AWS SAP-C02 Exam Questions

Total Questions: 470+
Demo Questions: 30
Version: Updated for 2025

Prepared and Verified by Cert Empire – Your Trusted IT Certification Partner

For Access to the full set of Updated Questions – Visit: SAP-C02 Exam Dumps by Cert Empire

Question: 1

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.

Correct Answer:

C

Explanation:

To implement a regional failover solution, a complete and independent application stack must be deployed in a secondary AWS Region. This includes a new Amazon API Gateway API and new AWS Lambda functions. For the data layer, Amazon DynamoDB global tables provide a fully managed, multi-active, and multi-Region database, ensuring data is available in the failover region. The most direct and precise method to manage the traffic redirection is using an Amazon Route 53 failover routing policy. This policy routes traffic to a primary endpoint (the initial region) and automatically switches to a secondary endpoint (the failover region) if the primary becomes unhealthy, as determined by Route 53 health checks.

Why Incorrect Options are Wrong:

A: An Amazon API Gateway API is a regional resource and cannot have a single endpoint that integrates with Lambda functions in multiple Regions.

B: While multivalue answer routing with health checks can provide failover, it is designed for distributing traffic among multiple healthy resources (active-active), not for a specific active-passive failover scenario.

D: AWS Lambda is a regional service; the concept of "global functions" does not exist, making this premise technically incorrect.

References:

1. Route 53 Failover Routing: AWS Documentation, Amazon Route 53 Developer Guide, "Failover routing policy". This section states, "Use failover routing to configure active-passive failover. You create two records for the same name... Route 53 monitors the health of the primary record... If the primary resource is unhealthy, Route 53 begins responding to DNS queries using the secondary record."

2. DynamoDB Global Tables: AWS Documentation, Amazon DynamoDB Developer Guide, "Global tables". The guide explains, "Global tables build on the global Amazon DynamoDB footprint to provide you with a fully managed, multi-Region, and multi-active database that delivers fast, local, read and write performance for massively scaled, global applications."

3. API Gateway Regional Nature: AWS Documentation, Amazon API Gateway Developer Guide, "Set up a regional API in API Gateway". This documentation confirms that API Gateway APIs are created within a specific AWS Region.

4. Route 53 Multivalue Answer Routing: AWS Documentation, Amazon Route 53 Developer Guide, "Multivalue answer routing policy". This section describes its use case: "If you want to route traffic randomly to multiple resources... use multivalue answer routing." This differentiates it from the specific active-passive failover requirement.

Question: 2

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.

Correct Answer:

C

Explanation:

The solution must address two key requirements: maintaining session consistency for a stateful application and scaling the database to handle increased load.

1. Stateful Application: An Application Load Balancer (ALB) is the appropriate choice as it operates at the application layer (Layer 7) and is designed for HTTP/HTTPS traffic. To ensure a consistent user experience with a stateful application, sticky sessions (session affinity) must be enabled. This directs all requests from a specific user to the same EC2 instance, preserving their session state.

2. Database Scaling: Amazon Aurora handles increased read traffic by adding Aurora Replicas. Aurora Auto Scaling is the feature designed to automatically adjust the number of Aurora Replicas in response to workload changes, thereby scaling the database's read capacity.

Option C correctly combines an Application Load Balancer with sticky sessions for the application tier and Aurora Auto Scaling for Aurora Replicas for the database tier.

Why Incorrect Options are Wrong:

A: A Network Load Balancer (NLB) operates at Layer 4 and is less suitable for a web application requiring application-level features like cookie-based stickiness, which an ALB provides.

B: Aurora Auto Scaling manages the number of Aurora Replicas (for read scaling), not the writer instance. Scaling the writer is a different, typically manual, vertical scaling operation.

D: This option is incorrect on both points: it specifies scaling Aurora writers instead of replicas and incorrectly suggests an NLB for a standard web application.

References:

1. AWS Documentation, Elastic Load Balancing User Guide: "Sticky sessions are a mechanism to route requests from the same client to the same target. Elastic Load Balancing supports sticky sessions for Application Load Balancers." (Reference: Application Load Balancers > Listeners for your Application Load Balancers > Sticky sessions).

2. AWS Documentation, Amazon Aurora User Guide: "With Aurora Auto Scaling, an Aurora DB cluster can automatically adjust the number of Aurora Replicas... This scaling is useful for workloads that have predictable cyclical traffic patterns or unpredictable traffic spikes." (Reference: Amazon Aurora User Guide > Managing an Amazon Aurora DB cluster > Using Amazon Aurora Auto Scaling with Aurora replicas).

3. AWS Documentation, Elastic Load Balancing User Guide: "An Application Load Balancer functions at the application layer, the seventh layer of the Open Systems Interconnection (OSI) model... A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model." (Reference: Elastic Load Balancing User Guide > What is Elastic Load Balancing? > Elastic Load Balancing components). This distinction makes the ALB the standard choice for web applications.

Question: 3

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

Correct Answer:

D

Explanation:

The core requirement is to inspect the User-Agent header of an incoming request and conditionally remove specific headers from the corresponding response before it reaches

the client. This logic needs to be executed at the edge of the network. Amazon CloudFront with Lambda@Edge is the ideal AWS service combination for this use case. A Lambda@Edge function can be configured to trigger on the viewer-response event. In this event, the function has access to both the original request (including the User-Agent header) and the response from the origin. The function can then implement logic to remove the unsupported headers from the response object before it is sent to the end-user's device, perfectly replicating the on-premises functionality in a serverless model.

Why Incorrect Options are Wrong:

A: While a CloudFront Function can manipulate headers, Lambda@Edge provides a more powerful and flexible execution environment, which is better suited for replicating and potentially extending custom logic from an on-premises system.

B: API Gateway "gateway responses" are used to customize error responses (e.g., 4XX, 5XX) generated by API Gateway itself, not for modifying headers on successful backend responses.

C: API Gateway mapping templates are designed for transforming the body of a request or response, not for implementing conditional logic to remove response headers based on an incoming request header.

References:

1. AWS Documentation on Lambda@Edge: "You can use Lambda functions to change CloudFront requests and responses... For example, you can... modify headers or cookies." This directly supports the use case of modifying response headers.

Source: AWS Documentation, What is Lambda@Edge?, Section: "How it works".

2. AWS Documentation on Lambda@Edge Event Triggers: The viewer-response event is specified for "a function to run when the origin returns a response, before CloudFront caches the response." This is the correct trigger to modify the response before it reaches the viewer.

Source: AWS Documentation, Lambda@Edge Developer Guide, Section: "The CloudFront event structure".

3. AWS Documentation on Choosing Edge Functions: "Use Lambda@Edge when you need... more complex or compute-intensive operations...". Replicating logic from a configurable on-premises load balancer fits the profile of a task better suited for the more powerful Lambda@Edge over the more lightweight CloudFront Functions.

Source: AWS Documentation, Amazon CloudFront Developer Guide, Section: "Choosing between CloudFront Functions and Lambda@Edge".

4. AWS Documentation on API Gateway Gateway Responses: "In API Gateway, a gateway response is identified by a response type... For example, if a client submits a request to an undefined API resource path, API Gateway returns a 403 Forbidden response." This confirms gateway responses are for API Gateway-generated errors.

Source: AWS Documentation, API Gateway Developer Guide, Section: "Set up gateway responses".

Question: 4

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 meetthese 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.

Correct Answer:

A

Explanation:

This solution meets all requirements most effectively. AWS Lambda supports container images, providing a fully serverless architecture that abstracts away all infrastructure management, thus minimizing operational complexity. For variable workloads, Lambda's pay-per-invocation pricing model is highly cost-effective as you only pay for the compute time you consume. Using Amazon API Gateway with separate integrations (or stages) is the standard, low-overhead method for managing distinct environments like production and testing. This approach directly addresses the core requirements of being serverless, container-based, low complexity, and cost-effective for variable loads.

Why Incorrect Options are Wrong:

B: While AWS Fargate is a serverless compute engine, managing Amazon ECS clusters, services, task definitions, and Application Load Balancers introduces more operational complexity than the fully managed Lambda and API Gateway solution.

C: Amazon EKS introduces significant operational complexity due to the nature of Kubernetes orchestration, which directly contradicts the requirement to minimize operational complexity.

D: AWS Elastic Beanstalk is a Platform as a Service (PaaS) offering that provisions and manages underlying resources like EC2 instances. It is not considered a serverless compute architecture.

References:

1. AWS Lambda Documentation - Container Images: "With Lambda, you provide your code in one of two ways: as a .zip file archive or a container image. ... When you use a container image to deploy your function, you benefit from the same operational simplicity, automatic scaling, high availability, and native integrations with other AWS services." This confirms Lambda's support for containers and its alignment with minimal operational complexity.

2. AWS Whitepaper - "Running Containers on AWS" (Page 5): This whitepaper compares container services. It positions AWS Lambda as the choice for "serverless functions" and highlights its event-driven nature and automatic scaling, contrasting it with AWS Fargate which is for "serverless containers" but still requires management of tasks and services within a cluster orchestrator (ECS or EKS). This supports the choice of Lambda for minimizing operational overhead.

