as the source.

Which solution will meet this requirement with the LEAST operational overhead?

A.Modify the EventBridge rule to invoke an AWS Lambda function to remove the security group inbound rule and to publish to the SNS topic. Deploy the updated rule to the NonProd OU.

B.Add the vpc-sg-open-only-to-authorized-ports AWS Config managed rule to the NonProd OU.

C.Configure an SCP to allow the ec2:AuthorizeSecurityGroupIngress action when the value of the aws:SourceIp condition key is not 0.0.0.0/0. Apply the SCP to the NonProd OU.

D.Configure an SCP to deny the ec2:AuthorizeSecurityGroupIngress action when the value of the aws:SourceIp condition key is 0.0.0.0/0. Apply the SCP to the NonProd OU.

Answer: D

Explanation:

Here's a detailed justification for why option D is the best solution, along with supporting concepts and authoritative links:

The requirement is to prevent the creation of EC2 security group inbound rules with 0.0.0.0/0 as the source (effectively opening the port to the world) in the NonProd OU with the least operational overhead.

Option D uses a Service Control Policy (SCP) to deny the ec2:AuthorizeSecurityGroupIngress action when the condition key aws:SourceIp has a value of 0.0.0.0/0. This is the most effective and efficient way to centrally enforce this restriction at the OU level. SCPs act as guardrails, preventing actions from being taken regardless of the IAM permissions granted within the accounts in the OU. This ensures consistent enforcement across all accounts in the NonProd OU. The administrative burden is low because the SCP is defined once and applied at the OU level.

Option A suggests modifying the EventBridge rule to invoke a Lambda function to remove the security group rule and publish to the SNS topic. This is reactive, not preventative. It allows the rule to be created and then attempts to remediate it. This approach also requires writing and managing Lambda code, increasing operational complexity. Moreover, EventBridge rules trigger after an event occurs, making this a detection and remediation approach, rather than prevention.

Option B proposes adding the vpc-sg-open-only-to-authorized-ports AWS Config managed rule to the NonProd OU. This is also reactive and not preventative. AWS Config rules evaluate resources against desired configurations and report on non-compliance. While it can identify security groups that violate the rule, it doesn't actively prevent their creation. Like Option A, it addresses the problem after the rule is created.

Option C suggests configuring an SCP to allow the ec2:AuthorizeSecurityGroupIngress action only when

aws:SourceIp is not 0.0.0.0/0. This is the logical inverse of option D but has a crucial flaw. Allow policies in SCPs are ineffective because if any SCP denies an action, the action is denied, regardless of any allow policies. This approach effectively removes the ability to create any security group ingress rule, breaking necessary functionality.

Therefore, Option D is the most appropriate because it directly prevents the prohibited action using SCPs, offering the least operational overhead and the strongest security posture.

Supporting Links:

AWS Organizations SCPs:
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scp.html
AWS Organizations Policy Types:
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies.html
EC2 AuthorizeSecurityGroupIngress API:
https://docs.aws.amazon.com/AWSEC2/latest/APIReference/API_AuthorizeSecurityGroupIngress.html **AWS Condition Keys:**https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_condition-keys.html

Question: 45

A company hosts a Git repository in an on-premises data center. The company uses webhooks to invoke functionality that runs in the AWS Cloud. The company hosts the webhook logic on a set of Amazon EC2 instances in an Auto Scaling group that the company set as a target for an Application Load Balancer (ALB). The Git server calls the ALB for the configured webhooks. The company wants to move the solution to a serverless architecture.

Which solution will meet these requirements with the LEAST operational overhead?

A.For each webhook, create and configure an AWS Lambda function URL. Update the Git servers to call the individual Lambda function URLs.

B.Create an Amazon API Gateway HTTP API. Implement each webhook logic in a separate AWS Lambda function. Update the Git servers to call the API Gateway endpoint.

C.Deploy the webhook logic to AWS App Runner. Create an ALB, and set App Runner as the target. Update the Git servers to call the ALB endpoint.

D.Containerize the webhook logic. Create an Amazon Elastic Container Service (Amazon ECS) cluster, and run the webhook logic in AWS Fargate. Create an Amazon API Gateway REST API, and set Fargate as the target. Update the Git servers to call the API Gateway endpoint.

Answer: B

Explanation:

The best solution for moving the on-premises Git webhook logic to a serverless architecture with the least operational overhead is option B: using Amazon API Gateway HTTP API and Lambda functions.

Here's why:

Serverless: Both API Gateway and Lambda are fully managed serverless services. This eliminates the need to manage servers, operating systems, scaling, and patching, significantly reducing operational overhead.

Scalability: API Gateway and Lambda automatically scale to handle the incoming webhook requests. You don't have to provision or manage scaling policies.

Cost-Effectiveness: You only pay for the actual usage of API Gateway and Lambda, leading to potential cost savings compared to running EC2 instances or container-based solutions.

Simplicity: Mapping webhook URLs to Lambda functions through API Gateway is a relatively straightforward

process. API Gateway acts as the entry point, routing requests to the corresponding Lambda functions based on the webhook path.

Each webhook logic is encapsulated within its own Lambda function, promoting modularity and maintainability.

Other options are less optimal:

A (Lambda function URLs): Lambda function URLs are a simple way to invoke a single Lambda function but lack the advanced features of API Gateway such as request validation, throttling, and custom domain names. It will require more configuration and management.

C (AWS App Runner with ALB): While App Runner simplifies container deployment, it still involves managing a container image and configuring an ALB. This adds operational complexity compared to a purely serverless approach.

D (ECS Fargate with API Gateway REST API): ECS Fargate also involves containerization, which adds operational overhead. Creating a REST API with Fargate requires more configuration than using API Gateway's HTTP API with Lambda, which is designed for lightweight, serverless applications.

Using API Gateway HTTP API and Lambda functions offers a balanced solution by providing serverless, scalable, and cost-effective architecture with minimal operational overhead for handling Git webhooks.

Supporting links:

Amazon API Gateway
AWS Lambda
API Gateway HTTP APIs vs REST APIs

Question: 46

A company is planning to migrate 1,000 on-premises servers to AWS. The servers run on several VMware clusters in the company's data center. As part of the migration plan, the company wants to gather server metrics such as CPU details, RAM usage, operating system information, and running processes. The company then wants to query and analyze the data.

Which solution will meet these requirements?

A.Deploy and configure the AWS Agentless Discovery Connector virtual appliance on the on-premises hosts. Configure Data Exploration in AWS Migration Hub. Use AWS Glue to perform an ETL job against the data. Query the data by using Amazon S3 Select.

B.Export only the VM performance information from the on-premises hosts. Directly import the required data into AWS Migration Hub. Update any missing information in Migration Hub. Query the data by using Amazon QuickSight.

C.Create a script to automatically gather the server information from the on-premises hosts. Use the AWS CLI to run the put-resource-attributes command to store the detailed server data in AWS Migration Hub. Query the data directly in the Migration Hub console.

D.Deploy the AWS Application Discovery Agent to each on-premises server. Configure Data Exploration in AWS Migration Hub. Use Amazon Athena to run predefined queries against the data in Amazon S3.

Answer: D

Explanation:

The correct answer is D. Here's why:

Why Option D is Correct:

AWS Application Discovery Agent: This agent is designed specifically to collect detailed server metrics (CPU,

RAM, OS, processes) directly from each server. This fulfills the requirement of gathering comprehensive information. It is more effective than agentless methods because it can discover running processes and other details impossible to obtain without direct OS access.

Data Exploration in Migration Hub: Migration Hub's Data Exploration feature provides a centralized view of the discovered data. This feature allows you to organize and categorize the data collected by the agents. This is essential for querying and analysis.

Amazon Athena: Athena is a serverless query service that enables analysis of data stored in Amazon S3 using SQL. The discovered data from Migration Hub is made available in S3. Athena fits the need for querying and analyzing the collected server metrics using predefined or custom queries.

Why Other Options are Incorrect:

Option A: AWS Agentless Discovery Connector is more suitable for identifying server inventory, not detailed performance metrics like running processes and RAM usage. While Glue and S3 Select could be used, they introduce unnecessary complexity compared to Athena.

Option B: Directly importing VM performance information into Migration Hub is insufficient. The requirement includes gathering detailed information such as running processes and OS information. Also, it assumes that you have these metrics readily available and formatted.

Option C: While put-resource-attributes can store data in Migration Hub, manually scripting data collection and using the CLI for each server is inefficient for 1,000 servers. The Migration Hub console isn't designed for in-depth data analysis; it's primarily for visualization and basic filtering.

In summary: Option D offers the most efficient and comprehensive solution for gathering, storing, and analyzing detailed server metrics, aligning with the requirements of the migration plan. The Application Discovery Agent provides in-depth data collection, Migration Hub facilitates data organization, and Athena allows for flexible and scalable querying of the collected data.

Supporting Links:

AWS Application Discovery Service:https://aws.amazon.com/application-discovery/ **AWS Migration Hub:**https://aws.amazon.com/migration-hub/
Amazon Athena:https://aws.amazon.com/athena/

Question: 47

A company is building a serverless application that runs on an AWS Lambda function that is attached to a VPC. The company needs to integrate the application with a new service from an external provider. The external provider supports only requests that come from public IPv4 addresses that are in an allow list.

The company must provide a single public IP address to the external provider before the application can start using the new service.

Which solution will give the application the ability to access the new service?

A.Deploy a NAT gateway. Associate an Elastic IP address with the NAT gateway. Configure the VPC to use the NAT gateway.

B.Deploy an egress-only internet gateway. Associate an Elastic IP address with the egress-only internet gateway. Configure the elastic network interface on the Lambda function to use the egress-only internet gateway.

C.Deploy an internet gateway. Associate an Elastic IP address with the internet gateway. Configure the Lambda function to use the internet gateway.

D.Deploy an internet gateway. Associate an Elastic IP address with the internet gateway. Configure the default route in the public VPC route table to use the internet gateway.

Answer: A

Explanation:

The correct answer is A: Deploy a NAT gateway. Associate an Elastic IP address with the NAT gateway. Configure the VPC to use the NAT gateway.

Here's why:

The scenario requires a Lambda function within a VPC to access an external service that requires a specific, allow-listed public IP address. Lambda functions deployed inside a VPC do not have direct public internet access by default.

NAT Gateway: A NAT (Network Address Translation) gateway allows instances in a private subnet to connect to the internet or other AWS services but prevents the internet from initiating a connection with those instances. This aligns perfectly with the requirement of accessing an external service without allowing inbound connections.

Elastic IP Address: An Elastic IP address is a static, public IPv4 address designed for dynamic cloud computing. By associating an Elastic IP with the NAT gateway, the company obtains a consistent public IP address that can be provided to the external service provider.

VPC Configuration: Configuring the VPC to use the NAT gateway ensures that all outbound traffic from the Lambda function, destined for the internet, is routed through the NAT gateway. This means the source IP address will be the Elastic IP associated with the NAT gateway, fulfilling the external provider's requirement.

Let's examine why the other options are incorrect:

B. Egress-Only Internet Gateway: Egress-only internet gateways are designed for IPv6 traffic and do not support associating with an Elastic IP. They only allow outbound communication from the VPC, specifically for IPv6, which does not solve the IPv4 public IP requirement.

C. Internet Gateway: While an Internet Gateway allows resources in the VPC to access the internet, directly attaching a Lambda function to an Internet Gateway is not a typical or recommended practice. Lambda functions in a VPC are usually placed in private subnets. Moreover, you can't directly associate an Elastic IP with a Lambda function's network interface in the same way you can with a NAT gateway.

D. Internet Gateway with Public Route Table: Although an Internet Gateway is necessary for enabling internet connectivity to a VPC, simply associating it with a public route table does not address the problem.

The Lambda function resides within the private subnet, and providing general internet access doesn't inherently give the Lambda function a specific, controlled, and static public IP address to be whitelisted by the third-party provider. Furthermore, this approach introduces security risks by potentially exposing the Lambda function to unsolicited inbound traffic.

Therefore, the NAT gateway solution provides the required static public IP address, allows outbound-only access, and maintains a secure configuration for the Lambda function within the VPC.

Authoritative Links:

NAT Gateway:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html
Elastic IP Addresses:https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html **AWS Lambda and VPCs:**https://docs.aws.amazon.com/lambda/latest/dg/configuration-vpc.html

Question: 48

A solutions architect has developed a web application that uses an Amazon API Gateway Regional endpoint and an AWS Lambda function. The consumers of the web application are all close to the AWS Region where the application will be deployed. The Lambda function only queries an Amazon Aurora MySQL database. The solutions architect has configured the database to have three read replicas.

During testing, the application does not meet performance requirements. Under high load, the application opens a large number of database connections. The solutions architect must improve the application's performance.

Which actions should the solutions architect take to meet these requirements? (Choose two.)

A.Use the cluster endpoint of the Aurora database.

B.Use RDS Proxy to set up a connection pool to the reader endpoint of the Aurora database.

C.Use the Lambda Provisioned Concurrency feature.

D.Move the code for opening the database connection in the Lambda function outside of the event handler.

E.Change the API Gateway endpoint to an edge-optimized endpoint.

Answer: BD

Explanation:

The correct answer is **BD**. Here's why:

B. Use RDS Proxy to set up a connection pool to the reader endpoint of the Aurora database.

Problem Addressed: The primary bottleneck is the excessive number of database connections opened by the Lambda function. Establishing a new connection for each Lambda invocation (or a significant portion of invocations under high load) is resource-intensive and quickly exhausts available database connections. **RDS Proxy Solution:** Amazon RDS Proxy sits between the Lambda function and the Aurora database, creating and managing a pool of database connections. The Lambda function interacts with the proxy, which reuses existing connections from the pool for multiple function invocations. This significantly reduces the overhead of creating new connections and improves overall performance.

Reader Endpoint: Using the reader endpoint allows the proxy to distribute read requests across the read replicas, further optimizing performance by offloading read operations from the primary Aurora instance. **Connection pooling efficiency:** RDS Proxy is designed for serverless environments like Lambda, specifically to address connection limitations to relational databases. It reduces database load, minimizes latency, and improves application scalability.

D. Move the code for opening the database connection in the Lambda function outside of the event handler.

Problem Addressed: Lambda functions operate on a stateless, event-driven model. If the database connection code is within the handler function, a new connection is potentially established with each invocation.

Connection Reuse: Moving the database connection initialization outside the handler (into the global scope of the Lambda function) allows the Lambda execution environment to reuse the established connection across multiple invocations (within the same execution environment). This significantly reduces the overhead of repeatedly creating and destroying database connections.

