Amazon

(AWS Certified Solutions Architect - Associate SAA-C03)

AWS Certified Solutions Architect - Associate SAA-C03

Total: **1019 Questions**

Link:

Question: 1

A company collects data for temperature, humidity, and atmospheric pressure in cities across multiple continents. The average volume of data that the company collects from each site daily is 500 GB. Each site has a high-speed Internet connection.
The company wants to aggregate the data from all these global sites as quickly as possible in a single Amazon S3 bucket. The solution must minimize operational complexity.
Which solution meets these requirements?

A. Turn on S3 Transfer Acceleration on the destination S3 bucket. Use multipart uploads to directly upload site data to the destination S3 bucket.

B. Upload the data from each site to an S3 bucket in the closest Region. Use S3 Cross-Region Replication to copy objects to the destination S3 bucket. Then remove the data from the origin S3 bucket.

C. Schedule AWS Snowball Edge Storage Optimized device jobs daily to transfer data from each site to the closest Region. Use S3 Cross-Region Replication to copy objects to the destination S3 bucket.

D. Upload the data from each site to an Amazon EC2 instance in the closest Region. Store the data in an Amazon Elastic Block Store (Amazon EBS) volume. At regular intervals, take an EBS snapshot and copy it to the Region that contains the destination S3 bucket. Restore the EBS volume in that Region.

Answer: A

Explanation:

The correct answer is A because it directly addresses the requirements of speed, minimal operational complexity, and leveraging existing high-speed internet connections. S3 Transfer Acceleration utilizes AWS's globally distributed edge locations to optimize data transfer speeds into an S3 bucket. Multipart uploads enhance reliability and speed, especially for large files (500 GB daily). This method avoids the complexity of managing intermediate buckets, EC2 instances, or physical devices like Snowball Edge.

Option B introduces unnecessary complexity with multiple S3 buckets and cross-region replication, increasing management overhead and costs. While replication handles data transfer, Transfer Acceleration is designed specifically for speed optimization in direct uploads.

Option C is unsuitable because AWS Snowball Edge is intended for environments with limited or no internet connectivity. Given the high-speed internet connection available at each site, Snowball Edge adds unnecessary logistical complexity and delays.

Option D involves managing EC2 instances, EBS volumes, and snapshots, significantly increasing operational overhead. Transferring EBS snapshots is also not an optimized method for data aggregation into S3 compared to direct S3 uploads, especially regarding speed and cost. The described method is more appropriate for disaster recovery of complete systems rather than daily data aggregation.

In summary, S3 Transfer Acceleration with multipart uploads is the most efficient and straightforward solution for quickly aggregating data from global sites into a single S3 bucket, aligning with the requirements for speed, minimal operational overhead, and leveraging existing high-speed internet.

Relevant links:

Amazon S3 Transfer Acceleration: https://aws.amazon.com/s3/transfer-acceleration/
Amazon S3 Multipart Upload Overview:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/mpuoverview.html

Question: 2

A company needs the ability to analyze the log files of its proprietary application. The logs are stored in JSON format in an Amazon S3 bucket. Queries will be simple and will run on-demand. A solutions architect needs to perform the analysis with minimal changes to the existing architecture.

What should the solutions architect do to meet these requirements with the LEAST amount of operational overhead?

A. Use Amazon Redshift to load all the content into one place and run the SQL queries as needed.

B. Use Amazon CloudWatch Logs to store the logs. Run SQL queries as needed from the Amazon CloudWatch console.

C. Use Amazon Athena directly with Amazon S3 to run the queries as needed.

D. Use AWS Glue to catalog the logs. Use a transient Apache Spark cluster on Amazon EMR to run the SQL queries as needed.

Answer: C

Explanation:

The best solution is **C. Use Amazon Athena directly with Amazon S3 to run the queries as needed.**

Here's why:

Minimal Operational Overhead: Athena is serverless. It eliminates the need to provision or manage infrastructure. It also allows you to query data directly in S3, which is a major advantage as the logs are already stored there.

Cost-Effectiveness: Athena charges based on the amount of data scanned per query. Since the queries are on-demand and presumably infrequent, this pay-per-query model is the most cost-effective option. **Simplicity:** Athena allows direct querying of JSON data stored in S3 using standard SQL. The log format is already compatible.

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

A. Amazon Redshift: Redshift requires setting up and managing a data warehouse cluster. This involves provisioning resources, handling scaling, and performing ETL (Extract, Transform, Load) to move the JSON data from S3 into Redshift. This adds significant operational overhead and cost compared to Athena.

B. Amazon CloudWatch Logs: CloudWatch Logs are best suited for real-time monitoring and centralized logging, not for complex analytical queries. While CloudWatch Logs Insights exists, its query language isn't SQL, and it's not designed for ad-hoc analysis of JSON files stored elsewhere (like S3). Migrating the logs would also be required.