3. AWS Fargate Documentation - "What is AWS Fargate?": "AWS Fargate is a serverless, pay-as-you-go compute engine... Fargate removes the need to provision and manage servers, lets you specify and pay for resources per application, and improves security through application isolation by design." While serverless, the documentation for setting up services shows it involves more components (task definitions, services, clusters) than a Lambda function.

4. AWS Elastic Beanstalk Documentation - "What is AWS Elastic Beanstalk?": "With Elastic Beanstalk, you can quickly deploy and manage applications in the AWS Cloud without worrying about the infrastructure that runs those applications. AWS Elastic Beanstalk reduces management complexity without restricting choice or control. You simply upload your application, and Elastic Beanstalk automatically handles the details of capacity provisioning, load balancing, scaling, and application health monitoring." This describes a PaaS model, not a serverless compute model like Lambda or Fargate.

Question: 5

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. Whatshould 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.

Correct Answer:

B

Explanation:

This solution describes a classic warm standby disaster recovery (DR) strategy. Option B correctly uses an Amazon Route 53 failover routing policy, which is specifically designed for active-passive configurations. A Route 53 health check continuously monitors the primary application endpoint. Upon failure, Route 53 automatically redirects traffic to the backup region's ALThe health check failure also triggers an Amazon SNS notification, which invokes an AWS Lambda function. This function performs the necessary recovery actions: promoting the RDS read replica to a standalone, writable primary instance and updating the Auto Scaling group's configuration to scale out the EC2 instances, thereby bringing the backup environment fully online. This integrated approach provides the required automated failover within the specified RTO.

Why Incorrect Options are Wrong:

A: Latency-based routing is used for performance optimization by routing users to the lowest-latency region; it is not a failover mechanism and will not redirect traffic away from an unhealthy region.

C: This option creates a hot standby by fully scaling the backup region, which violates the stated budget constraint against an active-active strategy. Additionally, snapshot-based replication has a higher RTO/RPO than promoting a read replica.

D: While AWS Global Accelerator can perform health checks and failover, Route 53 with a failover policy is the more direct, standard, and cost-effective DNS-level solution specifically designed for this active-passive DR scenario.

References:

1. Amazon Route 53 Developer Guide: Describes the use of failover routing policies and health checks for DR. It states, "You can configure active-passive failover, in which one resource...serves traffic when the resource is healthy, and a secondary resource serves traffic when the primary resource is unhealthy."

Source: Amazon Route 53 Developer Guide, "Configuring DNS failover".

2. AWS Whitepaper: Disaster Recovery Options in the Cloud: This paper outlines the "Warm Standby" approach, where a scaled-down version of the infrastructure runs in the DR region and is scaled up upon failover. The scenario in the question perfectly matches this pattern.

Source: AWS Whitepaper, "Disaster Recovery Options in the Cloud", Page 10, "Warm Standby" section.

3. Amazon RDS User Guide: The process of promoting a read replica is the standard procedure for database failover in a cross-region DR scenario.

Source: Amazon RDS User Guide, "Promoting a read replica to be a standalone DB instance".

4. Amazon EC2 Auto Scaling User Guide: A Lambda function can programmatically update the minimum, maximum, and desired capacity of an Auto Scaling group to scale out the application tier during a failover event.

Source: Amazon EC2 Auto Scaling User Guide, "Changing the desired capacity of an Auto Scaling group".

Question: 6

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 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 inbound resolver.

B: Associate the private hosted zone to all the VPCs. Deploy an Amazon EC2 conditional forwarder 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 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 VPCreate a Route 53 inbound resolver in the shared services VPAttach 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.

Correct Answer:

A

Explanation:

This architecture correctly addresses both DNS resolution requirements using native, highly available AWS services. A Route 53 Resolver inbound endpoint is the purpose-built,

managed service for receiving DNS queries from an on-premises network, offering the highest performance and scalability. Associating the private hosted zone with all VPCs ensures that resources within any VPC can resolve the cloud.example.com domain names directly using the VPC's default DNS resolver. The Transit Gateway provides the necessary network path from the on-premises network to the inbound resolver endpoint located in the shared services VPC.

Why Incorrect Options are Wrong:

B: Using a self-managed Amazon EC2 conditional forwarder introduces operational overhead and is less performant and resilient compared to the managed, auto-scaling Route 53 Resolver service.

C: A Route 53 outbound resolver is used for forwarding queries from a VPC to an on-premises network, which is the reverse of the required traffic flow for this scenario.

D: Associating the private hosted zone with only the shared services VPC prevents other VPCs from resolving the domain names, failing to meet a key requirement of the solution.

References:

1. AWS Documentation - Route 53 Developer Guide: "To resolve DNS queries for a private hosted zone from your on-premises network, you can create a Route 53 Resolver inbound endpoint. The inbound endpoint is a collection of elastic network interfaces in one or more Availability Zones in a VPYour on-premises DNS resolvers can forward DNS queries to the IP addresses of the network interfaces." (Reference: Resolving DNS queries between VPCs and your network, Section: Inbound endpoints).

2. AWS Documentation - Route 53 Developer Guide: "To allow Amazon EC2 instances in a VPC to resolve the domain names for your resources (for example, determining the IP address for a web server from the domain name www.example.com), you must associate the VPC with the private hosted zone." (Reference: Working with private hosted zones, Section: Associating a VPC with a private hosted zone).

3. AWS Whitepaper - Hybrid Cloud DNS Solutions for Amazon VPC: This paper details architectures for hybrid DNS. It describes the use of Route 53 Resolver inbound endpoints as the recommended pattern for on-premises networks to resolve domains in Route 53 private hosted zones. (Reference: Architecture 2: On-Premises DNS Resolution for a Private Hosted Zone in Amazon VPC).

Question: 7

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.

Correct Answer:

D

Explanation:

The most effective and secure solution is to restructure the application of the Service Control Policy (SCP). The core issue is the restrictive deny SCP at the organization's root, which is inherited by all accounts. By creating a temporary "Onboarding" OU and moving the restrictive SCP from the root to the "Production" OU, the new account can be placed in the Onboarding OU without being subject to the restriction. This allows its administrators to perform the necessary AWS Config updates. Once complete, moving the account to the "Production" OU ensures it inherits the required security policies. This approach maintains security for existing accounts, isolates the onboarding process, and improves the long-term SCP strategy by applying policies at a more granular level.

Why Incorrect Options are Wrong:

A: Removing the root SCP entirely eliminates a critical preventive control for all accounts in the organization, which is a significant security regression and fails to enforce current policies.

B: This is incorrect because an SCP that allows an action in a child OU cannot override an explicit deny policy inherited from a parent OU or the root.

C: Converting the entire SCP strategy from a deny list to an allow list is a massive operational change that is overly complex and disruptive for solving this temporary onboarding requirement.

References:

1. AWS Organizations User Guide, "SCP evaluation": This guide explains that an explicit deny in an SCP at any level in the hierarchy (including the root) overrides any allow at a lower level. This directly invalidates option The documentation states, "To determine whether a permission is allowed or denied for a specific account, AWS Organizations aggregates all policies that affect that account. If at any level a Deny statement applies to a specific action, then any Allow statements at a lower level for that same action have no effect."

2. AWS Organizations User Guide, "Strategies for using SCPs": This documentation recommends attaching SCPs as close to the resources they are meant to manage as possible, rather than applying broad policies at the root. Moving the SCP from the root to the Production OU, as suggested in option D, aligns with this best practice for creating a more flexible and scalable organization structure.

3. AWS Whitepaper, "Organizing Your AWS Environment Using Multiple Accounts" (Page 15-16): This whitepaper discusses structuring OUs based on function and policy requirements. Creating separate OUs for different stages (like onboarding) or with different security controls is a recommended pattern. Option D's use of a temporary Onboarding OU aligns with this architectural guidance.

Question: 8

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.

Correct Answer:

A, D, F

Explanation:

The goal is to eliminate single points of failure and enable automatic recovery for a three-tier application.

1. (A) EC2 Compute Layer: Placing EC2 instances in an Auto Scaling group with a minimum of two instances, distributed across Availability Zones and fronted by an Elastic Load Balancer, ensures high availability. If one instance fails, the Auto Scaling group automatically launches a replacement, and the load balancer directs traffic to the healthy instance(s), minimizing downtime.

2. (D) RDS Database Layer: Converting the RDS instance to a Multi-AZ deployment is the standard method for achieving high availability. AWS automatically creates and maintains a synchronous standby replica in a different Availability Zone. In the event of a primary instance failure, RDS automatically fails over to the standby replica.

3. (F) ElastiCache In-Memory Layer: For ElastiCache for Redis, creating a replication group with Multi-AZ enabled provides automatic failover. If the primary node fails, ElastiCache automatically promotes an available replica in another AZ to become the new primary.

Why Incorrect Options are Wrong:

B: T-series "unlimited mode" is a CPU performance feature for burstable instances; it does not provide high availability or automatic recovery from instance failure.

C: An RDS read replica is primarily for read scaling. Promoting it to a primary instance is a manual process, which does not meet the "automatically recover" requirement.

E: ElastiCache for Redis does not use EC2-style Auto Scaling groups for high availability failover. High availability is achieved through replication groups with Multi-AZ enabled.