Performance Improvement: Reusing database connections improves efficiency and reduces latency, especially under high load.

Cold Starts and Optimization: This optimization specifically targets the connection overhead. It can be implemented by instantiating a database connection outside the handler function, allowing it to persist between invocations if the same execution environment is reused. This reduces latency, especially during initial invocations or cold starts.

Why other options are not as effective:

A. Use the cluster endpoint of the Aurora database. While using the cluster endpoint is good practice for high availability, it doesn't directly address the connection pooling issue. It provides failover capabilities but doesn't reduce the number of connections opened.

C. Use the Lambda Provisioned Concurrency feature. Provisioned concurrency ensures that a specified

number of Lambda function instances are initialized and ready to respond to requests. While it can improve latency, it can also exacerbate the connection pool issue if each concurrent instance still opens its own database connection.

E. Change the API Gateway endpoint to an edge-optimized endpoint. Changing the API Gateway endpoint to an edge-optimized one is relevant for distributing API traffic globally through CloudFront. This approach will not address the excessive number of connections to the database server, which is the performance bottleneck. It primarily focuses on reducing latency for geographically dispersed users.

Supporting Documentation:

Amazon RDS Proxy:https://aws.amazon.com/rds/proxy/
Optimize Lambda connections to RDS:https://aws.amazon.com/blogs/compute/using-amazon-rds-proxy-with-aws-lambda/
Lambda Execution Environment:https://docs.aws.amazon.com/lambda/latest/dg/services-rds-tutorial.html

Question: 49

A company is planning to host a web application on AWS and wants to load balance the traffic across a group of Amazon EC2 instances. One of the security requirements is to enable end-to-end encryption in transit between the client and the web server.

Which solution will meet this requirement?

A.Place the EC2 instances behind an Application Load Balancer (ALB). Provision an SSL certificate using AWS Certificate Manager (ACM), and associate the SSL certificate with the ALB. Export the SSL certificate and install it on each EC2 instance. Configure the ALB to listen on port 443 and to forward traffic to port 443 on the instances.

B.Associate the EC2 instances with a target group. Provision an SSL certificate using AWS Certificate Manager (ACM). Create an Amazon CloudFront distribution and configure it to use the SSL certificate. Set CloudFront to use the target group as the origin server.

C.Place the EC2 instances behind an Application Load Balancer (ALB) Provision an SSL certificate using AWS Certificate Manager (ACM), and associate the SSL certificate with the ALB. Provision a third-party SSL certificate and install it on each EC2 instance. Configure the ALB to listen on port 443 and to forward traffic to port 443 on the instances.

D.Place the EC2 instances behind a Network Load Balancer (NLB). Provision a third-party SSL certificate and install it on the NLB and on each EC2 instance. Configure the NLB to listen on port 443 and to forward traffic to port 443 on the instances.

Answer: C

Explanation:

The correct answer is C. Here's why:

The primary requirement is end-to-end encryption. This means encryption must occur between the client and the load balancer, and then again between the load balancer and the EC2 instances.

Option C achieves this using an Application Load Balancer (ALB). An ALB can handle HTTPS traffic. The ACM-provisioned certificate is associated with the ALB to handle encryption between the client and the ALB. The third-party SSL certificate installed on the EC2 instances ensures encrypted communication between the ALB and the EC2 instances. Configuring the ALB to listen on port 443 (HTTPS) and forward to port 443 on the instances ensures that traffic remains encrypted throughout the entire path.

Option A is incorrect because it suggests exporting the ACM certificate and installing it on each EC2 instance, which is not the standard recommended practice. Using a separate certificate on the instances allows for independent certificate management and greater flexibility. Also, ACM certificates are designed to be used

with AWS services and exporting them is generally restricted.

Option B is incorrect because CloudFront, while capable of HTTPS, is primarily a Content Delivery Network (CDN). While CloudFront can encrypt traffic between the client and CloudFront, it doesn't inherently guarantee encryption between CloudFront and the EC2 instances unless explicitly configured. The description lacks the necessary detail to ensure end-to-end encryption is properly set up and managed.

Option D is incorrect because Network Load Balancers (NLBs) operate at the TCP layer (Layer 4). While they can forward traffic over port 443, they don't inherently handle SSL/TLS termination like ALBs do. NLBs do not have the capability to decrypt and re-encrypt the traffic; this functionality is provided by the ALB. While an NLB can pass encrypted traffic to EC2 instances that handle SSL/TLS termination, this approach is generally less efficient and more complex than using an ALB, which is designed for application-level load balancing and SSL/TLS management. Installing a certificate on the NLB implies it will do the TLS termination, a functionality that is not its primary design use.Here are some helpful links:

Application Load Balancer:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html
Network Load Balancer:https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html
AWS Certificate Manager:https://aws.amazon.com/certificate-manager/

Question: 50

A company wants to migrate its data analytics environment from on premises to AWS. The environment consists of two simple Node.js applications. One of the applications collects sensor data and loads it into a MySQL database. The other application aggregates the data into reports. When the aggregation jobs run, some of the load jobs fail to run correctly.

The company must resolve the data loading issue. The company also needs the migration to occur without interruptions or changes for the company's customers.

What should a solutions architect do to meet these requirements?

A.Set up an Amazon Aurora MySQL database as a replication target for the on-premises database. Create an Aurora Replica for the Aurora MySQL database, and move the aggregation jobs to run against the Aurora Replica. Set up collection endpoints as AWS Lambda functions behind a Network Load Balancer (NLB), and use Amazon RDS Proxy to write to the Aurora MySQL database. When the databases are synced, disable the replication job and restart the Aurora Replica as the primary instance. Point the collector DNS record to the NLB.

B.Set up an Amazon Aurora MySQL database. Use AWS Database Migration Service (AWS DMS) to perform continuous data replication from the on-premises database to Aurora. Move the aggregation jobs to run against the Aurora MySQL database. Set up collection endpoints behind an Application Load Balancer (ALB) as Amazon EC2 instances in an Auto Scaling group. When the databases are synced, point the collector DNS record to the ALDisable the AWS DMS sync task after the cutover from on premises to AWS.

C.Set up an Amazon Aurora MySQL database. Use AWS Database Migration Service (AWS DMS) to perform continuous data replication from the on-premises database to Aurora. Create an Aurora Replica for the Aurora MySQL database, and move the aggregation jobs to run against the Aurora Replica. Set up collection endpoints as AWS Lambda functions behind an Application Load Balancer (ALB), and use Amazon RDS Proxy to write to the Aurora MySQL database. When the databases are synced, point the collector DNS record to the ALB. Disable the AWS DMS sync task after the cutover from on premises to AWS.

D.Set up an Amazon Aurora MySQL database. Create an Aurora Replica for the Aurora MySQL database, and move the aggregation jobs to run against the Aurora Replica. Set up collection endpoints as an Amazon Kinesis data stream. Use Amazon Kinesis Data Firehose to replicate the data to the Aurora MySQL database. When the databases are synced, disable the replication job and restart the Aurora Replica as the primary instance. Point the collector DNS record to the Kinesis data stream.

Answer: C

Explanation:

The correct answer is C. Here's why:

Migration Strategy: AWS DMS is the ideal service for migrating databases to AWS with minimal downtime. It supports continuous data replication, allowing for a seamless transition.

Read Replica for Aggregation: Moving the aggregation jobs to an Aurora Replica offloads the read-heavy reporting workload from the primary database, preventing performance impact on the data loading process and addressing the original data loading failures. Aurora Replicas provide read scalability and improved availability.

Scalable Collection Endpoints: Using Lambda functions behind an ALB provides a serverless, scalable, and cost-effective solution for handling sensor data collection. Lambda functions can automatically scale based on demand. ALB allows for intelligent routing and load balancing of requests to the Lambda functions. **RDS Proxy for Connection Management:** RDS Proxy optimizes database connections, preventing connection exhaustion and improving application availability, especially when using serverless functions like Lambda which can generate numerous concurrent connections.

DNS Cutover: Pointing the collector DNS record to the ALB after the databases are synced directs incoming traffic to the migrated environment, completing the migration.

Disabling DMS: After the cutover and successful data synchronization, disabling the DMS task prevents unnecessary data replication and cost.

Option A is incorrect because it uses NLB which is more suitable for TCP traffic and is not the best choice for HTTP traffic from Lambda functions. While theoretically possible, it's not the most efficient architecture. It also promotes restarting the Aurora Replica as the primary instance which isn't typically recommended for simple cutover scenarios.

Option B is incorrect because it moves the aggregation jobs to the primary database, which can impact the data loading process, negating the original requirement. EC2 instances behind an ALB would work but are less cost-effective than serverless Lambda.

Option D is incorrect because Kinesis Data Streams are designed for real-time streaming data and would require significant code changes to the existing Node.js application. Using Kinesis Data Firehose to replicate to Aurora MySQL is not its intended purpose and is more suitable for data lakes or analytics use cases, not direct database replication for existing applications.

Supporting Documentation:

AWS Database Migration Service (DMS):https://aws.amazon.com/dms/
Amazon Aurora:https://aws.amazon.com/rds/aurora/
AWS Lambda:https://aws.amazon.com/lambda/
Application Load Balancer (ALB):https://aws.amazon.com/elasticloadbalancing/application-load-balancer/
Amazon RDS Proxy:https://aws.amazon.com/rds/proxy/

Question: 51

A health insurance company stores personally identifiable information (PII) in an Amazon S3 bucket. The company uses server-side encryption with S3 managed encryption keys (SSE-S3) to encrypt the objects. According to a new requirement, all current and future objects in the S3 bucket must be encrypted by keys that the company's security team manages. The S3 bucket does not have versioning enabled.

Which solution will meet these requirements?

A.In the S3 bucket properties, change the default encryption to SSE-S3 with a customer managed key. Use the AWS CLI to re-upload all objects in the S3 bucket. Set an S3 bucket policy to deny unencrypted PutObject requests.

B.In the S3 bucket properties, change the default encryption to server-side encryption with AWS KMS managed encryption keys (SSE-KMS). Set an S3 bucket policy to deny unencrypted PutObject requests. Use the AWS CLI to re-upload all objects in the S3 bucket.

C.In the S3 bucket properties, change the default encryption to server-side encryption with AWS KMS managed encryption keys (SSE-KMS). Set an S3 bucket policy to automatically encrypt objects on GetObject and PutObject requests.

D.In the S3 bucket properties, change the default encryption to AES-256 with a customer managed key. Attach a policy to deny unencrypted PutObject requests to any entities that access the S3 bucket. Use the AWS CLI to re-upload all objects in the S3 bucket.

Answer: B

Explanation:

The requirement is to encrypt S3 objects with keys managed by the company's security team, replacing the current SSE-S3 encryption. SSE-KMS is the correct encryption method for this because it allows customers to use AWS Key Management Service (KMS) to manage the encryption keys. This gives the company's security team control over the key lifecycle, including rotation, access policies, and auditing. Option A is incorrect because it refers to "SSE-S3 with a customer managed key," which is not a valid option. SSE-S3 uses keys managed entirely by AWS. Option D mentions AES-256 with a customer-managed key, which is not a valid S3 encryption type. AES-256 is the encryption algorithm used by SSE-S3.

To implement the solution in Option B: First, the default encryption for the S3 bucket must be changed to SSE-KMS. This ensures that all newly uploaded objects will be encrypted using the KMS key. Then, the existing objects, currently encrypted with SSE-S3, need to be re-encrypted using SSE-KMS. Re-uploading these objects using the AWS CLI achieves this re-encryption. Finally, a bucket policy that denies unencrypted PutObject requests prevents future uploads without encryption. Option C is not correct because S3 bucket policies do not automatically encrypt objects on GetObject or PutObject. Bucket policies control access and enforce conditions on requests. They can deny unencrypted requests, ensuring encryption is enforced but don't perform the encryption themselves.

Refer to the AWS documentation on server-side encryption to understand the different encryption options and the role of KMS:

https://docs.aws.amazon.com/AmazonS3/latest/userguide/serv-side-encryption.html
https://docs.aws.amazon.com/kms/latest/developerguide/overview.html

Question: 52

A company is running a web application in the AWS Cloud. The application consists of dynamic content that is created on a set of Amazon EC2 instances. The EC2 instances run in an Auto Scaling group that is configured as a target group for an Application Load Balancer (ALB).

The company is using an Amazon CloudFront distribution to distribute the application globally. The CloudFront distribution uses the ALB as an origin. The company uses Amazon Route 53 for DNS and has created an A record of www.example.com for the CloudFront distribution.

A solutions architect must configure the application so that itis highly available and fault tolerant.

Which solution meets these requirements?

A.Provision a full, secondary application deployment in a different AWS Region. Update the Route 53 A record to be a failover record. Add both of the CloudFront distributions as values. Create Route 53 health checks.

B.Provision an ALB, an Auto Scaling group, and EC2 instances in a different AWS Region. Update the CloudFront distribution, and create a second origin for the new ALCreate an origin group for the two origins. Configure one origin as primary and one origin as secondary.

C.Provision an Auto Scaling group and EC2 instances in a different AWS Region. Create a second target for the new Auto Scaling group in the ALB. Set up the failover routing algorithm on the ALB.

D.Provision a full, secondary application deployment in a different AWS Region. Create a second CloudFront distribution, and add the new application setup as an origin. Create an AWS Global Accelerator accelerator.

Add both of the CloudFront distributions as endpoints.

Answer: B

Explanation:

The correct answer is **B**. Here's a detailed justification:

The primary requirement is to achieve high availability and fault tolerance for a web application distributed globally via CloudFront. This necessitates redundancy and the ability to failover to a healthy environment in case of regional failures.

Option B offers the most suitable solution by creating a complete secondary deployment in a separate AWS Region. This secondary deployment mirrors the primary deployment, consisting of an ALB, an Auto Scaling group, and EC2 instances. This ensures that a fully functional backup is readily available.

Crucially, it updates the existing CloudFront distribution to include a second origin, the ALB from the secondary region. It then creates an origin group within CloudFront. This is the key to the failover mechanism.

The origin group allows you to designate one origin as primary and the other as secondary. CloudFront automatically routes traffic to the secondary origin if it detects that the primary origin is unavailable or unhealthy, effectively providing automatic failover.

Route 53's role is already established for DNS resolution to the initial CloudFront distribution. By leveraging CloudFront origin groups, the DNS configuration remains unchanged, simplifying the failover process. CloudFront is responsible for intelligent routing between origins based on health checks and pre-defined configuration.

Option A is less ideal. While using Route 53 failover records could work, relying solely on Route 53 for failover can be slower than utilizing CloudFront's origin groups, as DNS propagation delays can impact recovery time.