D. AWS Glue and Amazon EMR: This solution is an overkill. AWS Glue is for ETL and data cataloging, and EMR is for big data processing using frameworks like Spark. While suitable for very large and complex data analysis scenarios, it introduces unnecessary complexity and operational overhead for the stated requirements of simple, on-demand queries. It also has high cost compared to using AWS Athena.

Therefore, Athena aligns best with the requirement of minimal operational overhead and allows simple SQL queries on JSON logs stored in S3, making it the superior choice.

Authoritative Links:

Amazon Athena:https://aws.amazon.com/athena/
Amazon S3:https://aws.amazon.com/s3/

Question: 3

A company uses AWS Organizations to manage multiple AWS accounts for different departments. The management account has an Amazon S3 bucket that contains project reports. The company wants to limit access to this S3 bucket to only users of accounts within the organization in AWS Organizations.
Which solution meets these requirements with the LEAST amount of operational overhead?

A. Add the aws PrincipalOrgID global condition key with a reference to the organization ID to the S3 bucket policy.

B. Create an organizational unit (OU) for each department. Add the aws:PrincipalOrgPaths global condition key to the S3 bucket policy.

C. Use AWS CloudTrail to monitor the CreateAccount, InviteAccountToOrganization, LeaveOrganization, and RemoveAccountFromOrganization events. Update the S3 bucket policy accordingly.

D. Tag each user that needs access to the S3 bucket. Add the aws:PrincipalTag global condition key to the S3 bucket policy.

Answer: A

Explanation:

The correct answer is A. Here's why:

Option A leverages the aws:PrincipalOrgID global condition key in the S3 bucket policy. This condition key directly references the AWS Organizations organization ID. When applied, the S3 bucket will only allow access from AWS accounts that belong to the specified organization. This approach offers the least operational overhead because it's a simple, direct configuration within the S3 bucket policy, automatically encompassing all current and future accounts within the organization without requiring ongoing updates.

Option B, using aws:PrincipalOrgPaths, involves creating organizational units (OUs) for each department. While it provides more granular control based on OU membership, it necessitates managing and updating the S3 bucket policy whenever OUs change or accounts are moved, increasing operational overhead. It's unnecessary complexity for the stated requirement of granting access to all accounts within the entire organization.

Option C, using CloudTrail to monitor organizational changes and then updating the S3 bucket policy, is unnecessarily complex and creates significant operational overhead. It requires implementing a custom solution to react to CloudTrail events and programmatically modify the S3 bucket policy, which is prone to errors and maintenance issues. It is also not a real-time mechanism; there would be a delay between the organizational event and the policy update.

Option D, tagging users and using aws:PrincipalTag, is suitable for controlling access based on individual user attributes, not organizational membership. Applying and managing tags for each user across multiple accounts within the organization adds significant operational overhead and is not the appropriate tool for this specific requirement. Furthermore, tagging users across multiple accounts and keeping those tags consistent introduces administrative challenges.

Therefore, option A provides the most efficient and least complex solution for limiting S3 bucket access to only users of accounts within the organization, by directly referencing the organization ID in the S3 bucket policy, minimizing the manual intervention and operational overhead.

Relevant links for further research:

AWS Organizations Condition Keys
Controlling Access to S3 Buckets
AWS Organizations Overview

Question: 4

An application runs on an Amazon EC2 instance in a VPC. The application processes logs that are stored in an Amazon S3 bucket. The EC2 instance needs to access the S3 bucket without connectivity to the internet. Which solution will provide private network connectivity to Amazon S3?

A. Create a gateway VPC endpoint to the S3 bucket.

B. Stream the logs to Amazon CloudWatch Logs. Export the logs to the S3 bucket.

C. Create an instance profile on Amazon EC2 to allow S3 access.

D. Create an Amazon API Gateway API with a private link to access the S3 endpoint.

Answer: A

Explanation:

The correct answer is **A. Create a gateway VPC endpoint to the S3 bucket.**

Here's why:

Gateway VPC Endpoints for S3: Gateway VPC endpoints provide private connectivity between your VPC and S3 without requiring an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. This ensures that traffic between your EC2 instance and S3 remains within the AWS network.
https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints-s3.html

How it works: When you create a gateway VPC endpoint for S3, a route is automatically added to your VPC's route table. This route directs traffic destined for S3 to the endpoint instead of the internet. The EC2 instance, using its IAM role permissions to access S3, can now reach the S3 bucket privately.

Let's analyze the other options:

B. Stream the logs to Amazon CloudWatch Logs. Export the logs to the S3 bucket: While CloudWatch Logs is useful for centralized logging, exporting logs from CloudWatch Logs to S3 still requires connectivity to S3. It doesn't inherently solve the private connectivity requirement. Furthermore, it adds unnecessary complexity and cost.