References:

1. Auto Scaling Groups (for A): AWS Auto Scaling User Guide. "Auto Scaling groups let you use Amazon EC2 Auto Scaling features such as health check replacements and scaling policies. ... If you have an instance that is running and you want to ensure that there is always an instance of that same configuration running, you can create an Auto Scaling group that has a desired, minimum, and maximum capacity of 1." To achieve high availability, this minimum is set to 2 or more. (Source: AWS Auto Scaling User Guide, "What is Amazon EC2 Auto Scaling?")

2. RDS Multi-AZ (for D): Amazon RDS User Guide. "In a Multi-AZ DB cluster deployment, Amazon RDS automatically provisions and maintains a synchronous standby replica in a different Availability Zone. ... If there is a planned or unplanned outage of your primary DB instance, Amazon RDS automatically switches to a standby replica in another Availability Zone." (Source: Amazon RDS User Guide, "Multi-AZ DB cluster deployments")

3. ElastiCache Multi-AZ (for F): Amazon ElastiCache for Redis User Guide. "To improve fault tolerance, you can enable Multi-AZ on your Redis replication group. ... If the primary node fails, ElastiCache automatically detects the failure, selects a replica, and promotes it to be the new primary." (Source: Amazon ElastiCache for Redis User Guide, "Minimizing downtime in ElastiCache for Redis with Multi-AZ")

4. RDS Read Replicas (for C): Amazon RDS User Guide. "You can promote a read replica to be a standalone DB instance. When you promote a read replica, the DB instance is rebooted before it becomes available." This is a manual action, not an automatic failover.

(Source: Amazon RDS User Guide, "Promoting a read replica to be a standalone DB instance")

Question: 9

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 ALStatic 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 ALThe 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.

Correct Answer:

A, E

Explanation:

The most efficient and direct solution with the least operational overhead is to leverage the native features of Amazon CloudFront. The first step is to create the custom error page and host it in a location accessible by CloudFront, such as an Amazon S3 bucket configured for static website hosting (Option A).

The second step is to configure the CloudFront distribution to handle specific HTTP error codes from the origin. By creating a custom error response rule for the 502 status code, CloudFront can intercept this error from the Application Load Balancer (ALB) and serve the pre-configured custom error page from the S3 bucket instead (Option E). This approach is a simple, declarative configuration that requires no ongoing management, unlike reactive solutions involving AWS Lambda or DNS changes.

Why Incorrect Options are Wrong:

B: This is incorrect because Target.FailedHealthChecks indicates an unhealthy instance, not an intermittent application error. Also, using a Lambda function to modify ALB rules is a complex, high-overhead solution.

C: DNS failover with Route 53 is not suitable for intermittent, request-level application errors. It would cause a full site failover, impacting all users, rather than just serving a custom page for the error.

D: The Elb.InternalError metric tracks errors within the load balancer service itself, not 502 errors caused by the backend target. This solution is also overly complex and has high operational overhead.

References:

1. Amazon CloudFront Developer Guide - Serving custom error pages: "You can have CloudFront return a custom error page to the viewer... when your origin server returns an HTTP 4xx or 5xx status code to CloudFront. ... Instead of the origin's error page, you can provide a custom error page from another path in the same origin or from a different location, such as an Amazon S3 bucket." This directly supports the combination of options A and (Source: AWS Documentation, Amazon CloudFront Developer Guide, "Configuring custom error responses").

2. Amazon S3 User Guide - Hosting a static website using Amazon S3: "You can host a static website on Amazon S3. On a static website, individual webpages include static content." This supports Option A as the method for hosting the custom error page. (Source: AWS Documentation, Amazon S3 User Guide, "Hosting a static website using Amazon S3").

3. Amazon Elastic Load Balancing User Guide - CloudWatch metrics for your Application Load Balancer: This guide details the available metrics. The HTTPCodeTarget5XXCount metric counts 5xx errors from targets, while InternalError counts errors generated by the

load balancer itself. This shows why the metric in option D is incorrect. (Source: AWS Documentation, Elastic Load Balancing User Guide, "Application Load Balancer metrics").

Question: 10

A company is using multiple AWS accounts. The DNS records are stored in a private hosted zone for Amazon Route 53 in Account The company's applications and databases are running in Account B.

A solutions architect will deploy a two-tier application in a new VPTo 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 VPCreate 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 Configure Route 53 replication between AWS accounts.

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

Correct Answer:

C, E

Explanation:

To resolve DNS queries for a private hosted zone in one AWS account (Account A) from a VPC in another account (Account B), a two-step, cross-account association process is required. First, the account that owns the private hosted zone (Account A) must create an authorization, explicitly granting the VPC in Account B permission to associate with it. Second, the account that owns the VPC (Account B) must accept this authorization by creating the association between its VPC and the hosted zone in Account This establishes the necessary linkage for the VPC's DNS resolver to query the private hosted zone.

Why Incorrect Options are Wrong:

A: This is an architectural workaround that does not solve the fundamental cross-account DNS resolution issue. The new EC2 instance would still be unable to resolve records without the proper association.

B: Manually editing /etc/resolv.conf is not a scalable or persistent solution. This file is managed by the VPC's DHCP options set and any manual changes are typically overwritten.

D: This creates a separate, unmanaged DNS zone. Amazon Route 53 does not have a native feature to "replicate" records between private hosted zones in this manner, leading to management overhead.

References:

1. AWS Documentation - Amazon Route 53 Developer Guide: In the section "Working with private hosted zones," the process for cross-account association is detailed. It states: "To associate a VPC that you own with a private hosted zone that was created by using a different AWS account, you perform two steps:

The account that created the hosted zone submits a request to authorize the association of your VPC with their private hosted zone. (This corresponds to option C).

Using the account that created the VPC, you accept the request to associate the VPC with the private hosted zone. (This corresponds to option E)."

Source: AWS Documentation, Amazon Route 53 Developer Guide, "Associating a VPC and a private hosted zone that you created with different AWS accounts".

2. AWS Documentation - CreateVPCAssociationAuthorization API: The documentation for this API call confirms the first step: "Authorizes the AWS account that created a specified VPC to submit an AssociateVPCWithHostedZone request to associate the VPC with a specified hosted zone that was created by a different account." This directly supports option C.

Source: AWS CLI Command Reference, route53 create-vpc-association-authorization.

3. AWS Documentation - AssociateVPCWithHostedZone API: The documentation for this API call confirms the second step: "Associates an Amazon VPC with a public or private hosted zone... For a private hosted zone, you can associate any VPC with the hosted zone." This action is performed by the VPC owner after authorization, supporting option E.

Source: AWS CLI Command Reference, route53 associate-vpc-with-hosted-zone.

Question: 11

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.

Correct Answer:

C

Explanation:

This solution provides the most comprehensive re-architecture to meet all stated requirements by leveraging a serverless and event-driven pattern. Hosting the static web content in Amazon S3 and using it to store the uploaded videos is a highly scalable, durable, and cost-effective approach that significantly reduces operational overhead compared to managing EC2 instances and EBS volumes. The workflow of using S3 event notifications to trigger an SQS message, which is then processed by an AWS Lambda function, creates a decoupled, resilient, and fully managed processing pipeline. This eliminates the need for an Auto Scaling group of EC2 instances for the worker tier. Finally, replacing the custom software with the Amazon Rekognition API fulfills the requirement to use a managed AWS service for video analysis.

Why Incorrect Options are Wrong:

A: This option still relies on EC2-based compute (ECS and Spot instances), which retains significant operational overhead for managing clusters and instances compared to a serverless Lambda approach.

B: Using Amazon EFS is suboptimal for storing user-uploaded video objects; Amazon S3 is the more appropriate service. This option also retains EC2 instances for the web application.

D: AWS Elastic Beanstalk simplifies EC2 deployment but does not eliminate the operational overhead of managing the underlying compute environments, making it less efficient than the serverless model in option C.

References:

1. Amazon S3 for Storage and Static Hosting: The AWS Documentation states, "Amazon S3 is an object storage service... Use it to store and protect any amount of data for a range of use cases, such as websites, mobile applications...". It also details how to "use Amazon S3 to host a static website."

Source: AWS Documentation, "What is Amazon S3?", and "Hosting a static website using Amazon S3".

2. Event-Driven Architecture (S3 -> SQS -> Lambda): This is a common serverless pattern. The S3 documentation describes how to publish event notifications to SQS. The Lambda documentation explains how to use an SQS queue as an event source.

Source: AWS Documentation, "Configuring Amazon S3 event notifications", Section: "Walkthrough: Configuring a bucket for notifications (SNS topic or SQS queue)".

Source: AWS Documentation, "Using AWS Lambda with Amazon SQS".

3. Amazon Rekognition for Video Analysis: The service is designed to replace custom recognition software with a managed API. "Amazon Rekognition makes it easy to add image and video analysis to your applications using proven, highly scalable, deep learning technology that requires no machine learning expertise to use."

Source: AWS Documentation, "What Is Amazon Rekognition?".

4. AWS Lambda for Serverless Compute: Lambda is ideal for reducing operational overhead. "AWS Lambda is a serverless, event-driven compute service that lets you run code for virtually any type of application or backend service without provisioning or managing servers."

Source: AWS Documentation, "What is AWS Lambda?".

Question: 12

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 VPThe 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.