Option C is insufficient. Adding a second target to the ALB in a different region doesn't automatically achieve regional failover. ALBs are regional resources, and a single ALB cannot span regions.

Option D is unnecessarily complex and potentially more expensive. Creating a second CloudFront distribution and using AWS Global Accelerator introduces additional infrastructure and configuration without providing a significant advantage over the CloudFront origin group approach. The Global Accelerator would add an additional layer of routing that might increase latency compared to leveraging CloudFront's inherent capabilities.

In summary, the solution in option B effectively uses the native features of CloudFront (origin groups) to provide a seamless failover between the primary and secondary application deployments, meeting the requirements for high availability and fault tolerance with minimal complexity.

Authoritative Links:

Amazon CloudFront Origin Groups:
https://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/high_availability_origin_failover.html
Application Load Balancer:https://aws.amazon.com/elasticloadbalancing/application-load-balancer/ **Auto Scaling:**https://aws.amazon.com/autoscaling/

Question: 53

A company has an organization in AWS Organizations that has a large number of AWS accounts. One of the AWS accounts is designated as a transit account and has a transit gateway that is shared with all of the other AWS accounts. AWS Site-to-Site VPN connections are configured between all of the company’s global offices and the transit account. The company has AWS Config enabled on all of its accounts.

The company's networking team needs to centrally manage a list of internal IP address ranges that belong to the global offices. Developers will reference this list to gain access to their applications securely.

Which solution meets these requirements with the LEAST amount of operational overhead?

A.Create a JSON file that is hosted in Amazon S3 and that lists all of the internal IP address ranges. Configure an Amazon Simple Notification Service (Amazon SNS) topic in each of the accounts that can be invoked when the JSON file is updated. Subscribe an AWS Lambda function to the SNS topic to update all relevant security group rules with the updated IP address ranges.

B.Create a new AWS Config managed rule that contains all of the internal IP address ranges. Use the rule to check the security groups in each of the accounts to ensure compliance with the list of IP address ranges. Configure the rule to automatically remediate any noncompliant security group that is detected.

C.In the transit account, create a VPC prefix list with all of the internal IP address ranges. Use AWS Resource Access Manager to share the prefix list with all of the other accounts. Use the shared prefix list to configure security group rules in the other accounts.

D.In the transit account, create a security group with all of the internal IP address ranges. Configure the security groups in the other accounts to reference the transit account's security group by using a nested security group reference of "/sg-1a2b3c4d".

Answer: C

Explanation:

The correct answer is C because it provides a centralized, scalable, and least operationally overhead solution for managing IP address ranges across multiple AWS accounts within an organization.

Here's a detailed justification:

Centralized Management: VPC prefix lists allow you to group and manage collections of CIDR blocks as a single object. By creating the prefix list in the transit account, the networking team can maintain a single source of truth for internal IP address ranges.

Sharing with RAM: AWS Resource Access Manager (RAM) enables you to securely share AWS resources across AWS accounts, within your organization. Sharing the VPC prefix list with all other accounts allows these accounts to use it in their security group rules.

Simplified Security Group Management: By referencing the shared prefix list in security group rules, developers can automatically inherit any updates to the IP address ranges. This eliminates the need to manually update security groups in each account, reducing operational overhead.

Least Operational Overhead: Option A involves S3, SNS, Lambda, and custom code for security group updates which introduces significant management overhead. Option B requires custom AWS Config rule development and remediation, which is more complex. Option D using nested security group references across account boundaries isn't natively supported, thus infeasible. Option C directly uses AWS features built for resource sharing.

Scalability: Prefix lists can be updated easily, and changes are automatically propagated to all security groups that reference them. This ensures that the security posture remains consistent as the company's network evolves.

In summary, option C leverages AWS's native resource sharing capabilities (RAM and VPC prefix lists) to provide a scalable, centralized, and operationally efficient solution for managing IP address ranges across multiple AWS accounts.

Supporting Links:

VPC Prefix Lists:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-prefix-lists.html **AWS Resource Access Manager (RAM):**https://aws.amazon.com/ram/

Question: 54

A company runs a new application as a static website in Amazon S3. The company has deployed the application to a production AWS account and uses Amazon CloudFront to deliver the website. The website calls an Amazon API Gateway REST API. An AWS Lambda function backs each API method.

The company wants to create a CSV report every 2 weeks to show each API Lambda function's recommended configured memory, recommended cost, and the price difference between current configurations and the recommendations. The company will store the reports in an S3 bucket.

Which solution will meet these requirements with the LEAST development time?

A.Create a Lambda function that extracts metrics data for each API Lambda function from Amazon CloudWatch Logs for the 2-week period. Collate the data into tabular format. Store the data as a .csv file in an S3 bucket. Create an Amazon EventBridge rule to schedule the Lambda function to run every 2 weeks.

B.Opt in to AWS Compute Optimizer. Create a Lambda function that calls the ExportLambdaFunctionRecommendations operation. Export the .csv file to an S3 bucket. Create an Amazon EventBridge rule to schedule the Lambda function to run every 2 weeks.

C.Opt in to AWS Compute Optimizer. Set up enhanced infrastructure metrics. Within the Compute Optimizer console, schedule a job to export the Lambda recommendations to a .csv file. Store the file in an S3 bucket every 2 weeks.

D.Purchase the AWS Business Support plan for the production account. Opt in to AWS Compute Optimizer for AWS Trusted Advisor checks. In the Trusted Advisor console, schedule a job to export the cost optimization checks to a .csv file. Store the file in an S3 bucket every 2 weeks.

Answer: B

Explanation:

The correct answer is B because it leverages AWS Compute Optimizer to provide the desired recommendations with minimal development effort. Here's a detailed justification:

Requirement: The company needs a report with recommended memory, cost, and price difference for each API Lambda function, generated every two weeks.

AWS Compute Optimizer: This service analyzes the configuration and utilization metrics of AWS resources, including Lambda functions, and provides optimization recommendations.

ExportLambdaFunctionRecommendations API: AWS Compute Optimizer offers an API that allows exporting its recommendations in CSV format, directly addressing the requirement for a CSV report.

https://docs.aws.amazon.com/compute-optimizer/latest/APIReference/API_ExportLambdaFunctionRecommendations.html

Lambda Function & EventBridge: A Lambda function can call the ExportLambdaFunctionRecommendations API and store the resulting CSV file in an S3 bucket. Amazon EventBridge can schedule this Lambda function to run every two weeks. This automates the report generation process.

Least Development Time: This solution utilizes a pre-built service and a simple API call, minimizing the amount of custom code that needs to be written and maintained.

Why other options are less suitable:

A: Extracting metrics from CloudWatch Logs and manually calculating recommendations would require significant development effort and potentially complex logic.

C: While Compute Optimizer can export recommendations, scheduling the export job within the console every two weeks involves manual intervention, which is against the automation goal.

D: AWS Trusted Advisor's cost optimization checks provide general cost-saving recommendations but are not as granular or specific to Lambda function memory allocation and cost as Compute Optimizer. Also, purchasing a business support plan is not the most effective way to resolve this particular issue; moreover, Trusted Advisor doesn't have a direct export to csv as seamless as the suggested solution.

In summary, Option B provides the most efficient and automated solution by leveraging AWS Compute Optimizer's API to generate the required report with minimal development time.

Question: 55

A company's factory and automation applications are running in a single VPC. More than 20 applications run on a combination of Amazon EC2, Amazon Elastic Container Service (Amazon ECS), and Amazon RDS.

The company has software engineers spread across three teams. One of the three teams owns each application, and each time is responsible for the cost and performance of all of its applications. Team resources have tags that represent their application and team. The teams use IAM access for daily activities.

The company needs to determine which costs on the monthly AWS bill are attributable to each application or team. The company also must be able to create reports to compare costs from the last 12 months and to help forecast costs for the next 12 months. A solutions architect must recommend an AWS Billing and Cost Management solution that provides these cost reports.

Which combination of actions will meet these requirements? (Choose three.)

A.Activate the user-define cost allocation tags that represent the application and the team.
B.Activate the AWS generated cost allocation tags that represent the application and the team.
C.Create a cost category for each application in Billing and Cost Management.
D.Activate IAM access to Billing and Cost Management.
E.Create a cost budget.
F.Enable Cost Explorer.

Answer: ACF

Explanation:

The correct answer is ACF. Here's why:

A. Activate the user-defined cost allocation tags that represent the application and the team: Cost allocation tags are essential for tracking costs associated with specific resources. User-defined tags are created and applied by the user to resources, enabling the organization to categorize costs based on application and team. Activating these tags ensures that cost information is associated with these tags and available for reporting. https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/cost-alloc-tags.html

C. Create a cost category for each application in Billing and Cost Management: Cost Categories allow you to group costs based on dimensions such as tags, accounts, and services. By creating a cost category for each application, the company can aggregate and analyze costs specifically related to each application, providing a clear breakdown of expenses. https://docs.aws.amazon.com/cost-management/latest/userguide/cost-categories.html

F. Enable Cost Explorer: Cost Explorer is a tool within AWS Billing and Cost Management that allows you to visualize, understand, and manage your AWS costs and usage over time. It provides features for analyzing cost trends, identifying cost drivers, and forecasting future costs, thus meeting the requirement for creating reports to compare costs and forecast for the next 12 months. https://docs.aws.amazon.com/cost-management/latest/userguide/ce-what-is.html

Why other options are incorrect:

B. Activate the AWS generated cost allocation tags that represent the application and the team: AWS-generated tags typically relate to resource creation and management and are unlikely to directly represent the application and team context that the company defines.

D. Activate IAM access to Billing and Cost Management: While IAM access is crucial for security, it doesn't directly contribute to creating the necessary cost reports. IAM is about who can access cost data, not how costs are allocated and analyzed.

E. Create a cost budget: Creating a budget helps monitor spending against a predefined limit. However, it doesn't directly create the cost allocation reports required to track costs by application and team, although it could use the Cost Categories created to track budgets against each application.

Question: 56

An AWS customer has a web application that runs on premises. The web application fetches data from a third-party API that is behind a firewall. The third party accepts only one public CIDR block in each client's allow list.

The customer wants to migrate their web application to the AWS Cloud. The application will be hosted on a set of Amazon EC2 instances behind an Application Load Balancer (ALB) in a VPC. The ALB is located in public subnets. The EC2 instances are located in private subnets. NAT gateways provide internet access to the private subnets.

How should a solutions architect ensure that the web application can continue to call the third-party API after the migration?

A.Associate a block of customer-owned public IP addresses to the VPC. Enable public IP addressing for public subnets in the VPC.

B.Register a block of customer-owned public IP addresses in the AWS account. Create Elastic IP addresses from the address block and assign them to the NAT gateways in the VPC.

C.Create Elastic IP addresses from the block of customer-owned IP addresses. Assign the static Elastic IP addresses to the ALB.

D.Register a block of customer-owned public IP addresses in the AWS account. Set up AWS Global Accelerator to use Elastic IP addresses from the address block. Set the ALB as the accelerator endpoint.

Answer: B

Explanation:

The correct solution (B) focuses on ensuring the third-party API consistently sees the same source IP addresses originating from the AWS environment, allowing proper firewall rules to be maintained.

Option B describes how to use Customer Owned IP addresses (CoIPs). CoIPs bring the customer's own IP range into AWS. When CoIPs are assigned to the NAT Gateways, the NAT Gateways will use these IPs as the source IPs for all outbound traffic. This ensures that all traffic destined for the third-party API originates from a known, dedicated IP address or addresses belonging to the customer.

Option A is incorrect because merely associating CoIPs with the VPC and enabling public IP addressing on public subnets does not guarantee that traffic from the EC2 instances to the third-party API will originate from those CoIP addresses. Without explicit assignment to NAT Gateways, the instances might use ephemeral public IPs.

Option C is incorrect because ALBs do not support assigning static Elastic IP addresses directly. An ALB's IP addresses can change over time, making it unsuitable for a scenario requiring a fixed source IP for firewall purposes.

Option D is incorrect because AWS Global Accelerator with an ALB endpoint does not guarantee a static IP for outbound traffic originating from the EC2 instances. While Global Accelerator provides static entry points to your application, the traffic from your NAT Gateways to the third-party API would still use the NAT Gateway's IP, which must be predictable. Additionally, Global Accelerator introduces complexity and costs that are unnecessary in this scenario, as the primary requirement is a consistent source IP, not global distribution or accelerated traffic.

In conclusion, associating CoIPs with the NAT Gateways (Option B) provides the required static IPs needed for the third-party firewall, aligning with the customer's requirement and avoiding unnecessary complexity.