C. Create an instance profile on Amazon EC2 to allow S3 access: An instance profile (IAM role) grants permissions to the EC2 instance to access S3, but it doesn't establish private network connectivity. The EC2 instance still needs a way to reach S3 over the network, and without a gateway endpoint, it would need internet access.

D. Create an Amazon API Gateway API with a private link to access the S3 endpoint: This involves creating an API Gateway and configuring it to use a VPC endpoint service (PrivateLink). While PrivateLink can provide private connectivity, it is typically used for exposing services running within a VPC to other VPCs or on-premises networks, not for a simple EC2 to S3 communication within the same VPC. It is overkill for this scenario and adds significant complexity. Additionally, API Gateway adds latency and cost where the simple gateway endpoint of option A suffices.

Question: 5

A company is hosting a web application on AWS using a single Amazon EC2 instance that stores user-uploaded documents in an Amazon EBS volume. For better scalability and availability, the company duplicated the architecture and created a second EC2 instance and EBS volume in another Availability Zone, placing both behind an Application Load Balancer. After completing this change, users reported that, each time they refreshed the website, they could see one subset of their documents or the other, but never all of the documents at the same time.
What should a solutions architect propose to ensure users see all of their documents at once?

A. Copy the data so both EBS volumes contain all the documents

B. Configure the Application Load Balancer to direct a user to the server with the documents

C. Copy the data from both EBS volumes to Amazon EFS. Modify the application to save new documents to Amazon EFS

D. Configure the Application Load Balancer to send the request to both servers. Return each document from the correct server

Answer: C

Explanation:

The correct answer is C because it addresses the core issue of data consistency across multiple instances. The users are experiencing inconsistent document views because each EC2 instance has a separate EBS volume, and data isn't synchronized between them.

Option A, copying data between EBS volumes, is a short-term fix that becomes increasingly difficult to manage as the data grows and changes. It doesn't provide a scalable or reliable long-term solution. Consider the overhead involved in constantly syncing large EBS volumes.

Option B, using the Application Load Balancer to direct users to the server containing the documents, isn't feasible. The load balancer isn't aware of which server holds which documents. It would require complex session management based on document ownership, which is error-prone and inefficient.

Option D, sending requests to both servers simultaneously, won't resolve the issue. It might even exacerbate the problem by showing fragmented document sets more frequently. There would be significant overhead in merging results from multiple servers and the application would need to manage this complexity.

Option C, migrating to Amazon EFS, provides a centralized, shared file system accessible by both EC2 instances. EFS ensures data consistency and allows both instances to see the same set of documents. By copying the existing data to EFS and modifying the application to use EFS for storage, all users will access the same data, resolving the inconsistency. EFS is designed for this exact scenario - providing shared storage for multiple compute instances.

Amazon EFS is well-suited for web applications that require persistent, shared storage. It eliminates the need for data replication and synchronization across instances. It also offers scalability and performance to accommodate growing data needs.

Therefore, migrating to Amazon EFS is the most effective and scalable solution for ensuring users see all their documents simultaneously.

Further reading:

Amazon EFS:https://aws.amazon.com/efs/
Application Load Balancer:https://aws.amazon.com/elasticloadbalancing/application-load-balancer/

Question: 6

A company uses NFS to store large video files in on-premises network attached storage. Each video file ranges in size from 1 MB to 500 GB. The total storage is 70 TB and is no longer growing. The company decides to migrate the video files to Amazon S3. The company must migrate the video files as soon as possible while using the least possible network bandwidth.
Which solution will meet these requirements?

A. Create an S3 bucket. Create an IAM role that has permissions to write to the S3 bucket. Use the AWS CLI to copy all files locally to the S3 bucket.

B. Create an AWS Snowball Edge job. Receive a Snowball Edge device on premises. Use the Snowball Edge client to transfer data to the device. Return the device so that AWS can import the data into Amazon S3.

C. Deploy an S3 File Gateway on premises. Create a public service endpoint to connect to the S3 File Gateway. Create an S3 bucket. Create a new NFS file share on the S3 File Gateway. Point the new file share to the S3 bucket. Transfer the data from the existing NFS file share to the S3 File Gateway.

D. Set up an AWS Direct Connect connection between the on-premises network and AWS. Deploy an S3 File Gateway on premises. Create a public virtual interface (VIF) to connect to the S3 File Gateway. Create an S3 bucket. Create a new NFS file share on the S3 File Gateway. Point the new file share to the S3 bucket. Transfer the data from the existing NFS file share to the S3 File Gateway.

Answer: B

Explanation:

The correct solution is B: Using AWS Snowball Edge.

Here's a detailed justification:

The key requirements are migrating 70 TB of video files to S3 quickly while minimizing network bandwidth usage.

Option A (AWS CLI): Copying 70 TB over the internet using the AWS CLI would be extremely slow and consume a significant amount of network bandwidth. This contradicts the requirements.