Correct Answer:

A

Explanation:

The primary requirement is the lowest cost solution for storing and privately serving archived documents where availability and retrieval speed are not concerns. An Amazon S3-based solution is inherently more cost-effective for low-traffic scenarios than an EC2-based one, as it avoids the continuous cost of a running virtual server. The S3 One Zone-Infrequent Access (S3 One Zone-IA) storage class is specifically designed for this use case: it provides a lower storage cost than other instantly accessible tiers by storing data in a single Availability Zone. This is acceptable because the requirements state availability is not a concern and the documents are copies. An S3 interface endpoint provides secure, private access from the VPC, fulfilling the requirement that data not be publicly accessible.

Why Incorrect Options are Wrong:

B: This option is more expensive because it requires a constantly running Amazon EC2 instance and an Amazon EFS file system, which is not the most cost-effective for this use case.

C: Similar to option B, this solution incurs the higher cost of a continuously running EC2 instance and is therefore not the lowest-cost option for a low-request workload.

D: S3 Glacier Deep Archive is designed for archiving with retrieval times of 12 hours or more. It is incompatible with S3 static website hosting and cannot make documents "available" for immediate access on an intranet.

References:

1. Amazon S3 Storage Classes: The official AWS documentation details the use cases for different storage classes. S3 One Zone-IA is described as "for data that is accessed less frequently, but requires rapid access when needed. Unlike other S3 Storage Classes which store data in a minimum of three Availability Zones (AZs), S3 One Zone-IA stores data in a single AZ and costs 20% less than S3 Standard-IA." This aligns with the low-cost and non-critical availability requirements.

Source: AWS Documentation, Amazon S3 Storage Classes, "S3 One Zone-IA" section.

2. Amazon S3 Glacier Retrieval: The documentation for S3 Glacier storage classes states, "To access an object in S3 Glacier Instant Retrieval, S3 Glacier Flexible Retrieval, or S3 Glacier Deep Archive, you must first initiate a restore request." For S3 Glacier Deep Archive, standard retrieval takes up to 12 hours. This makes it unsuitable for direct access via an intranet.

Source: AWS Documentation, Amazon S3 User Guide, "Restoring an archived object" section.

3. Interface VPC endpoints for Amazon S3: "By using an interface VPC endpoint... you can provide your private applications in VPCs access to S3... without an internet gateway, NAT device, or virtual private gateway." This supports the private access requirement.

Source: AWS Documentation, Amazon S3 User Guide, "Accessing S3 using an interface endpoint" section.

4. Comparing S3 and EC2+EBS/EFS for Web Hosting: For static content with low traffic, S3 is the most cost-effective solution. "For a simple web application, you can reduce your costs by using serverless services. For example, you can host a static website on Amazon S3 instead of using a web server on an Amazon EC2 instance."

Source: AWS Well-Architected Framework, Cost Optimization Pillar, "Expenditure and usage awareness" section (COST 1).

Question: 13

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.

Correct Answer:

A

Explanation:

To accurately track costs using custom tags across an AWS Organization, the user-defined tag key must first be activated in the Billing and Cost Management console of the management account. This action makes the tag available for cost allocation purposes. Subsequently, configuring an AWS Cost and Usage Report (CUR) is the most effective method for obtaining granular, resource-level cost data. The CUR can be configured to include user-defined tags, allowing for precise filtering and calculation of costs associated

with specific resources, such as those tagged with costCenter:compliance. This provides the most accurate data required by the compliance team.

Why Incorrect Options are Wrong:

B: This is incorrect because user-defined cost allocation tags are activated centrally from the organization's management account, not from individual member accounts.

C: This is incorrect for the same reason as The activation of cost allocation tags is a centralized function performed within the management account.

D: AWS Trusted Advisor is a service that provides recommendations for cost optimization, security, and performance. It does not generate detailed, tag-based billing reports like the AWS Cost and Usage Report.

References:

1. AWS Billing User Guide, "Activating user-defined cost allocation tags": "To use your own tags in a cost report, you must first activate them... If you're using AWS Organizations, sign in using your management account credentials to view the cost allocation tags for member accounts. You can activate tags for each member account from the management account." This confirms that activation is performed in the management account.

2. AWS Cost and Usage Reports User Guide, "Cost and Usage Report data dictionary": The documentation details the columns available in a CUR, including resourcetagsuser prefixed columns, which contain the values for activated user-defined cost allocation tags. This confirms CUR is the correct tool for this granular analysis.

3. AWS Cost and Usage Reports User Guide, "Creating Cost and Usage Reports": This section outlines the process of setting up a CUR and specifies the option to "Include resource IDs" and by extension, resource tags, which is necessary for the required level of detail.

Question: 14

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 SLThe 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.

Correct Answer:

D

Explanation:

This solution provides the highest availability by distributing instances across three Availability Zones (AZs), a best practice for resilience. It correctly aligns the EC2 purchasing options with the workload requirements for maximum cost-effectiveness. The critical, SLA-bound scheduled jobs (65% of usage) are run on On-Demand Instances with Capacity Reservations, which guarantees capacity in each AZ without a long-term commitment. The non-critical user jobs (35% of usage) are run on highly cost-effective Spot Instances. This is appropriate as these jobs have no SLA and can tolerate interruptions, which is the defining

characteristic of Spot Instances. This configuration meets all technical and business requirements.

Why Incorrect Options are Wrong:

A: The allocation of 33% reserved capacity is insufficient for the 65% critical workload. Using only two AZs is also less resilient than using three.

B: Placing all reserved capacity in a single Availability Zone creates a single point of failure for the critical, SLA-bound workload, violating the high availability requirement.

C: Savings Plans require a 1- or 3-year term, which directly violates the specific requirement for "no long-term commitments."

References:

1. On-Demand Capacity Reservations: AWS EC2 User Guide for Linux Instances. In the section "On-Demand Capacity Reservations," it states, "Capacity Reservations enable you to reserve capacity for your Amazon EC2 instances in a specific Availability Zone for any duration... You can cancel a Capacity Reservation at any time." This confirms they guarantee capacity without a long-term commitment.

2. Spot Instances: AWS EC2 User Guide for Linux Instances. The "Spot Instances" section explains, "A Spot Instance is an unused EC2 instance that is available for less than the On-Demand price... If Amazon EC2 needs the capacity back, Spot Instances can be interrupted." This makes them ideal for fault-tolerant, non-critical workloads like the user jobs described.

3. High Availability and Availability Zones: AWS Well-Architected Framework, Reliability Pillar (July 2023). Under "REL 04: How do you design your workload architecture to withstand component failures?", the framework recommends deploying workloads across multiple Availability Zones to achieve high availability and fault tolerance. Distributing capacity across three AZs provides a higher level of resilience than two.

4. Savings Plans: AWS Savings Plans User Guide. The "What are Savings Plans?" section states they offer lower prices "in exchange for a commitment to a consistent amount of usage (measured in $/hour) for a 1 or 3 year term." This confirms they are a long-term commitment, making option C incorrect.

Question: 15

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.

Correct Answer:

B

Explanation:

The root cause of the errors is a single client overwhelming the backend services with a high volume of PUT requests. The most direct and effective architectural solution is to control the rate of requests at the entry point. Amazon API Gateway usage plans allow for the configuration of throttling (rate and burst limits) on a per-API-key basis. By associating the problematic client's API key with a usage plan that has appropriate limits, the solutions architect can prevent this single client from degrading the service for all users. The client application should be designed to handle the resulting 429 Too Many Requests HTTP status code gracefully, which aligns with the scenario's tolerance for retries.

Why Incorrect Options are Wrong:

A: This is a client-side mitigation strategy. While good practice, it does not solve the core server-side problem of being overwhelmed by excessive requests from a single source.

C: API caching is primarily effective for GET requests to serve repeatable reads. It is not an appropriate solution for PUT requests, which are intended to modify data.

D: Reserved concurrency for a Lambda function sets a maximum concurrency limit. This could exacerbate the problem by causing more throttling errors if the single client's traffic spike exceeds the reserved amount.

References:

1. AWS API Gateway Developer Guide, "Setting up usage plans with API keys": "Usage plans help you declare a plan for selected API clients that specifies throttling limits... You associate the plan with API keys that you distribute to the clients." This document directly supports using usage plans and API keys to enforce throttling on specific clients.

2. AWS Well-Architected Framework, "Reliability Pillar" whitepaper (Page 26): Under "REL 11: How do you design your workload to withstand component failures?", the framework advises to "Throttle requests: This is to prevent a workload or its dependency from being overwhelmed by traffic." This principle supports implementing throttling (Option B) as a reliability mechanism.

3. AWS Lambda Developer Guide, "Configuring reserved concurrency": "Reserving concurrency has two effects. First, it sets the maximum concurrency for the function... Second, it guarantees that the specified number of instances are available for your function at all times." This shows that reserved concurrency creates a hard cap, which is not ideal for handling an uncontrolled traffic source and could lead to more throttling, making Option D incorrect.

4. AWS API Gateway Developer Guide, "Enabling API caching to enhance responsiveness": "You can enable API caching in Amazon API Gateway to cache your endpoint's responses... When you enable caching for a stage, API Gateway caches responses from your endpoint for a specified time-to-live (TTL) period, in seconds." The documentation focuses on caching responses, which is applicable to idempotent methods like GET, not state-changing methods like PUT.

Question: 16

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 VPThe 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.