Here are some authoritative links for further research:

Customer-owned IP addresses (CoIPs):https://docs.aws.amazon.com/vpc/latest/coip/what-is-coip.html **NAT Gateway:**https://docs.aws.amazon.com/vpc/latest/userguide/vpc-nat-gateway.html
Elastic IP Addresses:https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/elastic-ip-addresses-eip.html
Application Load Balancer:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html
AWS Global Accelerator:https://aws.amazon.com/global-accelerator/

Question: 57

A company with several AWS accounts is using AWS Organizations and service control policies (SCPs). An administrator created the following SCP and has attached it to an organizational unit (OU) that contains AWS account 1111-1111-1111:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowsAllActions",
            "Effect": "Allow",
            "Action": "*",
            "Resource": "*"
        },
        {
            "Sid": "DenyCloudTrail",
            "Effect": "Deny",
            "Action": "cloudtrail:*",
            "Resource": "*"
        }
    ]
}
```

Developers working in account 1111-1111-1111 complain that they cannot create Amazon S3 buckets. How should the administrator address this problem?

A.Add s3:CreateBucket with "Allow" effect to the SCP.
B.Remove the account from the OU, and attach the SCP directly to account 1111-1111-1111.
C.Instruct the developers to add Amazon S3 permissions to their IAM entities.
D.Remove the SCP from account 1111-1111-1111.

Answer: C

Explanation:

C - Users and roles must still be granted permissions with appropriate IAM permission policies. A user without any IAM permission policies has no access at all, even if the applicable SCPs allow all services and all actions.

Question: 58

A company has a monolithic application that is critical to the company's business. The company hosts the application on an Amazon EC2 instance that runs Amazon Linux 2. The company's application team receives a directive from the legal department to back up the data from the instance's encrypted Amazon Elastic Block Store (Amazon EBS) volume to an Amazon S3 bucket. The application team does not have the administrative SSH key pair for the instance. The application must continue to serve the users.

Which solution will meet these requirements?

A.Attach a role to the instance with permission to write to Amazon S3. Use the AWS Systems Manager Session Manager option to gain access to the instance and run commands to copy data into Amazon S3.

B.Create an image of the instance with the reboot option turned on. Launch a new EC2 instance from the image. Attach a role to the new instance with permission to write to Amazon S3. Run a command to copy data into Amazon S3.

C.Take a snapshot of the EBS volume by using Amazon Data Lifecycle Manager (Amazon DLM). Copy the data to Amazon S3.

D.Create an image of the instance. Launch a new EC2 instance from the image. Attach a role to the new instance with permission to write to Amazon S3. Run a command to copy data into Amazon S3.

Answer: A

Explanation:

The correct answer is A. Here's why:

Requirement to Back Up EBS Data to S3: The primary goal is to back up the data from the encrypted EBS volume to an S3 bucket.

No SSH Access: The application team doesn't have the SSH key pair, which prevents direct access to the EC2 instance via SSH.

Application Must Continue Serving Users: The backup process should not interrupt the application's availability.

Why Option A is Correct:

1. **IAM Role for S3 Access:** Attaching an IAM role to the EC2 instance grants it the necessary permissions to write data to the designated S3 bucket without requiring SSH keys. This adheres to the principle of least privilege. https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html 2. **AWS Systems Manager Session Manager:** Session Manager provides secure and auditable instance management without the need to open inbound ports or maintain SSH keys. It allows running commands on the EC2 instance without direct SSH access, fulfilling the requirement of no key pair.
   https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager.html
3. **Copying Data to S3:** Using commands through Session Manager, the application team can copy data from the EBS volume to S3. The specific command would involve leveraging the AWS CLI, which can be configured to use the instance's IAM role for authentication.
4. **Minimal Downtime:** Option A provides a way to execute the backup procedure on the existing EC2 instance without requiring recreation or rebooting. The whole operation can happen while the instance continues to run.

Why Other Options are Incorrect:

Option B & D: Creating an image of the instance necessitates a restart, which would interrupt application service. It also involves creating a new EC2 instance which adds to the complexity and costs.

Option C: While Amazon DLM can create EBS snapshots, it does not directly copy the data contained within to S3. DLM manages the lifecycle of EBS snapshots but would require other means to access the volume data and copy it, which is not specified in the option, thus rendering it incomplete. The snapshots themselves are stored within EBS, not S3.

Question: 59

A solutions architect needs to copy data from an Amazon S3 bucket m an AWS account to a new S3 bucket in a new AWS account. The solutions architect must implement a solution that uses the AWS CLI.

Which combination of steps will successfully copy the data? (Choose three.)

A.Create a bucket policy to allow the source bucket to list its contents and to put objects and set object ACLs in the destination bucket. Attach the bucket policy to the destination bucket.

B.Create a bucket policy to allow a user in the destination account to list the source bucket's contents and read the source bucket's objects. Attach the bucket policy to the source bucket.

C.Create an IAM policy in the source account. Configure the policy to allow a user in the source account to list contents and get objects in the source bucket, and to list contents, put objects, and set object ACLs in the destination bucket. Attach the policy to the user.

D.Create an IAM policy in the destination account. Configure the policy to allow a user in the destination account to list contents and get objects in the source bucket, and to list contents, put objects, and set objectACLs in the destination bucket. Attach the policy to the user.

E.Run the aws s3 sync command as a user in the source account. Specify the source and destination buckets to copy the data.

F.Run the aws s3 sync command as a user in the destination account. Specify the source and destination buckets to copy the data.

Answer: BDF

Explanation:

The correct answer is BDF because it outlines the necessary steps to securely and successfully copy data from an S3 bucket in one AWS account to another using the AWS CLI. Here's a breakdown of why each choice is correct and why the others aren't:

B. Create a bucket policy to allow a user in the destination account to list the source bucket's contents and read the source bucket's objects. Attach the bucket policy to the source bucket. This is essential for allowing the destination account to access the source bucket's data. The bucket policy on the source bucket needs to explicitly grant permissions to the destination account's IAM user/role to read the bucket contents.

Without this, the destination account cannot retrieve the objects to copy. (Refer to https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html for bucket policy examples)

D. Create an IAM policy in the destination account. Configure the policy to allow a user in the destination account to list contents and get objects in the source bucket, and to list contents, put objects, and set objectACLs in the destination bucket. Attach the policy to the user. The user in the destination account, running the AWS CLI command, requires the necessary permissions to not only read from the source bucket but also to write to the destination bucket. This IAM policy grants those permissions. It's crucial to include s3:PutObject and s3:PutObjectAcl to allow writing objects and setting ACLs in the destination bucket. (Refer to https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html for IAM policies)

F. Run the aws s3 sync command as a user in the destination account. Specify the source and destination buckets to copy the data. This command initiates the data transfer. It must be run from the destination account because that account needs to pull the data. Running it from the source account wouldn't be secure, since we want to grant granular permissions to the destination account to copy the data. (Refer to https://awscli.amazonaws.com/v2/documentation/api/latest/reference/s3/sync.html for aws s3 sync documentation)

Here's why the other options are incorrect:

A. Attaching a bucket policy to the destination bucket granting the source bucket access doesn't make logical sense. The destination needs to pull data from the source. The source needs to allow the pull. **C.** While granting the source account's user permissions is possible, it's less secure and violates the principle of least privilege. The destination account should explicitly be granted permission.

E. Running `aws s3 sync` as a user in the source account is incorrect. The source account needs to allow access, but the destination account must pull the data.

Question: 60

A company built an application based on AWS Lambda deployed in an AWS CloudFormation stack. The last production release of the web application introduced an issue that resulted in an outage lasting several minutes. A solutions architect must adjust the deployment process to support a canary release.

Which solution will meet these requirements?

A.Create an alias for every new deployed version of the Lambda function. Use the AWS CLI update-alias command with the routing-config parameter to distribute the load.

B.Deploy the application into a new CloudFormation stack. Use an Amazon Route 53 weighted routing policy to distribute the load.

C.Create a version for every new deployed Lambda function. Use the AWS CLI update-function-configuration command with the routing-config parameter to distribute the load.

D.Configure AWS CodeDeploy and use CodeDeployDefault.OneAtATime in the Deployment configuration to distribute the load.

Answer: A

Explanation:

The correct answer is A. Here's a detailed justification:

A canary release is a deployment strategy where a new version of an application is rolled out to a small subset of users before a wider deployment. This allows for early detection of issues and minimizes the impact of potential bugs.

Option A leverages Lambda aliases and weighted routing to achieve this. Lambda aliases are pointers to specific Lambda function versions. By creating an alias for each new deployed version, you can control the traffic directed to each version. The `update-alias` command in the AWS CLI, along with the `routing-config` parameter, allows you to specify the percentage of traffic that should be routed to the new version. Gradually increasing the traffic to the new version allows you to monitor its performance and stability without impacting all users. This implements the core concept of a canary release.

Option B is less efficient because deploying into a new CloudFormation stack for each release is more complex and resource-intensive than simply updating a Lambda alias. While Route 53 weighted routing could direct traffic, it wouldn't be directly integrated with Lambda versioning and canary release strategies.

Option C is incorrect because `update-function-configuration` does not support traffic routing. It's for updating function configurations like memory and timeout settings.

Option D, using CodeDeploy, is overkill for a single Lambda function deployment. CodeDeploy is more suitable for deploying applications across multiple EC2 instances or managing deployments of multiple application components. Plus, `CodeDeployDefault.OneAtATime` isn't designed for canary deployments; it sequentially updates instances.

In summary, option A offers the most straightforward and efficient way to implement a canary release for a Lambda function by utilizing Lambda aliases and weighted traffic routing, allowing for granular control and monitoring of the new version.

Supporting Documentation:

AWS Lambda Versions and Aliases:https://docs.aws.amazon.com/lambda/latest/dg/configuration-versions.html
AWS CLI update-alias Command:
https://awscli.amazonaws.com/v2/documentation/api/latest/reference/lambda/update-alias.html

Question: 61

A finance company hosts a data lake in Amazon S3. The company receives financial data records over SFTP each night from several third parties. The company runs its own SFTP server on an Amazon EC2 instance in a public subnet of a VPC. After the files are uploaded, they are moved to the data lake by a cron job that runs on the same instance. The SFTP server is reachable on DNS sftp.example.com through the use of Amazon Route 53.

What should a solutions architect do to improve the reliability and scalability of the SFTP solution?

A.Move the EC2 instance into an Auto Scaling group. Place the EC2 instance behind an Application Load Balancer (ALB). Update the DNS record sftp.example.com in Route 53 to point to the ALB.

B.Migrate the SFTP server to AWS Transfer for SFTP. Update the DNS record sftp.example.com in Route 53 to point to the server endpoint hostname.

C.Migrate the SFTP server to a file gateway in AWS Storage Gateway. Update the DNS record sftp.example.com in Route 53 to point to the file gateway endpoint.

D.Place the EC2 instance behind a Network Load Balancer (NLB). Update the DNS record sftp.example.com in Route 53 to point to the NLB.

Answer: B

Explanation:

The best solution is **B. Migrate the SFTP server to AWS Transfer for SFTP. Update the DNS record sftp.example.com in Route 53 to point to the server endpoint hostname.**

Here's why:

AWS Transfer for SFTP is a fully managed service specifically designed for secure file transfers directly into and out of Amazon S3. This eliminates the need for managing EC2 instances, patching, scaling, and ensuring high availability. It inherently provides reliability and scalability that manual EC2-based SFTP servers lack.
https://aws.amazon.com/aws-transfer-family/

Reliability: AWS Transfer for SFTP automatically handles patching, updates, and backups. It is designed for high availability, ensuring SFTP services remain operational. The managed service minimizes the risk of downtime associated with EC2 instance failures or maintenance.

Scalability: AWS Transfer for SFTP scales automatically to handle fluctuating workloads. It can accommodate increasing data volumes and concurrent connections without requiring manual intervention. Scaling the SFTP infrastructure is hands-off, unlike scaling EC2 instances.

Security: AWS Transfer for SFTP integrates seamlessly with AWS Identity and Access Management (IAM) and other security services, enhancing the security posture of file transfers.

Cost-Effectiveness: While initially seeming more expensive, AWS Transfer for SFTP often reduces overall costs by eliminating the operational overhead of managing EC2 instances, EBS volumes, and related resources. The pay-as-you-go model optimizes spending.

Option A (Auto Scaling Group with ALB): While improving availability, it still involves managing EC2 instances, OS patching, and application updates. An Application Load Balancer is not ideal for SFTP traffic, as it is primarily for HTTP/HTTPS.

Option C (File Gateway): AWS Storage Gateway's File Gateway is for providing on-premises applications with file-based access to S3. It's not designed to replace an SFTP server directly receiving data from external third parties. Furthermore, it adds complexity compared to AWS Transfer for SFTP.
https://aws.amazon.com/storagegateway/file-gateway/

Option D (NLB): A Network Load Balancer (NLB) provides TCP-level load balancing, which can improve availability compared to a single EC2 instance. However, it still requires managing the underlying EC2 instance and SFTP server. More importantly, it doesn't offer the fully managed benefits and scalability of AWS Transfer for SFTP.

Question: 62

A company wants to migrate an application to Amazon EC2 from VMware Infrastructure that runs in an on-premises data center. A solutions architect must preserve the software and configuration settings during the migration.

What should the solutions architect do to meet these requirements?

A.Configure the AWS DataSync agent to start replicating the data store to Amazon FSx for Windows File Server. Use the SMB share to host the VMware data store. Use VM Import/Export to move the VMs to Amazon EC2.

B.Use the VMware vSphere client to export the application as an image in Open Virtualization Format (OVF) format. Create an Amazon S3 bucket to store the image in the destination AWS Region. Create and apply an IAM role for VM Import. Use the AWS CLI to run the EC2 import command.

C.Configure AWS Storage Gateway for files service to export a Common Internet File System (CIFS) share. Create a backup copy to the shared folder. Sign in to the AWS Management Console and create an AMI from the backup copy. Launch an EC2 instance that is based on the AMI.

D.Create a managed-instance activation for a hybrid environment in AWS Systems Manager. Download and install Systems Manager Agent on the on-premises VM. Register the VM with Systems Manager to be a managed instance. Use AWS Backup to create a snapshot of the VM and create an AMI. Launch an EC2 instance that is based on the AMI.

Answer: B

Explanation:

The correct answer is **B**. Here's a detailed justification:

Why Option B is Correct:

Option B leverages VM Import/Export, a service specifically designed to migrate virtual machine images from on-premises environments (like VMware) to Amazon EC2. The process involves:

1. **Exporting as OVF:** Exporting the application as an Open Virtualization Format (OVF) file preserves the entire virtual machine configuration, including software, OS, and settings. OVF is a packaging standard specifically created for virtual appliances, making it well-suited for this migration.

2. **Storing in S3:** Storing the OVF in an Amazon S3 bucket provides a cost-effective and durable storage location accessible to the VM Import service. S3 is the recommended storage location for VM Import.

3. **IAM Role:** Creating and applying an IAM role grants the VM Import service the necessary permissions to access S3 and create EC2 instances. This is crucial for security and access control.