Option B (AWS Snowball Edge): AWS Snowball Edge is designed for transferring large amounts of data offline. AWS ships a physical device (Snowball Edge) to the company. The company then copies the data onto the device locally. Once the transfer is complete, the device is shipped back to AWS, where the data is uploaded to S3. This avoids network bandwidth usage and enables a much faster transfer compared to online methods for this volume of data. The 70 TB size falls within the typical Snowball Edge capacity. This aligns perfectly with the requirements of minimizing bandwidth and ensuring a swift transfer.

Option C (S3 File Gateway with Public Service Endpoint): S3 File Gateway would cache the data before uploading it to S3. While it provides an NFS interface, it still requires transferring the data over the internet, which contradicts the requirement of minimizing network bandwidth. A public service endpoint still means data traversing the public internet. Furthermore, introducing File Gateway for a one-time migration adds unnecessary complexity.

Option D (S3 File Gateway with AWS Direct Connect): While Direct Connect provides a dedicated, faster connection to AWS, setting it up solely for a one-time migration is an overkill and adds significant cost and complexity. It doesn't eliminate the need to transfer 70 TB of data over a network.

Therefore, AWS Snowball Edge is the optimal solution because it directly addresses the constraints of large data volume and limited network bandwidth. It is significantly faster than transferring over the internet and avoids ongoing network usage.

Supporting concepts and links:

AWS Snowball Edge: AWS Snowball Edge is a data migration and edge computing device that comes in various configurations, including Storage Optimized. It allows customers to transfer large amounts of data into and out of AWS without relying on network connectivity.

https://aws.amazon.com/snowball/

Data Migration to AWS: AWS provides several services for data migration, each suitable for different scenarios. Snowball Edge is typically used for large-scale migrations when network bandwidth is limited.

https://aws.amazon.com/migration/

Final Answer: The final answer is $\boxed B $

Question: 7

A company has an application that ingests incoming messages. Dozens of other applications and microservices then quickly consume these messages. The number of messages varies drastically and sometimes increases suddenly to 100,000 each second. The company wants to decouple the solution and increase scalability. Which solution meets these requirements?

A. Persist the messages to Amazon Kinesis Data Analytics. Configure the consumer applications to read and process the messages.

B. Deploy the ingestion application on Amazon EC2 instances in an Auto Scaling group to scale the number of EC2 instances based on CPU metrics.

C. Write the messages to Amazon Kinesis Data Streams with a single shard. Use an AWS Lambda function to preprocess messages and store them in Amazon DynamoDB. Configure the consumer applications to read from DynamoDB to process the messages.

D. Publish the messages to an Amazon Simple Notification Service (Amazon SNS) topic with multiple Amazon Simple Queue Service (Amazon SOS) subscriptions. Configure the consumer applications to process the messages from the queues.

Answer: D

Explanation:

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

The scenario demands a highly scalable and decoupled message ingestion and consumption system. Let's analyze why option D, using Amazon SNS and SQS, best fulfills these requirements.

Option D (SNS and SQS) utilizes the publish-subscribe pattern offered by SNS and the message queuing capabilities of SQS. The ingestion application publishes messages to an SNS topic. Multiple SQS queues subscribe to this topic. Each consumer application then pulls messages from its own dedicated SQS queue. This provides excellent decoupling because the ingestion application is unaware of the consumers, and the consumers are isolated from each other. This isolation is crucial for scalability; if one consumer fails or becomes overloaded, it doesn't impact the others or the ingestion process. SQS provides buffering for messages, preventing data loss during surges.

SNS is designed for high-throughput, enabling it to handle the peak load of 100,000 messages per second.
SQS, being a fully managed queuing service, automatically scales to handle the message volume. The combination ensures messages are reliably delivered to all subscribed consumers without overwhelming them. SQS also facilitates asynchronous processing, allowing consumers to process messages at their own pace.

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

Option A (Kinesis Data Analytics) is designed for real-time data processing and analytics, not primarily for decoupling and distribution to multiple consumers. While it can process messages, it's not as efficient for simply fanning them out to many downstream applications.

Option B (EC2 Auto Scaling) addresses the scaling of the ingestion application, but not the decoupling or scaling of the message delivery to consumers. It doesn't solve the problem of distributing messages effectively to multiple, independent consumers.

Option C (Kinesis Data Streams and DynamoDB) introduces unnecessary complexity. Kinesis Data Streams can handle high throughput, but using a single shard would create a bottleneck. DynamoDB as an intermediary storage adds latency and complexity compared to the direct delivery from SQS. Furthermore, polling DynamoDB would not scale nearly as efficiently as SQS queues.

In summary, option D is the most appropriate choice because it directly addresses the requirements for decoupling, scalability, and reliable message delivery to multiple consumers using services specifically designed for these purposes.