Correct Answer:

B, D

Explanation:

The scenario describes the need for a centralized VPC management model where one account (the infrastructure account) owns the VPC, and other accounts can launch resources into its subnets. This is the exact use case for VPC Sharing, which is facilitated by AWS Resource Access Manager (RAM).

To implement VPC Sharing across an organization, two key steps are required. First, resource sharing must be enabled from the AWS Organizations management account, which is a prerequisite for sharing any resources via RAM within the organization. Second, the VPC owner (the infrastructure account) must create a resource share in RAM, specify

the subnets to be shared, and grant access to the consumer accounts, which can be done by targeting an entire Organizational Unit (OU).

Why Incorrect Options are Wrong:

A: A transit gateway is used to connect multiple VPCs and on-premises networks. It does not enable launching resources from one account into another account's subnets.

C: This describes VPC peering and requires creating a VPC in each account, which contradicts the requirement for centralized network management and that individual accounts cannot manage their own networks.

E: While prefix lists can be shared using RAM, sharing them does not allow other accounts to launch resources into the VPSharing subnets is the correct action for this purpose.

References:

1. Amazon VPC User Guide, "Share your VPC with other accounts": This guide explicitly states the process: "The owner of the VPC shares one or more subnets with other accounts (participants) that belong to the same organization in AWS Organizations. The participants can then view, create, modify, and delete their application resources in the subnets that are shared with them." This supports option D.

2. AWS Resource Access Manager User Guide, "Enable sharing with AWS Organizations": This document outlines the prerequisite for sharing resources within an organization: "To begin sharing resources that you own with accounts in your organization in AWS Organizations, you must first enable sharing with AWS Organizations from the AWS RAM console." This supports option B.

3. AWS Resource Access Manager User Guide, "Sharing your AWS resources": Under the "Shareable AWS resources" section, it lists "Subnet (for VPC sharing)" as a shareable resource type. It also details the process: "The resource owner creates a resource share to share resources with resource consumers." This further validates option D as the correct implementation step.

4. AWS Transit Gateway Documentation, "How transit gateways work": This documentation describes a transit gateway as a "network transit hub" used to "interconnect your virtual private clouds (VPCs) and on-premises networks." This confirms that its purpose is network connectivity, not sharing subnets for resource launching, making option A incorrect.

Question: 17

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.

Correct Answer:

A

Explanation:

AWS Systems Manager is designed to provide a unified operational view and control for hybrid cloud environments. By installing the SSM Agent on both on-premises servers and Amazon EC2 instances, they become managed nodes within the Systems Manager console. The Patch Manager capability within Systems Manager can then be used to automate the process of patching these managed nodes. Crucially, Patch Manager also provides centralized patch compliance reporting, allowing the company to generate a single, consolidated report showing the patch status of all servers, regardless of their location, thereby fulfilling all stated requirements.

Why Incorrect Options are Wrong:

B: While AWS OpsWorks can manage configuration, including patches, AWS Systems Manager is the purpose-built service for patch management and compliance reporting in a hybrid environment.

C: Amazon Inspector is a vulnerability management service that scans for software vulnerabilities, not a tool for generating patch compliance reports against a defined baseline.

D: AWS X-Ray is a service for tracing and debugging distributed applications. It is not used for infrastructure patching or compliance reporting.

References:

1. AWS Systems Manager User Guide, "Setting up Systems Manager for hybrid and multicloud environments": This section details how to register on-premises servers and virtual machines with Systems Manager, making them managed nodes alongside EC2 instances. This establishes the basis for unified management.

2. AWS Systems Manager User Guide, "AWS Systems Manager Patch Manager": "Patch Manager, a capability of AWS Systems Manager, automates the process of patching managed nodes with both security-related and other types of updates... You can use Patch Manager to apply patches for both operating systems and applications." This confirms its function for patching.

3. AWS Systems Manager User Guide, "About patch compliance": "After Patch Manager scans your nodes for compliance, you can view the compliance data in the Systems Manager console or retrieve the data using the AWS CLI or Systems Manager APIs... Patch compliance information is provided for each individual node." This confirms its integrated reporting capability for all managed nodes.

4. AWS Documentation, "What is Amazon Inspector?": "Amazon Inspector is a vulnerability management service that continuously scans your AWS workloads for software vulnerabilities and unintended network exposure." This differentiates its purpose from patch compliance reporting.

5. AWS Documentation, "What Is AWS OpsWorks?": "AWS OpsWorks is a configuration management service that provides managed instances of Chef and Puppet." This defines its primary role, which is broader and less specific to patching and compliance than Systems Manager.

Question: 18

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.

Correct Answer:

B

Explanation:

Using AWS Serverless Application Model (SAM) with its built-in AWS CodeDeploy integration directly addresses the requirements. This combination enables safe, gradual deployments (canary or linear) for Lambda functions. The DeploymentPreference property in an AWS SAM template automates the process of shifting a configurable percentage of traffic to the new function version. Pre-traffic and post-traffic Lambda functions can be used to run integration tests. If Amazon CloudWatch alarms are triggered during the deployment, CodeDeploy automatically and immediately rolls back the deployment by shifting traffic back

to the stable, previous version. This significantly reduces the time to detect errors and revert, while streamlining the deployment process.

Why Incorrect Options are Wrong:

A: While CloudFormation is used, a standard stack update is an all-or-nothing deployment. It does not provide the gradual traffic shifting needed for rapid error detection and low-impact rollbacks.

C: Refactoring CLI scripts is a minor operational improvement but does not change the underlying all-or-nothing deployment model or introduce the automated, gradual rollout and rollback capabilities required.

D: This blue/green strategy is overly complex and slow. Changing a CloudFront origin to a new API Gateway endpoint can take a significant amount of time to propagate globally, failing the goal of fast deployment and reversion.

References:

1. AWS SAM Developer Guide - Deploying serverless applications gradually: "You can use AWS SAM to perform gradual deployments for your Lambda functions... AWS SAM uses AWS CodeDeploy to shift traffic between two function versions based on the deployment preference type that you specify... If any of the alarms are breached, a rollback of the deployment is automatically triggered." This document details the use of DeploymentPreference, Alarms, and Hooks (PreTraffic and PostTraffic functions).

2. AWS CloudFormation User Guide - AWS::Serverless::Function resource type reference: The documentation for the AWS::Serverless::Function resource describes the DeploymentPreference property. It states, "Use this property to enable gradual deployments for a function's alias. AWS SAM uses AWS CodeDeploy to roll out new function versions in stages." This confirms that SAM natively integrates with CodeDeploy for this specific purpose.

3. AWS CodeDeploy User Guide - Introduction to deployments on an AWS Lambda compute platform: "AWS CodeDeploy can manage the deployment of a new version of your Lambda function. You can choose a deployment configuration that specifies the way you want to shift traffic to the new version... If there are issues with a deployment, CodeDeploy can roll back the changes automatically." This source explains the underlying mechanism that AWS SAM utilizes.

Question: 19

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.

Correct Answer:

B

Explanation:

This solution provides a reliable and event-driven mechanism to perform actions before an EC2 instance is terminated by an Auto Scaling group. An Auto Scaling lifecycle hook for the autoscaling:EC2INSTANCETERMINATING state transition pauses the termination process. This event is captured by an Amazon EventBridge rule, which invokes an AWS Lambda function. The Lambda function uses the AWS Systems Manager SendCommand operation to execute a predefined script (as an SM Document) on the instance to copy the log files. Once the script completes, the Lambda function signals the lifecycle hook to CONTINUE, allowing the Auto Scaling group to proceed with the instance termination. This ensures the log copy operation is completed successfully before the instance is lost.

Why Incorrect Options are Wrong:

A: Using ABANDON is incorrect. It instructs the Auto Scaling group to discard the lifecycle action and not proceed with termination, which is contrary to the goal of scaling in.

C: User data scripts run only once at instance launch, not on termination. Reducing the log copy interval to 5 minutes mitigates but does not eliminate the risk of data loss.

D: Using ABANDON is incorrect as it would prevent the Auto Scaling group from completing the termination process. The correct action to resume termination is CONTINUE.

References:

1. AWS Auto Scaling User Guide: Describes the use of lifecycle hooks to perform custom actions as an Auto Scaling group launches or terminates instances. The EC2INSTANCETERMINATING state is specifically designed for this scenario. The guide also provides tutorials that use EventBridge, Lambda, and Systems Manager.

Source: AWS Auto Scaling User Guide, "Amazon EC2 Auto Scaling lifecycle hooks", Section: "How lifecycle hooks work".

2. AWS Systems Manager User Guide: Explains how to use Run Command with a Systems Manager (SSM) Document to execute scripts on managed instances. This is the recommended method for running commands on EC2 instances programmatically.

Source: AWS Systems Manager User Guide, "AWS Systems Manager Run Command".

3. AWS Auto Scaling API Reference: The documentation for the CompleteLifecycleAction API call explicitly defines the CONTINUE and ABANDON results. CONTINUE signals the action was successful and the instance can proceed to the next state (termination).

Source: AWS Auto Scaling API Reference, "CompleteLifecycleAction", Parameter: LifecycleActionResult.

4. Amazon EventBridge User Guide: Details how to create rules that match events, such as those from Auto Scaling lifecycle actions, and route them to targets like AWS Lambda for processing.