4. **AWS CLI:** Using the AWS CLI to run the EC2 import command initiates the import process. This command orchestrates the conversion and deployment of the OVF file to an EC2 instance.

Why Other Options are Incorrect:

Option A (AWS DataSync & FSx): AWS DataSync is excellent for replicating data, but not for migrating entire virtual machines, including their operating systems and configurations. Using FSx as a direct replacement for the VMware datastore is not the intended use case, and SMB shares are not a suitable way to import VMs.

Option C (AWS Storage Gateway for files & CIFS): Storage Gateway primarily provides a hybrid cloud storage solution, integrating on-premises applications with AWS storage services. Backing up the contents of a VM to a CIFS share and then creating an AMI won't necessarily capture the complete VM configuration correctly and will cause operational overhead for manual image creation, which is not the desired method when preserving software and configuration settings is a must.

Option D (AWS Systems Manager & AWS Backup): While AWS Systems Manager is great for managing hybrid environments and AWS Backup is good for protecting EC2, they aren't optimized for initial VM migration from on-premises. Creating a managed instance is suitable after the migration, for continuous management, but is not the primary method to move the VM in its current state to AWS.

In summary: Option B provides the most direct and recommended approach for migrating a VMware VM to Amazon EC2 while preserving its software and configuration settings using VM Import/Export. The process uses standardized formats, secure access controls, and a designated migration service.

Authoritative Links:

VM Import/Export:https://docs.aws.amazon.com/vm-import/latest/userguide/vmimport-image-import.html
OVF Format:https://www.vmware.com/support/developer/ovf/

Question: 63

A video processing company has an application that downloads images from an Amazon S3 bucket, processes the images, stores a transformed image in a second S3 bucket, and updates metadata about the image in an Amazon DynamoDB table. The application is written in Node.js and runs by using an AWS Lambda function. The Lambda function is invoked when a new image is uploaded to Amazon S3.

The application ran without incident for a while. However, the size of the images has grown significantly. The Lambda function is now failing frequently with timeout errors. The function timeout is set to its maximum value. A solutions architect needs to refactor the application's architecture to prevent invocation failures. The company does not want to manage the underlying infrastructure.

Which combination of steps should the solutions architect take to meet these requirements? (Choose two.)

A.Modify the application deployment by building a Docker image that contains the application code. Publish the image to Amazon Elastic Container Registry (Amazon ECR).

B.Create a new Amazon Elastic Container Service (Amazon ECS) task definition with a compatibility type of AWS Fargate. Configure the task definition to use the new image in Amazon Elastic Container Registry (Amazon ECR). Adjust the Lambda function to invoke an ECS task by using the ECS task definition when a new file arrives in Amazon S3.

C.Create an AWS Step Functions state machine with a Parallel state to invoke the Lambda function. Increase the provisioned concurrency of the Lambda function.

D.Create a new Amazon Elastic Container Service (Amazon ECS) task definition with a compatibility type of Amazon EC2. Configure the task definition to use the new image in Amazon Elastic Container Registry (Amazon ECR). Adjust the Lambda function to invoke an ECS task by using the ECS task definition when a new file arrives in Amazon S3.

E.Modify the application to store images on Amazon Elastic File System (Amazon EFS) and to store metadata on an Amazon RDS DB instance. Adjust the Lambda function to mount the EFS file share.

Answer: AB

Explanation:

The problem is Lambda function timeouts due to increasing image sizes exceeding the execution limits. The

solution needs to avoid infrastructure management and prevent invocation failures.

Option A suggests containerizing the application using Docker and storing the image in Amazon ECR. This is a good first step because containerization allows the application to run consistently in different environments, including AWS Fargate.

Option B suggests creating an ECS task definition with Fargate compatibility and invoking this task from the Lambda function upon S3 events. Fargate eliminates the need to manage EC2 instances, aligning with the "no infrastructure management" requirement. By offloading the image processing to an ECS task running on Fargate, the Lambda function's execution time is significantly reduced, preventing timeouts. The Lambda function acts as a trigger and orchestration service, and the computationally intensive image processing runs in ECS/Fargate.

Options C, D, and E are incorrect. Option C suggests using Step Functions and increasing provisioned concurrency for Lambda. While Step Functions can help orchestrate complex workflows, it doesn't address the underlying resource constraints of Lambda. Provisioned concurrency can help with cold starts but won't prevent timeouts if the function exceeds its execution time limit. Option D suggests using ECS with EC2, which contradicts the "no infrastructure management" requirement. Option E suggests using EFS and RDS. While EFS can store large files, it doesn't inherently solve the timeout problem within the Lambda function.

Additionally, the application currently uses DynamoDB, so migrating to RDS introduces unnecessary complexity and is not directly related to solving the timeout issue.

Therefore, options A and B together provide a suitable solution by moving the compute-intensive image processing out of the Lambda function and into a serverless container orchestration service like ECS Fargate.

Supporting Links:

AWS Lambda Execution Environment and Available Resources:
https://docs.aws.amazon.com/lambda/latest/dg/services-efs.html
AWS Fargate:https://aws.amazon.com/fargate/
Amazon ECS Task Definitions:
https://docs.aws.amazon.com/AmazonECS/latest/developerguide/task_definitions.html

Question: 64

A company has an organization in AWS Organizations. The company is using AWS Control Tower to deploy a landing zone for the organization. The company wants to implement governance and policy enforcement. The company must implement a policy that will detect Amazon RDS DB instances that are not encrypted at rest in the company's production OU.

Which solution will meet this requirement?

A.Turn on mandatory guardrails in AWS Control Tower. Apply the mandatory guardrails to the production OU.

B.Enable the appropriate guardrail from the list of strongly recommended guardrails in AWS Control Tower. Apply the guardrail to the production OU.

C.Use AWS Config to create a new mandatory guardrail. Apply the rule to all accounts in the production OU.

D.Create a custom SCP in AWS Control Tower. Apply the SCP to the production OU.

Answer: B

Explanation:

Here's a detailed justification for why option B is the most suitable solution for detecting unencrypted RDS instances in a production OU managed by AWS Control Tower:

AWS Control Tower is designed to establish and maintain a secure, well-governed multi-account AWS

environment. It provides pre-configured governance rules known as "guardrails" that are automatically applied to your organization. These guardrails fall into two categories: preventative (SCP-based) and detective (AWS Config rule-based).

The problem requires detecting unencrypted RDS instances. SCPs (Service Control Policies, as in option D) prevent actions; they don't detect non-compliance. Mandatory guardrails, mentioned in option A, often focus on preventative measures, meaning they actively prevent launching unencrypted RDS instances. This doesn't address the scenario where existing unencrypted instances need to be identified.

AWS Config rules, used in options B and C, are specifically designed for continuous compliance assessment. They evaluate the configuration settings of your AWS resources against desired configurations. Therefore, using Config is the right approach for detecting non-compliant resources.

Option C suggests creating a new mandatory guardrail with AWS Config. While this is possible, it's more complex and time-consuming than necessary. Control Tower provides a set of "strongly recommended guardrails" that address common compliance requirements. One of these guardrails likely detects unencrypted RDS instances. Leveraging the built-in guardrail is the most efficient and least disruptive approach.

By enabling the appropriate "strongly recommended guardrail" in Control Tower (option B) and applying it to the production OU, you leverage a pre-built, well-tested compliance rule to automatically and continuously check for unencrypted RDS instances. This aligns perfectly with the requirements of detecting non-compliance within the managed environment.

Authoritative links:

AWS Control Tower Guardrails: https://docs.aws.amazon.com/controltower/latest/userguide/guardrails.html AWS Config Rules: https://docs.aws.amazon.com/config/latest/developerguide/evaluate-config.html

Question: 65

A startup company hosts a fleet of Amazon EC2 instances in private subnets using the latest Amazon Linux 2 AMI. The company's engineers rely heavily on SSH access to the instances for troubleshooting.

The company's existing architecture includes the following:

•A VPC with private and public subnets, and a NAT gateway.
•Site-to-Site VPN for connectivity with the on-premises environment.
•EC2 security groups with direct SSH access from the on-premises environment.

The company needs to increase security controls around SSH access and provide auditing of commands run by the engineers.

Which strategy should a solutions architect use?

A.Install and configure EC2 Instance Connect on the fleet of EC2 instances. Remove all security group rules attached to EC2 instances that allow inbound TCP on port 22. Advise the engineers to remotely access the instances by using the EC2 Instance Connect CLI.

B.Update the EC2 security groups to only allow inbound TCP on port 22 to the IP addresses of the engineer's devices. Install the Amazon CloudWatch agent on all EC2 instances and send operating system audit logs to CloudWatch Logs.

C.Update the EC2 security groups to only allow inbound TCP on port 22 to the IP addresses of the engineer's devices. Enable AWS Config for EC2 security group resource changes. Enable AWS Firewall Manager and apply a security group policy that automatically remediates changes to rules.

D.Create an IAM role with the AmazonSSMManagedInstanceCore managed policy attached. Attach the IAM role to all the EC2 instances. Remove all security group rules attached to the EC2 instances that allow inbound TCP on port 22. Have the engineers install the AWS Systems Manager Session Manager plugin for their devices and remotely access the instances by using the start-session API call from Systems Manager.

Answer: D

Explanation:

Here's a detailed justification for why option D is the best solution, along with supporting concepts and links:

Option D leverages AWS Systems Manager (SSM) Session Manager, which is the most secure and auditable way to provide SSH-like access to EC2 instances without opening inbound ports. It addresses the security requirement of increasing security controls around SSH access and also fulfills the auditing requirement.

Here's why it works:

1. **No Inbound SSH Ports:** Option D removes the need for inbound SSH (port 22) access to the EC2 instances by removing all security group rules that allow inbound TCP on port 22. This significantly reduces the attack surface because the instances are no longer directly exposed to the internet or the on-premises network via SSH.

2. **SSM Session Manager:** SSM Session Manager establishes secure, bi-directional communication between the engineer's device and the EC2 instance via the AWS Systems Manager service. The connection originates from the instance to AWS over HTTPS (port 443), eliminating the need for any inbound rules.

3. **IAM Role for SSM Access:** The IAM role with the AmazonSSMManagedInstanceCore policy grants the EC2 instances the necessary permissions to communicate with the SSM service. This is a secure way to manage access because the instance assumes a role, rather than using static credentials.

4. **Auditing:** Session Manager automatically logs session activity to Amazon CloudWatch Logs or Amazon S3, fulfilling the requirement for auditing commands run by engineers. This allows for comprehensive tracking and analysis of user actions.

5. **Centralized Management:** SSM provides centralized management of EC2 instances, including session management, patching, and configuration. This improves operational efficiency.

Options A, B, and C are less secure or don't adequately address the auditing requirement:

Option A (EC2 Instance Connect): While EC2 Instance Connect also allows connection without SSH keys, it still requires a security group rule allowing inbound SSH from the EC2 Instance Connect service IP ranges, increasing attack surface. It does not natively provide comprehensive command auditing.

Options B and C (IP-Based Security Group Rules): These options only limit access to specific IP addresses, which is better than open access but still exposes SSH. IP addresses can change and are not a strong form of authentication. CloudWatch agent for OS audit logs requires additional configuration and management effort. AWS Firewall Manager is overkill for this specific scenario and focuses on broad firewall policy enforcement, not granular SSH access control and auditing.

Authoritative Links:

AWS Systems Manager Session Manager:https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager.html
IAM Roles for EC2:https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_use_ec2.html
AmazonSSMManagedInstanceCore Policy:https://docs.aws.amazon.com/systems-manager/latest/userguide/security-iam-awsmanpol.html

Therefore, option D provides the most secure and auditable solution for managing SSH access to EC2 instances, aligning with security best practices.

Question: 66

A company that uses AWS Organizations allows developers to experiment on AWS. As part of the landing zone that the company has deployed, developers use their company email address to request an account. The company wants to ensure that developers are not launching costly services or running services unnecessarily. The company must give developers a fixed monthly budget to limit their AWS costs.

Which combination of steps will meet these requirements? (Choose three.)

A.Create an SCP to set a fixed monthly account usage limit. Apply the SCP to the developer accounts.

B.Use AWS Budgets to create a fixed monthly budget for each developer's account as part of the account creation process.

C.Create an SCP to deny access to costly services and components. Apply the SCP to the developer accounts.

D.Create an IAM policy to deny access to costly services and components. Apply the IAM policy to the developer accounts.

E.Create an AWS Budgets alert action to terminate services when the budgeted amount is reached. Configure the action to terminate all services.

F.Create an AWS Budgets alert action to send an Amazon Simple Notification Service (Amazon SNS) notification when the budgeted amount is reached. Invoke an AWS Lambda function to terminate all services.

Answer: BCF

Explanation:

Here's a detailed justification for why options B, C, and F are the correct choices:

B. Use AWS Budgets to create a fixed monthly budget for each developer's account as part of the account creation process: AWS Budgets is designed precisely for this purpose – setting cost boundaries. It enables you to track actual vs. planned spending and provides alerts when spending approaches or exceeds the budget. This directly addresses the requirement of a fixed monthly budget for each developer. By incorporating the budget creation into the account creation process, it ensures that every developer account is automatically subject to cost constraints from the outset. https://aws.amazon.com/aws-cost-management/aws-budgets/

C. Create an SCP to deny access to costly services and components. Apply the SCP to the developer accounts: Service Control Policies (SCPs) are a powerful mechanism to enforce AWS account governance within an AWS Organization. They function as guardrails, defining the maximum permissions available to member accounts. By creating an SCP that denies access to specific costly services (e.g., very large EC2 instances, GPU-intensive resources, etc.), you can prevent developers from unintentionally or intentionally incurring significant charges, regardless of their IAM permissions within their own account.
https://docs.aws.amazon.com/organizations/latest/userguide/orgs_manage_policies_scps.html

F. Create an AWS Budgets alert action to send an Amazon Simple Notification Service (Amazon SNS) notification when the budgeted amount is reached. Invoke an AWS Lambda function to terminate all services: While AWS Budgets can send alerts, it doesn't natively terminate resources directly. Combining it with SNS and Lambda provides the necessary automation. When the budget threshold is reached, AWS Budgets triggers an SNS notification. This notification, in turn, invokes a Lambda function that is programmed to systematically terminate the resources within the developer's account to prevent further cost overruns. This is the best practice to programatically shut down all resources based on budget utilization.
https://aws.amazon.com/premiumsupport/knowledge-center/terminate-ec2-when-budget-exceeded/

Here's why the other options are not ideal:

A. Create an SCP to set a fixed monthly account usage limit. Apply the SCP to the developer accounts: SCPs cannot directly enforce spending limits in a way that is useful. SCPs mainly focus on what actions (API calls) are allowed or denied. They don't inherently track or limit usage.

D. Create an IAM policy to deny access to costly services and components. Apply the IAM policy to the

developer accounts: While IAM policies can restrict access, they are less effective than SCPs in an AWS Organizations environment. Developers could potentially bypass IAM policies by creating roles with broader permissions (if they have sufficient initial access). SCPs, on the other hand, act as an organization-level guardrail that cannot be overridden by individual accounts.