Supporting Links:

Amazon SNS:https://aws.amazon.com/sns/
Amazon SQS:https://aws.amazon.com/sqs/
Decoupled Architecture on AWS:https://aws.amazon.com/solutions/guidance/decoupling-applications-aws/

Question: 8

A company is migrating a distributed application to AWS. The application serves variable workloads. The legacy platform consists of a primary server that coordinates jobs across multiple compute nodes. The company wants to modernize the application with a solution that maximizes resiliency and scalability.
How should a solutions architect design the architecture to meet these requirements?

A. Configure an Amazon Simple Queue Service (Amazon SQS) queue as a destination for the jobs. Implement the compute nodes with Amazon EC2 instances that are managed in an Auto Scaling group. Configure EC2 Auto Scaling to use scheduled scaling.

B. Configure an Amazon Simple Queue Service (Amazon SQS) queue as a destination for the jobs. Implement the compute nodes with Amazon EC2 instances that are managed in an Auto Scaling group. Configure EC2 Auto Scaling based on the size of the queue.

C. Implement the primary server and the compute nodes with Amazon EC2 instances that are managed in an Auto Scaling group. Configure AWS CloudTrail as a destination for the jobs. Configure EC2 Auto Scaling based on the load on the primary server.

D. Implement the primary server and the compute nodes with Amazon EC2 instances that are managed in an Auto Scaling group. Configure Amazon EventBridge (Amazon CloudWatch Events) as a destination for the jobs. Configure EC2 Auto Scaling based on the load on the compute nodes.

Answer: B

Explanation:

The correct answer is B. Here's why:

The problem statement emphasizes resiliency, scalability, and variable workloads. Option B best addresses these requirements.

Amazon SQS: Using SQS decouples the primary server (job coordinator) from the compute nodes. This promotes resiliency. If the primary server fails, the messages in the queue persist, and processing can resume when a new primary server is available. https://aws.amazon.com/sqs/
EC2 Auto Scaling Group: Placing the compute nodes in an Auto Scaling group provides scalability and high availability. Auto Scaling automatically adjusts the number of EC2 instances based on demand. This ensures that the application can handle variable workloads efficiently. https://aws.amazon.com/autoscaling/ **Scaling based on Queue Size:** Configuring Auto Scaling to scale based on the size of the SQS queue is crucial. As the queue grows (more jobs are waiting), Auto Scaling will launch more EC2 instances to process the jobs. Conversely, if the queue shrinks, Auto Scaling can terminate instances to reduce costs. This reactive, queue-driven scaling aligns directly with the variable workload requirement.

Option A is incorrect because scheduled scaling doesn't react to real-time workload changes. It scales instances at predefined times regardless of the actual queue size. While useful for predictable workload patterns, it's less effective for variable workloads.

Option C is incorrect because CloudTrail is an auditing service, not a job queue. Furthermore, scaling based on the primary server's load doesn't directly reflect the number of waiting jobs. If the primary server is overloaded, that is a symptom of a problem, not the core driver for scaling compute nodes.

Option D is incorrect because EventBridge (formerly CloudWatch Events) is typically used for event-driven architectures, not as a direct queue for jobs in this scenario. Also, scaling based on the load of the compute nodes becomes a lagging metric. The queue length provides earlier indication that scaling is needed.
Furthermore, managing the primary server within the auto-scaling group might be more complex than necessary and is less resilient compared to the option leveraging SQS.

Question: 9

A company is running an SMB file server in its data center. The file server stores large files that are accessed frequently for the first few days after the files are created. After 7 days the files are rarely accessed.
The total data size is increasing and is close to the company's total storage capacity. A solutions architect must increase the company's available storage space without losing low-latency access to the most recently accessed files. The solutions architect must also provide file lifecycle management to avoid future storage issues. Which solution will meet these requirements?

A. Use AWS DataSync to copy data that is older than 7 days from the SMB file server to AWS.

B. Create an Amazon S3 File Gateway to extend the company's storage space. Create an S3 Lifecycle policy to transition the data to S3 Glacier Deep Archive after 7 days.

C. Create an Amazon FSx for Windows File Server file system to extend the company's storage space.

D. Install a utility on each user's computer to access Amazon S3. Create an S3 Lifecycle policy to transition the data to S3 Glacier Flexible Retrieval after 7 days.

Answer: B

Explanation:

The correct solution is B, which involves using Amazon S3 File Gateway and S3 Lifecycle policies. Here's why:

S3 File Gateway: This service provides a seamless way to extend on-premises file storage to Amazon S3 without requiring significant application changes. It presents a local file system interface to the on-premises SMB server, which then transparently caches frequently accessed data locally while storing the full dataset in S3. This addresses the requirement for low-latency access to recent files.
https://aws.amazon.com/storagegateway/file-gateway/

S3 Lifecycle Policies: These policies automate the process of moving less frequently accessed data to lower-cost storage tiers such as S3 Glacier Deep Archive after a specified period (7 days in this case). This addresses the requirement for file lifecycle management and helps to reduce storage costs without deleting the data. S3 Glacier Deep Archive is suitable for long-term archival where retrieval times of several hours are acceptable.
https://aws.amazon.com/s3/storage-classes/

Option A is incorrect because AWS DataSync is primarily a data migration tool, not a continuous storage extension solution. It would require ongoing manual or scripted execution to move files after 7 days, which is less efficient than a managed solution like S3 File Gateway with lifecycle policies.