Source: Amazon EventBridge User Guide, "Creating a rule for an AWS service".

Question: 20

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.

Correct Answer:

A

Explanation:

This solution correctly leverages AWS IAM Identity Center (formerly AWS Single Sign-On), which is the recommended service for centrally managing access across multiple AWS accounts within AWS Organizations. Connecting to the on-premises Active Directory via a SAML 2.0 identity provider meets the requirement to use the existing authentication service. Using the System for Cross-domain Identity Management (SCIM) protocol automates the

provisioning of users and groups from Active Directory into IAM Identity Center, ensuring identities are managed in a single location. Finally, attribute-based access control (ABAC) allows for the creation of fine-grained, conditional access rules based on user attributes (such as group membership) passed from Active Directory, fulfilling the security policy requirement.

Why Incorrect Options are Wrong:

B: This option is incorrect because it proposes using IAM Identity Center as the identity source, which contradicts the requirement to use the company's existing on-premises Active Directory.

C: This option is incorrect because it suggests provisioning persistent IAM users for federated identities, which is against federation best practices and creates unnecessary management overhead. Federation should use temporary credentials via IAM roles.

D: While technically feasible, this describes a legacy and more complex method of managing federation. It lacks the centralized management and deep integration with AWS Organizations that IAM Identity Center provides.

References:

1. AWS IAM Identity Center User Guide, "Choose your identity source": This section details the options for an identity source, including connecting to an external identity provider. It states, "You can use an external IdP that supports Security Assertion Markup Language (SAML) 2.0 as your identity source." It also describes using SCIM for automatic provisioning.

2. AWS IAM Identity Center User Guide, "Attribute for access control": This page explains how to implement ABAC within IAM Identity Center. It states, "You can configure attributes in IAM Identity Center for ABAC... When your users federate into the AWS account, their attributes are passed to IAM and can be used for access control decisions."

3. AWS Security Blog, "How to implement SAML 2.0-based federation with AWS": This blog post contrasts the legacy IAM-based federation setup with the modern, recommended approach using AWS IAM Identity Center, highlighting the latter's advantages for multi-account governance.

4. AWS IAM User Guide, "About SAML 2.0-based federation": This documentation describes the underlying mechanism of SAML federation, which involves users assuming IAM roles to obtain temporary security credentials, not creating persistent IAM users as suggested in option C.

Question: 21

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.

Correct Answer:

A

Explanation:

This solution provides the largest cost reduction by addressing both storage and compute costs. Storing the 200 TB of data in Amazon S3 is significantly cheaper than on a self-managed file system running on EC2 instances. S3 Intelligent-Tiering further optimizes storage costs by automatically moving data to lower-cost tiers when it's not accessed. For the compute part, creating a high-performance Amazon FSx for Lustre file system only for the 72-hour job duration is highly cost-effective. The key is using lazy loading, which only loads data from S3 as it is requested by the application. Since the job reads only a subset of the data, this minimizes the provisioned FSx capacity, startup time, and cost compared to loading the entire dataset.

Why Incorrect Options are Wrong:

B: Amazon EBS Multi-Attach is limited to a small number of instances within a single Availability Zone and is not designed for high-performance access from hundreds of instances.

C: This option is less cost-effective than Batch loading would pre-load the entire 200 TB dataset, increasing startup time and requiring a larger, more expensive FSx file system.

D: AWS Storage Gateway is designed for hybrid cloud scenarios to provide on-premises applications with access to S3, not as a high-performance, massively parallel file system for cloud-native compute clusters.

References:

1. Amazon FSx for Lustre Documentation - Using data repositories with Amazon FSx for Lustre: "You can link your file system to an Amazon S3 bucket... When you link to an S3 bucket, you can choose to load data into your file system from S3 as it is accessed by your application (lazy loading)." This directly supports the mechanism described in option A.

2. Amazon FSx for Lustre Documentation - Performance: "Amazon FSx for Lustre provides scalable, high-speed storage for High Performance Computing (HPC) workloads. It's designed for applications that require the speed and scale of a parallel file system." This confirms its suitability for the high-performance requirement.

3. Amazon S3 Documentation - S3 Intelligent-Tiering: "The S3 Intelligent-Tiering storage class is designed to optimize storage costs by automatically moving data to the most cost-effective access tier when access patterns change." This validates the cost-saving aspect for the infrequently accessed data.

4. Amazon EBS Documentation - Attach your volume to multiple instances with Amazon EBS Multi-Attach: "You can attach a single Provisioned IOPS SSD (io1 or io2) volume to a maximum of 16 Nitro-based instances in the same Availability Zone." This highlights the scalability limitation that makes option B unsuitable for "hundreds of instances".

Question: 22

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.

Correct Answer:

A

Explanation:

This solution correctly uses AWS Secrets Manager, which is the designated AWS service for securely storing, managing, and automatically rotating secrets like database credentials. The AWS::SecretsManager::RotationSchedule CloudFormation resource is a high-level, managed construct that automates the entire rotation process. It integrates directly with a specified AWS Lambda function to handle the rotation logic for supported databases like Amazon RDS for MySQL. This approach directly fulfills all requirements (secure storage,

automated rotation, CloudFormation deployment) with the least number of manually configured components, thus minimizing operational overhead.

Why Incorrect Options are Wrong:

B: This is incorrect because Parameter Store RotationSchedule is not a valid AWS CloudFormation resource type. AWS Systems Manager Parameter Store does not have a built-in, managed rotation schedule feature like Secrets Manager.

C: While technically feasible, this option has higher operational overhead. It requires manually creating and managing a separate Amazon EventBridge rule to trigger the Lambda function, whereas the RotationSchedule resource in option A handles this scheduling natively.

D: This is incorrect because AWS AppSync is a managed service for developing GraphQL APIs and has no functionality related to rotating database credentials.

References:

1. AWS Secrets Manager, Rotating Your Secrets: "For secrets that store database credentials, you can configure Secrets Manager to rotate the secret for you automatically. Rotation is the process of periodically updating a secret." This document outlines the native rotation capability.

Source: AWS Secrets Manager User Guide, "Rotate AWS Secrets Manager secrets".

2. AWS CloudFormation User Guide, AWS::SecretsManager::RotationSchedule: This resource "enables you to configure rotation for a secret." It specifies properties like SecretId, RotationLambdaARN, and RotationRules (which includes AutomaticallyAfterDays), demonstrating a fully managed and declarative approach to setting up rotation schedules.

Source: AWS CloudFormation User Guide, "AWS::SecretsManager::RotationSchedule".

3. AWS Secrets Manager User Guide, How Rotation Works: "When Secrets Manager rotates a secret, it calls the associated Lambda rotation function twice... Secrets Manager can call the rotation function on a schedule that you configure." This confirms the integrated Lambda-based rotation mechanism.

Source: AWS Secrets Manager User Guide, "How rotation works".

4. AWS Systems Manager User Guide, AWS Systems Manager Parameter Store: A review of the Parameter Store documentation and its associated CloudFormation resources (e.g., AWS::SSM::Parameter) shows no native, integrated rotation schedule resource, confirming that option B is based on a non-existent feature.

Source: AWS Systems Manager User Guide, "AWS Systems Manager Parameter Store".

Question: 23

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.

Correct Answer:

A, C

Explanation:

To meet the requirements, a two-part solution is necessary: sharing the central resource and automating deployment in member accounts.

First, the AWS Transit Gateway, located in the management account, must be shared with all member accounts within the AWS Organization. The designated service for sharing resources across accounts is AWS Resource Access Manager (RAM). This allows member accounts to see the shared Transit Gateway and create attachments to it.

Second, the process of creating a VPC and a transit gateway attachment in new member accounts must be automated. AWS CloudFormation StackSets are designed for this purpose, enabling the deployment of a consistent set of resources (a stack) across multiple

accounts and regions from a single management account. The stack set template would define the new VPC and the VPC transit gateway attachment, which would reference the shared Transit Gateway's ID.

Why Incorrect Options are Wrong:

B: Service Control Policies (SCPs) are used for governance and to enforce permission guardrails, not for sharing resources across accounts.

D: A "peering" transit gateway attachment is used to connect two transit gateways together, not to connect a VPC to a transit gateway.

E: AWS Service Catalog is used to create and manage a portfolio of approved IT services, not as the primary mechanism for sharing a core network resource like a Transit Gateway.

References:

1. AWS Transit Gateway Documentation, "Share a transit gateway": "To attach a VPC...that is in another AWS account to your transit gateway, you must share your transit gateway with the other account. You can share your transit gateway using the AWS Resource Access Manager (RAM). After you share a transit gateway, the account owner can view the transit gateway and create attachments to it."

2. AWS CloudFormation User Guide, "StackSets concepts": "AWS CloudFormation StackSets extends the functionality of stacks by enabling you to create, update, or delete stacks across multiple accounts and AWS Regions with a single operation...You can use stack sets to deploy a baseline set of AWS resources to new accounts within your organization."

3. AWS Transit Gateway Documentation, "Transit gateway attachments": This document clearly differentiates attachment types. A vpc attachment is for connecting a VPC, while a peering attachment is for "a peering connection with another transit gateway."