E. Create an AWS Budgets alert action to terminate services when the budgeted amount is reached. Configure the action to terminate all services: AWS Budget alert actions do not support direct termination of services. Integration with SNS and Lambda are required.

Question: 67

A company has applications in an AWS account that is named Source. The account is in an organization in AWS Organizations. One of the applications uses AWS Lambda functions and stores inventory data in an Amazon Aurora database. The application deploys the Lambda functions by using a deployment package. The company has configured automated backups for Aurora.

The company wants to migrate the Lambda functions and the Aurora database to a new AWS account that is named Target. The application processes critical data, so the company must minimize downtime.

Which solution will meet these requirements?

A.Download the Lambda function deployment package from the Source account. Use the deployment package and create new Lambda functions in the Target account. Share the automated Aurora DB cluster snapshot with the Target account.

B.Download the Lambda function deployment package from the Source account. Use the deployment package and create new Lambda functions in the Target account. Share the Aurora DB cluster with the Target account by using AWS Resource Access Manager AWS RAM). Grant the Target account permission to clone the Aurora DB cluster.

C.Use AWS Resource Access Manager (AWS RAM) to share the Lambda functions and the Aurora DB cluster with the Target account. Grant the Target account permission to clone the Aurora DB cluster.

D.Use AWS Resource Access Manager (AWS RAM) to share the Lambda functions with the Target account. Share the automated Aurora DB cluster snapshot with the Target account.

Answer: B

Explanation:

The requirement is to migrate Lambda functions and an Aurora database from a Source account to a Target account with minimal downtime. Option B provides the most suitable approach.

Firstly, Lambda functions can't be directly shared between accounts using AWS RAM. Instead, the deployment package from the Source account should be downloaded and used to recreate the Lambda functions in the Target account. This ensures the Lambda functions are properly deployed in the new environment, taking into account IAM roles and other configurations specific to the Target account. This approach avoids any unexpected permission issues that can arise from RAM-based sharing of compute resources.

Secondly, to minimize downtime during the Aurora database migration, sharing the Aurora DB cluster directly using AWS RAM and then granting the Target account permission to clone it is optimal. Sharing the DB cluster with RAM allows the Target account to create a clone without interrupting the source database. Cloning creates a point-in-time copy of the database. Once the clone is created and validated in the Target account, application traffic can be switched over, minimizing downtime. Sharing a snapshot instead of the cluster requires restoring the snapshot to the Target account, which would take significantly longer and increase downtime.

Option A is incorrect because sharing the snapshot increases the migration time. Options C and D are incorrect because Lambda functions cannot be directly shared via AWS RAM.

Therefore, option B combines the correct approach for migrating both Lambda functions and Aurora database with minimal downtime, aligning with the requirement to process critical data.

Relevant documentation:

AWS RAM:https://docs.aws.amazon.com/ram/latest/userguide/what-is.html
Aurora Cloning:
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Managing.Clone.html

Question: 68

A company runs a Python script on an Amazon EC2 instance to process data. The script runs every 10 minutes. The script ingests files from an Amazon S3 bucket and processes the files. On average, the script takes approximately 5 minutes to process each file The script will not reprocess a file that the script has already processed.

The company reviewed Amazon CloudWatch metrics and noticed that the EC2 instance is idle for approximately 40% of the time because of the file processing speed. The company wants to make the workload highly available and scalable. The company also wants to reduce long-term management overhead.

Which solution will meet these requirements MOST cost-effectively?

A.Migrate the data processing script to an AWS Lambda function. Use an S3 event notification to invoke the Lambda function to process the objects when the company uploads the objects.

B.Create an Amazon Simple Queue Service (Amazon SQS) queue. Configure Amazon S3 to send event notifications to the SQS queue. Create an EC2 Auto Scaling group with a minimum size of one instance. Update the data processing script to poll the SQS queue. Process the S3 objects that the SQS message identifies.

C.Migrate the data processing script to a container image. Run the data processing container on an EC2 instance. Configure the container to poll the S3 bucket for new objects and to process the resulting objects.

D.Migrate the data processing script to a container image that runs on Amazon Elastic Container Service (Amazon ECS) on AWS Fargate. Create an AWS Lambda function that calls the Fargate RunTaskAPI operation when the container processes the file. Use an S3 event notification to invoke the Lambda function.

Answer: A

Explanation:

Here's a detailed justification for why option A is the most cost-effective solution and why the other options are less suitable:

Why Option A is the Best Choice: Serverless Simplicity and Cost Efficiency

Option A, migrating the data processing script to an AWS Lambda function triggered by S3 events, is the most cost-effective and efficient solution because it leverages serverless architecture. Lambda eliminates the need to manage EC2 instances, reducing operational overhead significantly. S3 event notifications trigger the Lambda function immediately upon file upload, ensuring near real-time processing. The cost model of Lambda is pay-per-execution, meaning you only pay for the compute time used during file processing, making it more cost-effective than running a continuously available EC2 instance, especially given the 40% idle time. Lambda scales automatically to handle the workload, addressing the scalability requirement. By using S3 event notifications, the process is also highly available. The script's logic to prevent reprocessing eliminates duplicates.

Why Other Options Are Less Suitable:

Option B (SQS and EC2 Auto Scaling): While this option provides scalability and availability, it is more complex and expensive than Lambda. Managing an SQS queue and an EC2 Auto Scaling group introduces additional overhead. The EC2 instance would still experience periods of idle time, leading to wasted resources. Polling SQS is also inherently less efficient than event-driven execution.

Option C (Container on EC2): This option is similar to the original setup, but packaged in a container. It still necessitates managing an EC2 instance, which can be costly. The instance still might have idle time, negating cost optimization. It doesn't inherently improve availability or scalability without additional configuration like Auto Scaling.

Option D (ECS Fargate and Lambda): This option is the most complex and expensive. Using Fargate (serverless containers) is a good approach, but the added complexity of using another Lambda function to invoke Fargate's RunTask API is unnecessary. Also, triggering the process from S3 upload events directly to the Lambda function is far more efficient.

Authoritative Links:

AWS Lambda:https://aws.amazon.com/lambda/
Amazon S3 Event Notifications:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/EventNotifications.html
AWS Serverless Computing:https://aws.amazon.com/serverless/

Question: 69

A financial services company in North America plans to release a new online web application to its customers on AWS. The company will launch the application in the us-east-1 Region on Amazon EC2 instances. The application must be highly available and must dynamically scale to meet user traffic. The company also wants to implement a disaster recovery environment for the application in the us-west-1 Region by using active-passive failover.

Which solution will meet these requirements?

A.Create a VPC in us-east-1 and a VPC in us-west-1. Configure VPC peering. In the us-east-1 VPC, create an Application Load Balancer (ALB) that extends across multiple Availability Zones in both VPCs. Create an Auto Scaling group that deploys the EC2 instances across the multiple Availability Zones in both VPCs. Place the Auto Scaling group behind the ALB.

B.Create a VPC in us-east-1 and a VPC in us-west-1. In the us-east-1 VPC, create an Application Load Balancer (ALB) that extends across multiple Availability Zones in that VPC. Create an Auto Scaling group that deploys the EC2 instances across the multiple Availability Zones in the us-east-1 VPC. Place the Auto Scaling group behind the ALSet up the same configuration in the us-west-1 VPC. Create an Amazon Route 53 hosted zone.
Create separate records for each ALEnable health checks to ensure high availability between Regions.

C.Create a VPC in us-east-1 and a VPC in us-west-1. In the us-east-1 VPC, create an Application Load Balancer (ALB) that extends across multiple Availability Zones in that VPCreate an Auto Scaling group that deploys the EC2 instances across the multiple Availability Zones in the us-east-1 VPPlace the Auto Scaling group behind the ALB. Set up the same configuration in the us-west-1 VPCreate an Amazon Route 53 hosted zone. Create separate records for each ALB. Enable health checks and configure a failover routing policy for each record.

D.Create a VPC in us-east-1 and a VPC in us-west-1. Configure VPC peering. In the us-east-1 VPC, create an Application Load Balancer (ALB) that extends across multiple Availability Zones in both VPCs. Create an Auto Scaling group that deploys the EC2 instances across the multiple Availability Zones in both VPCs. Place the Auto Scaling group behind the ALB. Create an Amazon Route 53 hosted zone. Create a record for the ALB.

Answer: C

Explanation:

Option C is the most suitable solution because it effectively addresses the requirements of high availability, dynamic scaling, and active-passive failover across two AWS regions.

Here's why:

Regional Isolation: Creating separate VPCs in us-east-1 and us-west-1 ensures regional isolation, a fundamental aspect of disaster recovery.
High Availability within a Region: Using an ALB and Auto Scaling group spanning multiple Availability Zones

within each VPC guarantees high availability and scalability within each region. The ALB distributes traffic across healthy EC2 instances in the Auto Scaling group.
Active-Passive Failover: Configuring an identical setup in us-west-1 provides the passive disaster recovery environment. It is ready to take over if us-east-1 becomes unavailable.
Route 53 Failover: Amazon Route 53 is used to implement the active-passive failover. By creating separate records for each ALB (one for us-east-1 and one for us-west-1) and enabling health checks and a failover routing policy, Route 53 automatically redirects traffic to the us-west-1 ALB if the health checks for the us-east-1 ALB fail.

Let's look at why the other options are not as optimal:

Option A & D: VPC peering across regions can be complex to manage and might introduce latency. Also, ALBs and Auto Scaling groups cannot span across multiple AWS Regions.
Option B: While it sets up ALB and ASG correctly within each region, it doesn't specify the critical part of using Route 53 failover routing policy with health checks to automate the failover process between the regions. It also mentions creating separate records for each ALB without specifying that they are used with failover routing, meaning the application wouldn't automatically failover in the event of a disaster.

Authoritative Links:

AWS Route 53 Failover Routing:https://docs.aws.amazon.com/Route53/latest/DeveloperGuide/dns-failover.html
Application Load Balancer:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html
Auto Scaling Group:https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-groups.html

Question: 70

A company has an environment that has a single AWS account. A solutions architect is reviewing the environment to recommend what the company could improve specifically in terms of access to the AWS Management Console. The company's IT support workers currently access the console for administrative tasks, authenticating with named IAM users that have been mapped to their job role.

The IT support workers no longer want to maintain both their Active Directory and IAM user accounts. They want to be able to access the console by using their existing Active Directory credentials. The solutions architect is using AWS IAM Identity Center (AWS Single Sign-On) to implement this functionality.

Which solution will meet these requirements MOST cost-effectively?

A.Create an organization in AWS Organizations. Turn on the IAM Identity Center feature in Organizations. Create and configure a directory in AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) with a two-way trust to the company's on-premises Active Directory. Configure IAM Identity Center and set the AWS Managed Microsoft AD directory as the identity source. Create permission sets and map them to the existing groups within the AWS Managed Microsoft AD directory.

B.Create an organization in AWS Organizations. Turn on the IAM Identity Center feature in Organizations. Create and configure an AD Connector to connect to the company's on-premises Active Directory. Configure IAM Identity Center and select the AD Connector as the identity source. Create permission sets and map them to the existing groups within the company's Active Directory.

C.Create an organization in AWS Organizations. Turn on all features for the organization. Create and configure a directory in AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) with a two-way trust to the company's on-premises Active Directory. Configure IAM Identity Center and select the AWS Managed Microsoft AD directory as the identity source. Create permission sets and map them to the existing groups within the AWS Managed Microsoft AD directory.

D.Create an organization in AWS Organizations. Turn on all features for the organization. Create and configure an AD Connector to connect to the company's on-premises Active Directory. Configure IAM Identity Center and set the AD Connector as the identity source. Create permission sets and map them to the existing groups within the company's Active Directory.

Answer: D

Explanation:

The most cost-effective solution for enabling IT support workers to use their existing Active Directory credentials to access the AWS Management Console through AWS IAM Identity Center is option D.

Here's why:

AWS Organizations and IAM Identity Center: AWS Organizations is necessary to enable IAM Identity Center, as it provides the organizational structure needed for centralized identity management. Turning on "all features" in Organizations is required to enable delegated administration and trust relationships needed for cross-account access.

AD Connector vs. AWS Managed Microsoft AD: AD Connector is a directory gateway that allows you to connect to your existing on-premises Active Directory without replicating the directory in AWS. AWS Managed Microsoft AD, on the other hand, creates a fully managed Active Directory domain in the AWS cloud.

Because the company already has an on-premises Active Directory, using AD Connector is more cost-effective as it avoids the cost of running and maintaining a separate Active Directory domain in AWS. AD Connector leverages the existing on-premises AD infrastructure.

IAM Identity Center Configuration: IAM Identity Center can be configured to use the AD Connector as an identity source, allowing users to authenticate using their existing Active Directory credentials.

Permission Sets and Group Mapping: Permission sets define the level of access granted to users. These permission sets can be mapped to existing groups in the company's Active Directory, ensuring that users receive the appropriate permissions based on their roles.

Options A and C are less cost-effective because they involve creating and managing a new AWS Managed Microsoft AD directory, which adds unnecessary infrastructure and management overhead when the company already has an on-premises Active Directory. Option B is incorrect because it doesn't enable "all features" in AWS Organizations, which are required to enable trust relationships and delegated administration.

In summary, AD Connector provides a cost-effective bridge between the on-premises Active Directory and AWS IAM Identity Center, allowing users to authenticate with their existing credentials without duplicating the directory in AWS.

Here are some authoritative links for further research:

AWS IAM Identity Center:https://aws.amazon.com/iam/identity/
AWS Directory Service:https://aws.amazon.com/directoryservice/
AD Connector:https://docs.aws.amazon.com/directoryservice/latest/admin-guide/directory_ad_connector.html
AWS Managed Microsoft AD:https://docs.aws.amazon.com/directoryservice/latest/admin-guide/directory_microsoft_ad.html
AWS Organizations:https://aws.amazon.com/organizations/

Question: 71

A video streaming company recently launched a mobile app for video sharing. The app uploads various files to an Amazon S3 bucket in the us-east-1 Region. The files range in size from 1 GB to 10 GB.

Users who access the app from Australia have experienced uploads that take long periods of time. Sometimes the files fail to completely upload for these users. A solutions architect must improve the app's performance for these uploads.

Which solutions will meet these requirements? (Choose two.)

A.Enable S3 Transfer Acceleration on the S3 bucket. Configure the app to use the Transfer Acceleration

endpoint for uploads.

B.Configure an S3 bucket in each Region to receive the uploads. Use S3 Cross-Region Replication to copy the files to the distribution S3 bucket.

C.Set up Amazon Route 53 with latency-based routing to route the uploads to the nearest S3 bucket Region.

D.Configure the app to break the video files into chunks. Use a multipart upload to transfer files to Amazon S3.

E.Modify the app to add random prefixes to the files before uploading.

Answer: AD

Explanation:

Here's a detailed justification for choosing options A and D as the best solutions to improve video upload performance for users in Australia accessing an S3 bucket in us-east-1, addressing the challenges of latency and reliability associated with large file uploads across long distances:

Option A: Enable S3 Transfer Acceleration

Justification: S3 Transfer Acceleration is designed to improve transfer speeds for data being uploaded to S3 buckets over long distances. It leverages Amazon CloudFront's globally distributed edge locations. When a user uploads to the Transfer Acceleration endpoint, the data is routed to the nearest edge location. The edge location then uses optimized network paths to upload the data to the S3 bucket in the specified region (us-east-1 in this case). This minimizes latency and maximizes the available bandwidth, resulting in faster uploads.

Cloud Computing Concept: This solution directly addresses network latency, a common problem with geographically distributed users. The architecture leverages the proximity of CloudFront edge locations to users to significantly reduce the distance data has to travel over the public internet before entering the AWS network.

Authoritative Link:https://aws.amazon.com/s3/transfer-acceleration/

Option D: Configure Multipart Uploads

Justification: Multipart upload allows you to upload a single object as a set of parts. Each part can be uploaded independently, and if a part upload fails, you can retry the upload of that specific part without re-uploading the entire file. This significantly increases the reliability of uploads, especially for large files and unreliable network connections. Furthermore, multipart upload can also improve upload speed as it allows concurrent uploads of different parts of the file. The application splits the large video files (1-10 GB) into smaller, manageable chunks.

Cloud Computing Concept: This solution addresses both reliability and performance. Multipart uploads mitigate the risk of a complete upload failure due to intermittent network issues. The parallelism offered by the ability to upload parts concurrently also improves throughput.

Authoritative Link:https://docs.aws.amazon.com/AmazonS3/latest/userguide/mpuoverview.html

Why other options are not the best:

Option B: While configuring an S3 bucket in each Region could potentially improve upload performance, it introduces significant complexity in managing multiple buckets and ensuring data consistency. S3 Cross-Region Replication adds latency and cost and doesn't necessarily improve the initial upload performance for Australian users.

Option C: Latency-based routing with Route 53 directs traffic to the closest AWS Region. While this might seem relevant, users are specifically uploading to a bucket in us-east-1, according to the problem description.

Route 53 would not help here as it only routes the initial DNS resolution, and the application would still ultimately upload the data to us-east-1. Additionally, this option doesn't address the issue of file size.

Option E: Adding random prefixes to filenames is beneficial for distributing objects evenly across S3 partitions to avoid hot partitions. However, it primarily improves read performance, not upload performance, and does not address the latency or reliability issues users in Australia are experiencing.

In summary, enabling S3 Transfer Acceleration optimizes the network path and reduces latency, while using multipart upload improves reliability and can increase upload speed by enabling parallel uploads of smaller chunks, which are paramount for the video streaming company facing upload challenges. These two solutions address the core problems of latency and unreliable connections over long distances.

Question: 72

An application is using an Amazon RDS for MySQL Multi-AZ DB instance in the us-east-1 Region. After a failover test, the application lost the connections to the database and could not re-establish the connections. After a restart of the application, the application re-established the connections.

A solutions architect must implement a solution so that the application can re-establish connections to the database without requiring a restart.

Which solution will meet these requirements?

A.Create an Amazon Aurora MySQL Serverless v1 DB instance. Migrate the RDS DB instance to the Aurora Serverless v1 DB instance. Update the connection settings in the application to point to the Aurora reader endpoint.

B.Create an RDS proxy. Configure the existing RDS endpoint as a target. Update the connection settings in the application to point to the RDS proxy endpoint.

C.Create a two-node Amazon Aurora MySQL DB cluster. Migrate the RDS DB instance to the Aurora DB cluster. Create an RDS proxy. Configure the existing RDS endpoint as a target. Update the connection settings in the application to point to the RDS proxy endpoint.

D.Create an Amazon S3 bucket. Export the database to Amazon S3 by using AWS Database Migration Service (AWS DMS). Configure Amazon Athena to use the S3 bucket as a data store. Install the latest Open Database Connectivity (ODBC) driver for the application. Update the connection settings in the application to point to the Athena endpoint

Answer: B

Explanation:

The correct answer is B. Here's why:

Problem: The application experiences connection loss after an RDS Multi-AZ failover and requires a restart to reconnect. This indicates a failure in the application's ability to automatically handle failover events.

Solution B (RDS Proxy): RDS Proxy sits between the application and the RDS database. It manages database connections, multiplexing them and reusing them efficiently. Critically, RDS Proxy is designed to automatically handle failovers. When a failover occurs in the underlying RDS instance, RDS Proxy intercepts the connection loss and automatically reconnects to the new primary instance without requiring any changes on the application side. The application only connects to the proxy endpoint, which remains consistent regardless of the failover. This eliminates the need for application restarts.

Why other options are incorrect:

A (Aurora Serverless v1): While Aurora Serverless v1 offers automatic scaling, it doesn't inherently solve the connection interruption issue during failover. Although it reduces the probability of a failover (due to the underlying architecture), a failover can still occur. The application still needs a mechanism to handle connection disruption. Switching to the reader endpoint after failover might help, but the prompt implies that the solution should allow to re-establish connections to the database without requiring a restart.

C (Aurora DB Cluster + RDS Proxy): While using Aurora DB cluster is an excellent choice for high availability and disaster recovery in relational databases, adding RDS proxy on top of it in this scenario is redundant.

Aurora cluster already provides high availability by having a primary instance and one or more Aurora Replicas, but Aurora Proxy, in this case, does not offer any additional connection re-establishment capabilities in case of failover. It introduces unnecessary complexity and cost.

D (S3, DMS, Athena, ODBC): This approach completely changes the application's database access pattern. It moves the data to S3, then makes it accessible using Athena. It is not an appropriate solution to maintaining MySQL RDS connection upon failover, and completely out of scope.

Supporting Concepts and Links:

RDS Proxy: Improves application availability and reduces failover times for RDS databases.

https://aws.amazon.com/rds/proxy/

RDS Multi-AZ: Provides high availability for RDS databases through automatic failover to a standby replica.

https://aws.amazon.com/rds/features/multi-az/

Connection Pooling: RDS Proxy manages connection pooling, which is crucial for handling failovers gracefully. https://docs.aws.amazon.com/rds/proxy/latest/userguide/rds-proxy.how-it-works.html

In summary, RDS Proxy offers a straightforward and effective solution to the connection disruption problem caused by RDS failovers, allowing the application to reconnect automatically without requiring a restart.

Question: 73

A company is building a solution in the AWS Cloud. Thousands or devices will connect to the solution and send data. Each device needs to be able to send and receive data in real time over the MQTT protocol. Each device must authenticate by using a unique X.509 certificate.

Which solution will meet these requirements with the LEAST operational overhead?

A.Set up AWS IoT Core. For each device, create a corresponding Amazon MQ queue and provision a certificate. Connect each device to Amazon MQ.

B.Create a Network Load Balancer (NLB) and configure it with an AWS Lambda authorizer. Run an MQTT broker on Amazon EC2 instances in an Auto Scaling group. Set the Auto Scaling group as the target for the NLConnect each device to the NLB.

C.Set up AWS IoT Core. For each device, create a corresponding AWS IoT thing and provision a certificate. Connect each device to AWS IoT Core.

D.Set up an Amazon API Gateway HTTP API and a Network Load Balancer (NLB). Create integration between API Gateway and the NLB. Configure a mutual TLS certificate authorizer on the HTTP API. Run an MQTT broker on an Amazon EC2 instance that the NLB targets. Connect each device to the NLB.

Answer: C

Explanation:

The correct answer is C because it leverages AWS IoT Core, a managed service specifically designed for IoT solutions. AWS IoT Core natively supports the MQTT protocol, device authentication via X.509 certificates, and scalability to handle thousands of devices, minimizing operational overhead. Each device is represented as an "AWS IoT thing," simplifying device management and control.

Option A is incorrect because Amazon MQ is a managed message broker service primarily designed for application-to-application messaging, not for handling direct device connections and MQTT protocol natively.
Creating individual Amazon MQ queues for each device would be operationally complex and inefficient.

Option B involves setting up a custom MQTT broker on EC2 instances behind a Network Load Balancer, which significantly increases operational overhead compared to using a managed service like AWS IoT Core.

Managing the MQTT broker, scaling the EC2 instances, and handling authentication through Lambda authorizers would be more complex.

Option D is also incorrect. While API Gateway with mutual TLS can handle certificate authentication, integrating it with a Network Load Balancer and an EC2-based MQTT broker again introduces significant operational overhead. This approach is not optimized for IoT device management and real-time data streaming via MQTT. AWS IoT Core offers direct support for both device certificates and MQTT.

Therefore, utilizing AWS IoT Core is the most efficient solution as it provides a managed platform for device connectivity, message routing, and security tailored specifically for IoT scenarios, minimizing the effort to establish a secure and scalable environment for real-time data exchange.

Relevant links:

AWS IoT Core: https://aws.amazon.com/iot-core/
MQTT Protocol: http://mqtt.org/
X.509 Certificates: https://www.ssl.com/article/introduction-to-x-509-certificates/

Question: 74

A company is running several workloads in a single AWS account. A new company policy states that engineers can provision only approved resources and that engineers must use AWS CloudFormation to provision these resources.
A solutions architect needs to create a solution to enforce the new restriction on the IAM role that the engineers use for access.

What should the solutions architect do to create the solution?

A.Upload AWS CloudFormation templates that contain approved resources to an Amazon S3 bucket. Update the IAM policy for the engineers' IAM role to only allow access to Amazon S3 and AWS CloudFormation. Use AWS CloudFormation templates to provision resources.

B.Update the IAM policy for the engineers' IAM role with permissions to only allow provisioning of approved resources and AWS CloudFormation. Use AWS CloudFormation templates to create stacks with approved resources.

C.Update the IAM policy for the engineers' IAM role with permissions to only allow AWS CloudFormation actions. Create a new IAM policy with permission to provision approved resources, and assign the policy to a new IAM service role. Assign the IAM service role to AWS CloudFormation during stack creation.

D.Provision resources in AWS CloudFormation stacks. Update the IAM policy for the engineers' IAM role to only allow access to their own AWS CloudFormation stack.

Answer: C

Explanation:

The correct answer is C. Here's a detailed justification:

The requirement is to restrict engineers to provisioning only approved resources using CloudFormation. Option C achieves this goal by decoupling the engineers' access to CloudFormation itself from the permission to provision specific resources.

1. **Restricting CloudFormation Actions:** Updating the engineers' IAM role to only allow AWS CloudFormation actions ensures they can only interact with CloudFormation and cannot directly create resources outside of it. This is crucial for enforcing the "CloudFormation only" policy.

2. **IAM Service Role for Approved Resources:** Creating a separate IAM service role with permissions to provision approved resources allows CloudFormation, acting on behalf of the engineers, to create the required infrastructure. This aligns with the principle of least privilege, as the engineers' role doesn't need direct access to create resources.

3. **Delegating Authority:** Assigning the IAM service role to CloudFormation during stack creation is a critical step. It gives CloudFormation temporary permissions to provision the resources defined in the template. This is an example of role-based access control, where CloudFormation assumes a role with specific permissions.

4. **Separation of Duties:** This approach separates the responsibility of who can use CloudFormation (the engineers) from what CloudFormation can provision (the approved resources via the service role). This separation promotes security and auditability.

Options A and B are not ideal because:

Option A: Restricting engineers to S3 and CloudFormation doesn't enforce the 'approved resources' rule. Engineers could still create templates with unapproved resources.

Option B: Updating the IAM policy for engineers to directly allow provisioning of approved resources bypasses the use of IAM roles for CloudFormation's use, meaning their IAM Role would need the create permissions negating the restrictions. This directly contradicts the 'approved resources' constraint, as engineers could directly create resources outside CloudFormation using their IAM role.

Option D is not ideal because restricting access to their own CloudFormation stack does not address the requirement that engineers can only provision approved resources.

Supporting Concepts and Links:

IAM Roles:https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html
IAM Policies:https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies.html
CloudFormation Service Role:https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/using-iam-service-role.html
Principle of Least Privilege:https://docs.aws.amazon.com/IAM/latest/UserGuide/best-practices.html

Question: 75

A solutions architect is designing the data storage and retrieval architecture for a new application that a company will be launching soon. The application is designed to ingest millions of small records per minute from devices all around the world. Each record is less than 4 KB in size and needs to be stored in a durable location where it can be retrieved with low latency. The data is ephemeral and the company is required to store the data for 120 days only, after which the data can be deleted.

The solutions architect calculates that, during the course of a year, the storage requirements would be about 10-15 TB.

Which storage strategy is the MOST cost-effective and meets the design requirements?

A.Design the application to store each incoming record as a single .csv file in an Amazon S3 bucket to allow for indexed retrieval. Configure a lifecycle policy to delete data older than 120 days.

B.Design the application to store each incoming record in an Amazon DynamoDB table properly configured for the scale. Configure the DynamoDB Time to Live (TTL) feature to delete records older than 120 days.

C.Design the application to store each incoming record in a single table in an Amazon RDS MySQL database. Run a nightly cron job that runs a query to delete any records older than 120 days.

D.Design the application to batch incoming records before writing them to an Amazon S3 bucket. Update the metadata for the object to contain the list of records in the batch and use the Amazon S3 metadata search feature to retrieve the data. Configure a lifecycle policy to delete the data after 120 days.

Answer: B

Explanation:

Here's a detailed justification for why option B is the most cost-effective storage strategy for this scenario:

Option B, using Amazon DynamoDB with TTL, is the most suitable choice due to its high scalability, low latency, and cost-effectiveness for handling the specific requirements of this application.

Scalability and Low Latency: DynamoDB is a fully managed NoSQL database service that can handle millions of requests per second with consistent low latency. The application needs to ingest millions of records per minute and retrieve them with low latency, which DynamoDB is designed to handle.

Cost-Effectiveness: DynamoDB's on-demand capacity mode allows you to pay only for what you use, making it cost-effective for variable workloads. Given the relatively small record size and defined retention period, DynamoDB is usually more affordable than alternatives.

Ephemeral Data Handling: The requirement to store data for only 120 days is perfectly addressed by DynamoDB's Time to Live (TTL) feature. TTL automatically deletes items from the table after a specified expiry time, simplifying data management and reducing storage costs by automatically removing expired data.

Alternative A (S3 with CSVs): Storing each record as a separate CSV in S3 is not cost-effective or performant. Managing millions of small files in S3 can lead to higher operational overhead and increased request costs. While S3 Lifecycle Policies can handle deletion, retrieving specific records from individual CSV files can be slow and inefficient, especially if indexed retrieval is needed.