Option C is not ideal because Amazon FSx for Windows File Server is a fully managed Windows file server in the cloud. While it provides scalable storage, it doesn't directly address the need to seamlessly integrate with the existing on-premises SMB server and leverage cheaper archival storage. It would be better suited as a complete replacement to the on-premise server.

Option D is incorrect because it involves manually installing a utility on each user's computer. It is complex to manage and prone to user error. Additionally, while S3 Lifecycle policies are used, S3 Glacier Flexible Retrieval is a more expensive storage option than Glacier Deep Archive for archival purposes. This is not an ideal solution.

In summary, option B provides the most cost-effective and manageable solution for extending on-premises file storage to the cloud, maintaining low-latency access to recent files, and implementing automated lifecycle management.

Question: 10

A company is building an ecommerce web application on AWS. The application sends information about new

orders to an Amazon API Gateway REST API to process. The company wants to ensure that orders are processed in the order that they are received.
Which solution will meet these requirements?

A. Use an API Gateway integration to publish a message to an Amazon Simple Notification Service (Amazon SNS) topic when the application receives an order. Subscribe an AWS Lambda function to the topic to perform processing.

B. Use an API Gateway integration to send a message to an Amazon Simple Queue Service (Amazon SQS) FIFO queue when the application receives an order. Configure the SQS FIFO queue to invoke an AWS Lambda function for processing.

C. Use an API Gateway authorizer to block any requests while the application processes an order.

D. Use an API Gateway integration to send a message to an Amazon Simple Queue Service (Amazon SQS) standard queue when the application receives an order. Configure the SQS standard queue to invoke an AWS Lambda function for processing.

Answer: B

Explanation:

The correct answer is B because it directly addresses the requirement of processing orders in the order they are received. Here's why:

FIFO (First-In, First-Out) Queues: Amazon SQS FIFO queues guarantee that messages are retrieved from the queue in the exact order they were placed. This is crucial for order processing, where the sequence of events matters.
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/FIFO-queues.html

API Gateway Integration: Integrating API Gateway with SQS allows you to seamlessly send order data received through the API directly into the FIFO queue.

Lambda Function Invocation: Configuring the SQS queue to invoke a Lambda function on message arrival ensures that each order is automatically processed as it becomes available in the queue.

Option A is incorrect because Amazon SNS is a publish/subscribe service, not a queuing service. It's designed for broadcasting messages to multiple subscribers simultaneously, not for preserving the order of messages. This makes SNS unsuitable for ordered processing.

Option C is incorrect because API Gateway authorizers are used for authentication and authorization, not for controlling the processing order of requests. Blocking requests with an authorizer might prevent processing altogether, but won't guarantee correct sequencing.

Option D is incorrect because Amazon SQS standard queues do not guarantee message order. While they aim for best-effort ordering, messages can sometimes be delivered out of sequence, which violates the stated requirement.
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/standard-queues.html

In summary, using an API Gateway integrated with an SQS FIFO queue and a Lambda function ensures that order information is captured in the correct sequence and then processed sequentially, meeting the application's specific requirements. This approach leverages the ordered message delivery capability of FIFO queues to maintain data integrity within the ecommerce application's workflow.

Question: 11

A company has an application that runs on Amazon EC2 instances and uses an Amazon Aurora database. The EC2 instances connect to the database by using user names and passwords that are stored locally in a file. The company wants to minimize the operational overhead of credential management.

What should a solutions architect do to accomplish this goal?

A. Use AWS Secrets Manager. Turn on automatic rotation.

B. Use AWS Systems Manager Parameter Store. Turn on automatic rotation.

C. Create an Amazon S3 bucket to store objects that are encrypted with an AWS Key Management Service (AWS KMS) encryption key. Migrate the credential file to the S3 bucket. Point the application to the S3 bucket.

D. Create an encrypted Amazon Elastic Block Store (Amazon EBS) volume for each EC2 instance. Attach the new EBS volume to each EC2 instance. Migrate the credential file to the new EBS volume. Point the application to the new EBS volume.

Answer: A

Explanation:

The best solution is to use AWS Secrets Manager with automatic rotation. Here's why:

Secrets Manager Purpose: AWS Secrets Manager is specifically designed for securely storing and managing secrets, like database credentials. It eliminates the need to hardcode or store secrets within application code or configuration files.