4. AWS Organizations User Guide, "Service control policies (SCPs)": "SCPs are a type of organization policy that you can use to manage permissions in your organization...SCPs don't grant permissions." This confirms SCPs are for control, not sharing.

Question: 24

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 VPThe 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 VPAll 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 VPConfigure 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.

Correct Answer:

A

Explanation:

This solution correctly uses AWS PrivateLink, which is designed for secure, private connectivity to services hosted on AWS without exposing traffic to the public internet. The company, as the service consumer, creates an interface VPC endpoint within its VPThis endpoint connects to the endpoint service created by the third-party SaaS provider. This architecture ensures traffic remains on the AWS private network. Crucially, the connection is unidirectional; it only allows the company to initiate connections to the SaaS application, preventing the third-party from accessing any resources in the company's VPAssociating a security group with the interface endpoint allows for granular, least-privilege access control from within the company's VPC.

Why Incorrect Options are Wrong:

B: An AWS Site-to-Site VPN connection encrypts traffic over the public internet, which violates the requirement for connectivity that does not traverse the internet. It also establishes bidirectional connectivity.

C: A VPC peering connection is bidirectional, allowing resources in both VPCs to communicate. This violates the strict security policy that no external resources can access the company's VPC.

D: This option incorrectly reverses the roles. The company is the service consumer and should create the interface endpoint. The third-party SaaS provider is the service provider and should create the endpoint service.

References:

1. AWS Documentation - AWS PrivateLink concepts: This document outlines the roles of service consumers and providers. The consumer creates an interface endpoint to connect to a service, while the provider creates an endpoint service to offer their application. This supports why option A is correct and D is incorrect.

Source: AWS Documentation, VPC User Guide, "AWS PrivateLink concepts", Section: "Service consumers" and "Service providers".

2. AWS Documentation - Interface VPC endpoints (AWS PrivateLink): This source confirms that interface endpoints enable private connectivity to services powered by AWS PrivateLink. It explicitly states, "Traffic between your VPC and the service does not leave the Amazon network." It also details how security groups can be associated with the endpoint network interface to control traffic, fulfilling the least privilege requirement.

Source: AWS Documentation, VPC User Guide, "Access an AWS service using an interface VPC endpoint".

3. AWS Documentation - Compare VPC peering and VPC endpoints: This documentation directly compares connectivity options. It highlights that VPC peering allows "full bidirectional connectivity," which is unsuitable for this scenario's security requirements. In contrast, it describes VPC endpoints as providing "unidirectional access" from the consumer VPC to the service VPThis justifies why option C is incorrect and A is correct.

Source: AWS Documentation, Amazon Virtual Private Cloud Connectivity Options whitepaper, Page 19, Table 3 - "Comparing AWS PrivateLink and VPC peering".

Question: 25

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.

Correct Answer:

C

Explanation:

The root cause of the buffering and timeouts is the inefficient delivery of large video files from a centralized origin (EC2 instances serving from an EFS volume). The most cost-efficient and scalable solution is to decouple the storage and delivery of this static media content. Migrating the videos to Amazon S3 provides a highly durable, scalable, and cost-effective object storage service. By then configuring an Amazon CloudFront distribution with the S3 bucket as the origin, the video files are cached at AWS edge locations globally. This significantly reduces latency for users, offloads traffic from the primary web application infrastructure (EC2/EFS), and directly resolves the performance issues. This architecture is an AWS best practice for distributing large static assets.

Why Incorrect Options are Wrong:

A: Changing the EFS performance mode to Max I/O might increase throughput but does not address the fundamental network latency issue for globally distributed users, which a CDN solves.

B: Using ephemeral instance stores for persistent blog content is architecturally unsound, complex to manage, and creates a high risk of data loss upon instance termination or failure.

D: While placing CloudFront before the ALB is a valid pattern, it leaves the videos on EFS. S3 is a more performant and cost-effective service for storing and serving large media files.

References:

1. Amazon CloudFront Developer Guide: "Using Amazon S3 origins". The documentation states, "When you use Amazon S3 as an origin for your distribution, you place the objects that you want CloudFront to deliver in an S3 bucket... CloudFront gets your objects from your origin and serves them to viewers." This supports using S3 as the origin for content delivery.

2. Amazon S3 User Guide: "Tutorial: Configuring a static website on Amazon S3". This guide demonstrates the primary use case for S3 in web architectures: serving static content. It notes, "Amazon S3 is an object storage service that offers industry-leading scalability, data availability, security, and performance," making it ideal for video files.

3. AWS Whitepaper: "Building a Media & Entertainment Transcoding Platform on AWS": This paper outlines best practices for media workflows. In the "Storage" section, it consistently recommends Amazon S3 as the "highly durable and available storage for source media files" and for storing the output of transcoding jobs, which are then delivered via CloudFront. (Specific reference to VOD workflows).

4. Amazon EFS User Guide: "Amazon EFS performance". The guide describes EFS as a service that "provides a simple, scalable, elastic file system for Linux-based workloads for use with AWS Cloud services and on-premises resources." Its primary use case is for shared file storage for applications, not as a high-throughput origin for a global CDN, for which S3 is better suited.

Question: 26

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 Lambda@Edge function.

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

Correct Answer:

B, C, F

Explanation:

This combination represents a serverless, highly available, and low-maintenance solution. An Application Load Balancer (ALB) is created to handle incoming HTTP and HTTPS traffic (B). To support HTTPS, a single SSL certificate from AWS Certificate Manager (ACM) is created with all 10 domains listed as Subject Alternative Names (SANs) and associated with the ALB's HTTPS listener (F). The ALB is configured to forward requests to an AWS Lambda function as its target (C). This Lambda function contains the business logic to read the JSON file, identify the requested domain from the event data, and return the appropriate HTTP 301/302 redirect response. This architecture avoids managing servers, aligning with the "least amount of operational effort" requirement.

Why Incorrect Options are Wrong:

A: An Amazon EC2 instance requires managing the underlying operating system and application runtime, which constitutes significant operational effort compared to a serverless approach.

D: While Amazon API Gateway can invoke Lambda, an ALB is a more direct and cost-effective service for handling simple HTTP/S traffic routing and redirection at scale.

E: Amazon CloudFront with Lambda@Edge is designed for content delivery and executing logic at edge locations globally. It is overly complex for this non-global redirection requirement.

References:

1. Application Load Balancer with Lambda: The AWS documentation explicitly describes using a Lambda function as a target for an ALThis allows the ALB to handle the web traffic ingress while Lambda executes the request-processing logic.

AWS Documentation, Elastic Load Balancing User Guide, "Lambda functions as targets for your Application Load Balancer".

2. ACM Certificate for ALB: To serve HTTPS traffic from a custom domain, an ALB listener must have an SSL/TLS certificate. ACM is the managed service for this.

AWS Documentation, Elastic Load Balancing User Guide, "Create an HTTPS listener for your Application Load Balancer", Section: "Listener security".

3. Lambda Function Logic for Redirects: The Lambda function handler receives an event from the ALB containing request details, such as the host header. The function can use this information to perform a lookup and construct a redirect response.

AWS Documentation, AWS Lambda Developer Guide, "Using AWS Lambda with Elastic Load Balancing", Section: "Response format".

4. Subject Alternative Names (SANs) in ACM: ACM certificates support including multiple domain names in a single certificate using the SAN field, which is ideal for this scenario with 10 domains.

AWS Documentation, AWS Certificate Manager User Guide, "Requesting a public certificate", Section: "Step 1: Specify domain names".

Question: 27

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 NLCreate 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 NLCreate 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 ALCreate 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.

Correct Answer:

C

Explanation:

This solution correctly addresses all requirements. A Network Load Balancer (NLB) is the appropriate choice for load balancing TCP traffic at Layer 4. By assigning one Elastic IP (EIP) address per Availability Zone to the NLB itself, the solution provides a set of stable, static public IP addresses that other companies can add to their allow lists. The NLB, being inherently highly available, distributes traffic to the backend EC2 instances across multiple Availability Zones. Using an Amazon Route 53 Alias (A) record to point the custom domain name (my.service.com) to the NLB is the most efficient and recommended method for DNS resolution with AWS resources.

Why Incorrect Options are Wrong:

A: This design bypasses the load balancer for public access by pointing DNS to the EIPs of individual instances, which defeats the goal of high availability.

B: This option is flawed because it suggests pointing DNS directly to cluster IPs, bypassing the load balancer's HA function, and uses imprecise terminology ("public IP addresses for the ECS cluster").

D: This option is incorrect because it proposes an Application Load Balancer (ALB), which operates at Layer 7 (HTTP/HTTPS) and is not suitable for generic TCP traffic. ALBs also lack static IP addresses.

References:

1. AWS Documentation, Network Load Balancers: "For an internet-facing load balancer, you can optionally specify one Elastic IP address per subnet... A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. It can handle millions of requests per second... It attempts to open a TCP connection to the selected target on the port specified in the listener configuration."

Source: AWS Documentation, What is a Network Load Balancer? and Elastic IP addresses.

2. AWS Documentation, Elastic Load Balancing Features: This document explicitly differentiates load balancer types. "Network Load Balancer is best suited for load balancing of Transmission Control Protocol (TCP), User Datagram Protocol (UDP) and Transport Layer Security (TLS) traffic where extreme performance is required." In contrast, "Application Load Balancer is best suited for load balancing of HTTP and HTTPS traffic".