Alternative C (RDS MySQL): Using Amazon RDS MySQL might not be cost-effective for this scale of data ingestion. While MySQL is a relational database, handling millions of small records per minute can be challenging and expensive in terms of database resources. Running nightly cron jobs to delete data adds operational complexity.

Alternative D (Batched S3 with Metadata): While batching records in S3 improves storage efficiency, using S3 metadata search is not optimal for retrieving specific records within a batch. It would require complex parsing and filtering, adding to the application's complexity.

In summary, DynamoDB provides the best balance of scalability, low latency, cost-effectiveness, and ease of management with its TTL feature, making it the most suitable option for handling ephemeral data in this scenario.

Relevant Documentation:

Amazon DynamoDB Pricing
Using Time To Live (TTL) to Expire Items
Amazon S3 Lifecycle

Question: 76

A retail company is hosting an ecommerce website on AWS across multiple AWS Regions. The company wants the website to be operational at all times for online purchases. The website stores data in an Amazon RDS for MySQL DB instance.

Which solution will provide the HIGHEST availability for the database?

A.Configure automated backups on Amazon RDS. In the case of disruption, promote an automated backup to be a standalone DB instance. Direct database traffic to the promoted DB instance. Create a replacement read replica that has the promoted DB instance as its source.

B.Configure global tables and read replicas on Amazon RDS. Activate the cross-Region scope. In the case of disruption, use AWS Lambda to copy the read replicas from one Region to another Region.

C.Configure global tables and automated backups on Amazon RDS. In the case of disruption, use AWS Lambda to copy the read replicas from one Region to another Region.

D.Configure read replicas on Amazon RDS. In the case of disruption, promote a cross-Region and read replica to be a standalone DB instance. Direct database traffic to the promoted DB instance. Create a replacement read replica that has the promoted DB instance as its source.

Answer: D

Explanation:

Here's a detailed justification for why option D is the best solution for achieving the highest database availability for the retail company's e-commerce website, compared to the other options.

Option D leverages cross-Region read replicas in Amazon RDS to provide a robust disaster recovery and high availability solution. A read replica in a different region acts as a standby database that is constantly replicating data from the primary database. In case of an outage in the primary region, the cross-region read replica can be promoted to a standalone database instance, minimizing downtime. This failover process ensures continued availability of the database for the website's online purchases. Directing traffic to the promoted instance restores the application's database connectivity. Creating a new read replica based on the promoted instance re-establishes the high availability setup.

Option A, while utilizing automated backups, requires restoring from a backup in case of a disruption. The time needed for restoration is significantly higher compared to promoting a read replica. This longer recovery time negatively impacts the website's availability, making it a less optimal solution.

Option B mentions global tables and read replicas, along with Lambda for copying read replicas. While global tables offer automatic replication, it is only available for Amazon Aurora, not MySQL. The use of Lambda to copy replicas is also unclear and inefficient for disaster recovery.

Option C suffers from the same limitations as option B regarding global tables for MySQL and the inefficient use of Lambda.

Therefore, only option D directly addresses the requirement for high availability by using cross-region read replicas for quick failover, making it the best choice.

Key Concepts and Why They Matter:

Read Replicas: Read replicas provide read-only copies of the primary database, distributing read workload and enhancing read performance. Crucially, a cross-region read replica serves as a hot standby for disaster recovery.

Cross-Region Replication: Replicating data to another geographical region protects against regional failures, ensuring business continuity.

Promotion of Read Replica: Converting a read replica into a standalone database instance is a standard procedure in RDS to recover from primary instance failures or regional outages.

Recovery Time Objective (RTO): This refers to the acceptable duration of downtime after an outage. Promoting a read replica offers a significantly lower RTO compared to restoring from backups.

High Availability (HA): This refers to a system's ability to remain operational during failures. Replicating your databases to other locations reduces the risk of data loss and data availability.

Authoritative Links for Further Research:

Amazon RDS Read Replicas:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.ReadReplicas.html
Amazon RDS Cross-Region Read Replicas:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_ReadRepl.html#USER_ReadRepl.XRgn **Amazon RDS Disaster Recovery:**https://aws.amazon.com/rds/disaster-recovery/

Question: 77

Example Corp. has an on-premises data center and a VPC named VPC A in the Example Corp. AWS account. The on-premises network connects to VPC A through an AWS Site-To-Site VPN. The on-premises servers can properly access VPC A. Example Corp. just acquired AnyCompany, which has a VPC named VPC B. There is no IP address

overlap among these networks. Example Corp. has peered VPC A and VPC B.

Example Corp. wants to connect from its on-premise servers to VPC B. Example Corp. has properly set up the network ACL and security groups.

Which solution will meet this requirement with the LEAST operational effort?

A.Create a transit gateway. Attach the Site-to-Site VPN, VPC A, and VPC B to the transit gateway. Update the transit gateway route tables for all networks to add IP range routes for all other networks.

B.Create a transit gateway. Create a Site-to-Site VPN connection between the on-premises network and VPC B, and connect the VPN connection to the transit gateway. Add a route to direct traffic to the peered VPCs, and add an authorization rule to give clients access to the VPCs A and B.

C.Update the route tables for the Site-to-Site VPN and both VPCs for all three networks. Configure BGP propagation for all three networks. Wait for up to 5 minutes for BGP propagation to finish.

D.Modify the Site-to-Site VPN's virtual private gateway definition to include VPC A and VPC B. Split the two routers of the virtual private getaway between the two VPCs.

Answer: A

Explanation:

The most efficient and scalable solution for connecting on-premises servers to VPC B, leveraging the existing VPN connection to VPC A and the VPC peering, is to use an AWS Transit Gateway (TGW).

Here's why option A is the best choice:

Centralized Routing: A TGW acts as a central hub for interconnecting VPCs and on-premises networks. This simplifies network management compared to managing individual route tables for each connection. **Simplified Routing:** By attaching the Site-to-Site VPN, VPC A, and VPC B to the TGW, you only need to update the TGW's route tables to enable communication between all networks. This involves adding routes for each network's IP address range to direct traffic to the appropriate destination.

Scalability: TGWs are designed to handle a large number of connections, making them suitable for growing and complex network environments.

No VPN Redundancy: Option B suggests creating another Site-to-Site VPN connection, which is redundant considering the existing VPN connection through VPC A.

Avoids Complex Route Table Management: Option C involves updating multiple route tables and configuring BGP propagation, which is more complex and time-consuming than using a TGW.

Limitations of Virtual Private Gateways: Option D is not viable. Virtual Private Gateways cannot span multiple VPCs in the way described.

Therefore, a TGW provides the least operational effort by centralizing routing and offering a scalable solution for interconnecting the on-premises network, VPC A, and VPC B.

Relevant Documentation:

AWS Transit Gateway:https://aws.amazon.com/transit-gateway/
Transit Gateway Routing:https://docs.aws.amazon.com/vpc/latest/tgw/tgw-route-tables.html

Question: 78

A company recently completed the migration from an on-premises data center to the AWS Cloud by using a replatforming strategy. One of the migrated servers is running a legacy Simple Mail Transfer Protocol (SMTP) service that a critical application relies upon. The application sends outbound email messages to the company's customers. The legacy SMTP server does not support TLS encryption and uses TCP port 25. The application can use SMTP only.

The company decides to use Amazon Simple Email Service (Amazon SES) and to decommission the legacy SMTP

server. The company has created and validated the SES domain. The company has lifted the SES limits.

What should the company do to modify the application to send email messages from Amazon SES?

A.Configure the application to connect to Amazon SES by using TLS Wrapper. Create an IAM role that has ses:SendEmail and ses:SendRawEmail permissions. Attach the IAM role to an Amazon EC2 instance.

B.Configure the application to connect to Amazon SES by using STARTTLS. Obtain Amazon SES SMTP credentials. Use the credentials to authenticate with Amazon SES.

C.Configure the application to use the SES API to send email messages. Create an IAM role that has ses:SendEmail and ses:SendRawEmail permissions. Use the IAM role as a service role for Amazon SES.

D.Configure the application to use AWS SDKs to send email messages. Create an IAM user for Amazon SES. Generate API access keys. Use the access keys to authenticate with Amazon SES.

Answer: B

Explanation:

The correct answer is B. Here's why:

The key problem is the legacy application only supports SMTP. Options C and D, which involve using the SES API or AWS SDKs, would require significant code changes to the application, which contradicts the stated constraint. Option A is incorrect because the legacy SMTP service doesn't support TLS, so a TLS wrapper won't work.

Option B allows the company to use SES while working within the application's limitations. STARTTLS allows for establishing an unencrypted connection first and then upgrading to an encrypted connection using TLS. This is beneficial because the legacy application can initiate the connection and then TLS is initiated, addressing security concerns raised by not using encryption on port 25. By obtaining SES SMTP credentials, the application can authenticate with SES and relay emails through Amazon's service, effectively decommissioning the legacy SMTP server.

IAM roles are primarily used for AWS resources to access other AWS resources and IAM users are mainly used for individuals, external entities, and services that operate outside of the AWS environment.

Further Reading:

Amazon SES SMTP Interface
Securing Email with Transport Layer Security (TLS)

Question: 79

A company recently acquired several other companies. Each company has a separate AWS account with a different billing and reporting method. The acquiring company has consolidated all the accounts into one organization in AWS Organizations. However, the acquiring company has found it difficult to generate a cost report that contains meaningful groups for all the teams.

The acquiring company’s finance team needs a solution to report on costs for all the companies through a self-managed application.

Which solution will meet these requirements?

A.Create an AWS Cost and Usage Report for the organization. Define tags and cost categories in the report. Create a table in Amazon Athena. Create an Amazon QuickSight dataset based on the Athena table. Share the dataset with the finance team.

B.Create an AWS Cost and Usage Report for the organization. Define tags and cost categories in the report. Create a specialized template in AWS Cost Explorer that the finance department will use to build reports.

C.Create an Amazon QuickSight dataset that receives spending information from the AWS Price List Query API. Share the dataset with the finance team.

D.Use the AWS Price List Query API to collect account spending information. Create a specialized template in AWS Cost Explorer that the finance department will use to build reports.

Answer: A

Explanation:

The most effective solution for generating cost reports with meaningful groupings from a consolidated AWS Organizations setup, accessible through a self-managed application for the finance team, is option A. Here's why:

AWS Cost and Usage Report (CUR): The CUR provides the most comprehensive and granular data on AWS costs and usage. It delivers detailed line items, enabling precise analysis.
https://docs.aws.amazon.com/cur/latest/userguide/what-is-cur.html

Tags and Cost Categories: These features allow for grouping costs based on various dimensions, such as company, team, project, or service. They provide the "meaningful groups" the finance team needs.
https://docs.aws.amazon.com/awsaccountbilling/latest/aboutv2/cost-categories.html

Amazon Athena: Athena allows querying the CUR data stored in S3 using standard SQL. This enables extracting, transforming, and loading (ETL) the data into a format suitable for reporting.
https://aws.amazon.com/athena/

Amazon QuickSight: QuickSight is a business intelligence (BI) service that can connect to Athena and create interactive dashboards and reports. Sharing the QuickSight dataset allows the finance team to create their own self-managed reports. https://aws.amazon.com/quicksight/

Option B is less flexible and self-managed. Cost Explorer templates are predefined and might not fully meet the finance team's specific reporting requirements.

Options C and D are insufficient. The AWS Price List Query API provides pricing information but doesn't offer detailed cost and usage data. It lacks the granularity required for generating cost reports with meaningful groupings based on tags and cost categories. Further, relying solely on the Price List API misses crucial details such as discounts and actual usage costs.

Question: 80

A company runs an IoT platform on AWS. IoT sensors in various locations send data to the company's Node.js API servers on Amazon EC2 instances running behind an Application Load Balancer. The data is stored in an Amazon RDS MySQL DB instance that uses a 4 TB General Purpose SSD volume.

The number of sensors the company has deployed in the field has increased over time, and is expected to grow significantly. The API servers are consistently overloaded and RDS metrics show high write latency.

Which of the following steps together will resolve the issues permanently and enable growth as new sensors are provisioned, while keeping this platform cost-efficient? (Choose two.)

A.Resize the MySQL General Purpose SSD storage to 6 TB to improve the volume's IOPS.

B.Re-architect the database tier to use Amazon Aurora instead of an RDS MySQL DB instance and add read replicas.

C.Leverage Amazon Kinesis Data Streams and AWS Lambda to ingest and process the raw data.

D.Use AWS X-Ray to analyze and debug application issues and add more API servers to match the load.

E.Re-architect the database tier to use Amazon DynamoDB instead of an RDS MySQL DB instance.

Answer: CE

Explanation:

The correct answer is CE. Here's why:

C. Leverage Amazon Kinesis Data Streams and AWS Lambda to ingest and process the raw data: The API servers are overloaded due to the increasing number of sensors sending data. Introducing Kinesis Data Streams acts as a buffer, decoupling the sensors from the API servers and RDS database. Sensors can stream data to Kinesis, which then invokes Lambda functions to process the data in batches before writing to the database. This offloads the processing burden from the API servers and allows them to handle requests more efficiently. This aligns with the principle of asynchronous processing and microservices architecture, improving scalability and resilience. https://aws.amazon.com/kinesis/data-streams/

E. Re-architect the database tier to use Amazon DynamoDB instead of an RDS MySQL DB instance: The high write latency in RDS suggests that the relational database is struggling to handle the increasing write load from the IoT platform. DynamoDB, a NoSQL database, is designed for high-volume, low-latency writes.

Switching to DynamoDB is ideal for time-series data from IoT sensors as DynamoDB's scalable, distributed architecture can handle the increased throughput, resolving the high write latency issue. This decision aligns with choosing the right database for the workload and recognizing the limitations of relational databases for high-write scenarios. https://aws.amazon.com/dynamodb/

Here's why the other options are less suitable:

A. Resize the MySQL General Purpose SSD storage to 6 TB to improve the volume's IOPS: While increasing the storage size does increase IOPS for General Purpose SSD, this is not the most scalable or cost-effective solution. It might temporarily alleviate the issue but won't fundamentally address the architectural bottleneck of overloading the API servers and the limitations of MySQL for high-volume writes.

B. Re-architect the database tier to use Amazon Aurora instead of an RDS MySQL DB instance and add read replicas: Aurora provides performance improvements over MySQL, but it's still a relational database. The underlying issue is the relational database schema's suitability for high-volume IoT data ingestion and the bottleneck at the API server level. Read replicas address read scalability, not the write bottleneck.

D. Use AWS X-Ray to analyze and debug application issues and add more API servers to match the load: X-Ray helps identify bottlenecks, but it's more of a debugging tool. Adding more API servers without addressing the underlying architecture will only scale the problem, potentially increasing costs without effectively resolving the write latency issue at the database level. You still need to change the ingestion and storage strategy.

Thank you

Thank you for being so interested in the premium exam material. I'm glad to hear that you found it informative and helpful.

But Wait

I wanted to let you know that there is more content available in the full version. The full paper contains additional sections and information that you may find helpful, and I encourage you to download it to get a more comprehensive and detailed view of all the subject matter.

Total: **529 Questions**