Reduced Operational Overhead: By using Secrets Manager, the company centralizes credential management, which simplifies the process of updating, rotating, and auditing secrets. This reduces the operational burden associated with managing credentials manually on each EC2 instance.

Automatic Rotation: Secrets Manager's automatic rotation feature automates the process of changing credentials regularly without application downtime. This enhances security by limiting the window of opportunity for compromised credentials to be exploited.

IAM Integration: Secrets Manager integrates with AWS Identity and Access Management (IAM) to control access to secrets, ensuring that only authorized EC2 instances or roles can retrieve the database credentials.

Auditing and Monitoring: Secrets Manager provides auditing capabilities through AWS CloudTrail, allowing the company to track secret access and modifications.

Why other options are less suitable:

AWS Systems Manager Parameter Store: While Parameter Store can store sensitive information, it's primarily designed for storing configuration data and not for managing secrets with rotation capabilities.

While Parameter Store offers SecureString parameters, it lacks the robust rotation features of Secrets Manager.

Amazon S3: Storing credentials in an S3 bucket, even if encrypted, is not the best practice for secret management. It requires custom logic to retrieve and manage the credentials, which increases operational overhead and doesn't provide automatic rotation.

Encrypted EBS Volume: Storing credentials on encrypted EBS volumes attached to each EC2 instance offers some level of security, but it doesn't centralize credential management or provide automatic rotation capabilities. It also requires managing separate volumes for each instance.

In summary, AWS Secrets Manager with automatic rotation is the most efficient and secure way to manage database credentials for EC2 instances connecting to Aurora, minimizing operational overhead and enhancing security posture.

Authoritative Links:

AWS Secrets Manager:https://aws.amazon.com/secrets-manager/
Rotate AWS Secrets Manager secrets:
https://docs.aws.amazon.com/secretsmanager/latest/userguide/rotating-secrets.html

Question: 12

A global company hosts its web application on Amazon EC2 instances behind an Application Load Balancer (ALB). The web application has static data and dynamic data. The company stores its static data in an Amazon S3 bucket.
The company wants to improve performance and reduce latency for the static data and dynamic data. The company is using its own domain name registered with Amazon Route 53.
What should a solutions architect do to meet these requirements?

A. Create an Amazon CloudFront distribution that has the S3 bucket and the ALB as origins. Configure Route 53 to route traffic to the CloudFront distribution.

B. Create an Amazon CloudFront distribution that has the ALB as an origin. Create an AWS Global Accelerator standard accelerator that has the S3 bucket as an endpoint Configure Route 53 to route traffic to the CloudFront distribution.

C. Create an Amazon CloudFront distribution that has the S3 bucket as an origin. Create an AWS Global Accelerator standard accelerator that has the ALB and the CloudFront distribution as endpoints. Create a custom domain name that points to the accelerator DNS name. Use the custom domain name as an endpoint for the web application.

D. Create an Amazon CloudFront distribution that has the ALB as an origin. Create an AWS Global Accelerator standard accelerator that has the S3 bucket as an endpoint. Create two domain names. Point one domain name to the CloudFront DNS name for dynamic content. Point the other domain name to the accelerator DNS name for static content. Use the domain names as endpoints for the web application.

Answer: A

Explanation:

The best solution is to use CloudFront with both the S3 bucket (for static content) and the ALB (for dynamic content) as origins and then point Route 53 to the CloudFront distribution. This approach efficiently leverages CloudFront's caching capabilities to reduce latency and improve performance for both types of content globally.

Here's why other options are less optimal:

Option B & D: While Global Accelerator can improve performance, it primarily focuses on TCP and UDP traffic, not HTTP/HTTPS, which are used by web applications. Also, Global Accelerator is an overkill for static content served from S3 and adds unnecessary complexity. Global Accelerator is most useful for improving the performance of applications with users distributed globally where network congestion or routing issues can impact performance. Furthermore, managing separate domain names, as suggested in option D, would complicate the web application's architecture and potentially lead to inconsistent user experience.

Option C: Using Global Accelerator as a front for CloudFront is not a typical use case. CloudFront is already a global content delivery network (CDN) designed to optimize content delivery. Global Accelerator's benefits would be redundant.

Option A directly addresses the requirements by:

1. **Improving Performance and Reducing Latency:** CloudFront caches static content from the S3 bucket at edge locations globally, bringing data closer to users and reducing latency.
2. **Serving Dynamic Content:** CloudFront can also be configured to forward requests for dynamic content to the ALB. This allows CloudFront to cache the responses when possible, further improving performance.
3. **Using Existing Domain Name:** Route 53 is configured to route traffic to the CloudFront distribution, which means you can continue using your existing domain name.