Source: AWS Documentation, Elastic Load Balancing features.

3. AWS Documentation, Amazon Route 53: "Amazon Route 53 alias records provide a Route 53–specific extension to DNS functionality. Alias records let you route traffic to selected AWS resources, such as... Elastic Load Balancing load balancers... Unlike a CNAME record, you can create an alias record at the zone apex (e.g., example.com)."

Source: AWS Documentation, Choosing between alias and non-alias records.

Question: 28

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.

Correct Answer:

A, C

Explanation:

The requirement is for a serverless, automatically scaling, public HTTPS API to access Amazon DynamoDAmazon API Gateway is the appropriate service for this.

Option A is a valid solution. API Gateway REST APIs support direct AWS service integrations, allowing API methods to be mapped directly to DynamoDB actions (e.g., GetItem, PutItem) without intermediate compute. This is a fully serverless and scalable pattern.

Option C is also a valid solution. Both API Gateway HTTP APIs and REST APIs can integrate with AWS Lambda functions. The Lambda function contains the business logic to query the DynamoDB tables. This architecture is a common serverless pattern that is highly scalable and meets all requirements.

Why Incorrect Options are Wrong:

B: API Gateway HTTP APIs do not support direct AWS service integrations for services like DynamoDThis feature is specific to REST APIs.

D: AWS Global Accelerator is a networking service for improving global application availability and performance, not for creating API endpoints. Lambda@Edge integrates with Amazon CloudFront.

E: A Network Load Balancer (NLB) operates at the transport layer (Layer 4) and does not terminate HTTPS (TLS), making it unsuitable for building an HTTPS API.

References:

1. AWS API Gateway Developer Guide - Choosing between HTTP APIs and REST APIs: This document explicitly states that direct AWS service integrations are a feature of REST APIs. It notes, "For HTTP APIs, you can send incoming API requests to AWS Lambda functions or to any publicly routable HTTP endpoint." This supports why A and C are valid patterns and B is not. (Source: AWS Documentation, API Gateway Developer Guide, "Choosing between HTTP APIs and REST APIs").

2. AWS API Gateway Developer Guide - Setting up a REST API integration with an AWS service: "For an AWS service integration, you must configure both an integration request and an integration response... For example, to call the DynamoDB GetItem action, you configure the integration request..." This confirms the validity of the pattern described in option (Source: AWS Documentation, API Gateway Developer Guide, "Tutorial: Create a REST API as an AWS service proxy").

3. AWS API Gateway Developer Guide - Working with AWS Lambda integrations for HTTP APIs: This guide details how to configure an HTTP API to send requests to a Lambda function, which is the pattern described in option (Source: AWS Documentation, API Gateway Developer Guide, "Working with AWS Lambda integrations for HTTP APIs").

4. Elastic Load Balancing User Guide - Network Load Balancer: "A Network Load Balancer functions at the fourth layer of the Open Systems Interconnection (OSI) model. It can handle millions of requests per second." This confirms the NLB operates at Layer 4 and is not suitable for HTTPS API termination, invalidating option (Source: AWS Documentation, Elastic Load Balancing User Guide, "What is a Network Load Balancer?").

Question: 29

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.

Correct Answer:

A

Explanation:

The solution requires both redundancy and multi-region connectivity from the same pair of Direct Connect connections. A Direct Connect gateway is the specific AWS service designed for this purpose. It is a global resource that allows you to associate private virtual interfaces (VIFs) from any Direct Connect location and connect them to Virtual Private Gateways (VGWs) or Transit Gateways in any AWS Region.

By creating a second connection, creating new private VIFs on both, and attaching them to a Direct Connect gateway, the company achieves high availability. By associating this Direct Connect gateway with the VGW of the VPC, connectivity is established. As the company expands, it can associate the same Direct Connect gateway with VGWs in other Regions, meeting the multi-region requirement without new connections.

Why Incorrect Options are Wrong:

B: This provides redundancy for a single VPC in a single Region but does not offer a scalable way to connect to other Regions using the same connections.

C: A public virtual interface is used to access public AWS services, not for private connectivity to a VPThis configuration is technically invalid for the stated goal.

D: A Transit Gateway is a regional resource. While it can be used with Direct Connect (via a Direct Connect gateway), the Direct Connect gateway itself is the essential component for multi-region connectivity.

References:

1. AWS Direct Connect Documentation - Direct Connect gateways: "A Direct Connect gateway is a global resource. You can create the Direct Connect gateway in any Region and access it from all other Regions... You associate a Direct Connect gateway with either a virtual private gateway for a VPC or a transit gateway." This confirms its role in multi-region connectivity.

2. AWS Direct Connect Documentation - Redundancy and high availability: The "High availability" section describes a recommended architecture using two connections to a Direct Connect gateway to provide redundancy for accessing VPCs.

3. AWS Direct Connect Documentation - Virtual interfaces: "Private virtual interface: To access a VPC in any AWS Region (except the AWS China Regions), you must use a private virtual interface to connect to a Direct Connect gateway." This specifies the VIF type needed for the required solution.

4. AWS Transit Gateway Documentation - FAQs: "Q. Can I connect my AWS Direct Connect to a transit gateway? Yes, you can connect your AWS Direct Connect gateway to your transit gateway." This shows the relationship, where the Direct Connect gateway is the prerequisite for connecting DX to a TGW.

Question: 30

A company has an on-premises monitoring solution using a PostgreSQL database for persistence of events. The database is unable to scale due to heavy ingestion and it frequently runs out of storage. The company wants to create a hybrid solution and has already set up a VPN connection between its network and AWS. The solution should include the following attributes: * Managed AWS services to minimize operational complexity * A buffer that automatically scales to match the throughput of data and requires no on-going administration. * A visualization toot to create dashboards to observe events in near-real time. * Support for semi -structured JSON data and dynamic schemas. Which combination of components will enabled company to create a monitoring solution that will satisfy these requirements" (Select TWO.)

A: Use Amazon Kinesis Data Firehose to buffer events Create an AWS Lambda function 10 process and transform events

B: Create an Amazon Kinesis data stream to buffer events Create an AWS Lambda function to process and transform evens

C: Configure an Amazon Aurora PostgreSQL DB cluster to receive events Use Amazon Quick Sight to read from the database and create near-real-time visualizations and dashboards

D: Configure Amazon Elasticsearch Service (Amazon ES) to receive events Use the Kibana endpoint deployed with Amazon ES to create near-real-time visualizations and dashboards.

E: Configure an Amazon Neptune 0 DB instance to receive events Use Amazon QuickSight to read from the database and create near-real-time visualizations and dashboards

Correct Answer:

A, D

Explanation:

This solution requires two main components: an ingestion/buffering mechanism and a persistence/visualization layer.

Option A (Amazon Kinesis Data Firehose): This service perfectly fulfills the requirement for a managed, scalable buffer with no ongoing administration. Kinesis Data Firehose is a serverless offering that automatically scales to handle the throughput of incoming data and can deliver it to various destinations. It can also invoke an AWS Lambda function for data transformation, satisfying another potential need in an event processing pipeline.

Option D (Amazon Elasticsearch Service): This service is the ideal backend for storing and visualizing the monitoring events. It is purpose-built for log and event analytics, natively supporting semi-structured JSON data and dynamic schemas. The integrated Kibana endpoint provides a powerful tool for creating the required near-real-time visualizations and dashboards.

Why Incorrect Options are Wrong:

B: Amazon Kinesis Data Streams require manual management of shards for scaling, which contradicts the specific requirement for a buffer that "requires no on-going administration."

C: Amazon Aurora PostgreSQL is a relational database. While it can store JSON, it is less suitable for dynamic schemas than Elasticsearch and the company is explicitly moving away from a PostgreSQL-based solution.

E: Amazon Neptune is a graph database service designed for data with complex relationships, which is not appropriate for the time-series event monitoring data described in the scenario.

References:

1. Amazon Kinesis Data Firehose Documentation: "Amazon Kinesis Data Firehose is a fully managed service for delivering real-time streaming data... It automatically scales to match the throughput of your data and requires no ongoing administration."

Source: AWS Documentation, "What Is Amazon Kinesis Data Firehose?", Introduction.

2. Amazon OpenSearch Service (formerly Elasticsearch Service) Documentation: "You can use Amazon OpenSearch Service to search, analyze, and visualize data in real time... a popular choice for use cases such as log analytics, real-time application monitoring, and clickstream analytics." and "It stores data as JSON documents."

Source: AWS Documentation, "What Is Amazon OpenSearch Service?", Introduction.

3. Kibana with Amazon OpenSearch Service: "OpenSearch Service provides an installation of OpenSearch Dashboards (or Kibana, for legacy Elasticsearch versions) with every OpenSearch Service domain. You can use Dashboards to visualize your data and build dashboards to share with others."

Source: AWS Documentation, "Using OpenSearch Dashboards with Amazon OpenSearch Service".

4. Amazon Kinesis Data Streams Documentation: "The unit of throughput of an Amazon Kinesis data stream is a shard... You can increase or decrease the number of shards in a stream as needed."

Source: AWS Documentation, "Amazon Kinesis Data Streams Terminology and Concepts", Shard section. This implies manual capacity management.