Authoritative Links:

Amazon CloudFront:https://aws.amazon.com/cloudfront/
Amazon S3:https://aws.amazon.com/s3/
Application Load Balancer:https://aws.amazon.com/elasticloadbalancing/application-load-balancer/ **AWS Global Accelerator:**https://aws.amazon.com/global-accelerator/

Amazon Route 53:https://aws.amazon.com/route53/

Question: 13

A company performs monthly maintenance on its AWS infrastructure. During these maintenance activities, the company needs to rotate the credentials for its Amazon RDS for MySQL databases across multiple AWS Regions. Which solution will meet these requirements with the LEAST operational overhead?

A. Store the credentials as secrets in AWS Secrets Manager. Use multi-Region secret replication for the required Regions. Configure Secrets Manager to rotate the secrets on a schedule.

B. Store the credentials as secrets in AWS Systems Manager by creating a secure string parameter. Use multi-Region secret replication for the required Regions. Configure Systems Manager to rotate the secrets on a schedule.

C. Store the credentials in an Amazon S3 bucket that has server-side encryption (SSE) enabled. Use Amazon EventBridge (Amazon CloudWatch Events) to invoke an AWS Lambda function to rotate the credentials.

D. Encrypt the credentials as secrets by using AWS Key Management Service (AWS KMS) multi-Region customer managed keys. Store the secrets in an Amazon DynamoDB global table. Use an AWS Lambda function to retrieve the secrets from DynamoDB. Use the RDS API to rotate the secrets.

Answer: A

Explanation:

Here's a detailed justification for why option A is the best solution for rotating RDS for MySQL database credentials across multiple AWS Regions with the least operational overhead:

Option A leverages AWS Secrets Manager's built-in capabilities for secret management and rotation across Regions. Secrets Manager is designed specifically for storing and managing secrets like database credentials. Its multi-Region secret replication feature automatically replicates secrets to specified Regions, ensuring consistency. The automatic rotation feature allows you to define a rotation schedule, where Secrets Manager handles the process of generating new credentials and updating them in the database, significantly reducing manual intervention and operational overhead.

Option B is incorrect because AWS Systems Manager Parameter Store, while capable of storing secrets, doesn't natively support multi-Region secret replication or automated rotation in the same streamlined manner as Secrets Manager. Implementing similar functionality with Parameter Store would require custom scripting and automation, increasing operational complexity.

Option C, using S3 and Lambda, involves more manual setup and management. While S3 can store encrypted data, it doesn't inherently offer secret rotation capabilities. The Lambda function would need to be custom-coded to generate new credentials, update the database, and update the secret in S3, adding to operational overhead and potential error points.

Option D, using DynamoDB and Lambda with KMS encryption, is the most complex. DynamoDB global tables provide replication, but managing the encryption with KMS and implementing secret rotation logic within Lambda, including calling the RDS API, adds significant overhead. KMS is mainly for encryption key management, not secret rotation.

Therefore, option A provides the most straightforward and automated solution for rotating RDS database credentials across multiple Regions due to Secrets Manager's features like built-in rotation and multi-region replication, minimizing operational overhead.

Relevant Links:

AWS Secrets Manager:https://aws.amazon.com/secrets-manager/
Secrets Manager Multi-Region Secrets:

https://docs.aws.amazon.com/secretsmanager/latest/userguide/replication.html
Secrets Manager Rotation:https://docs.aws.amazon.com/secretsmanager/latest/userguide/rotating-secrets.html

Question: 14

A company runs an ecommerce application on Amazon EC2 instances behind an Application Load Balancer. The instances run in an Amazon EC2 Auto Scaling group across multiple Availability Zones. The Auto Scaling group scales based on CPU utilization metrics. The ecommerce application stores the transaction data in a MySQL 8.0 database that is hosted on a large EC2 instance.
The database's performance degrades quickly as application load increases. The application handles more read requests than write transactions. The company wants a solution that will automatically scale the database to meet the demand of unpredictable read workloads while maintaining high availability.
Which solution will meet these requirements?

A. Use Amazon Redshift with a single node for leader and compute functionality.

B. Use Amazon RDS with a Single-AZ deployment Configure Amazon RDS to add reader instances in a different Availability Zone.

C. Use Amazon Aurora with a Multi-AZ deployment. Configure Aurora Auto Scaling with Aurora Replicas.

D. Use Amazon ElastiCache for Memcached with EC2 Spot Instances.

Answer: C

Explanation:

The correct solution is to use Amazon Aurora with a Multi-AZ deployment and configure Aurora Auto Scaling with Aurora Replicas. Here's why:

Scalability: Aurora Auto Scaling automatically adds or removes Aurora Replicas in response to changes in application load. This addresses the need to scale the database to meet unpredictable read workloads.

Read-Heavy Workloads: Aurora Replicas are designed for read operations. By directing read traffic to these replicas, the primary database instance is relieved, improving overall performance.

High Availability: A Multi-AZ deployment ensures that the database remains available even if there is an infrastructure failure in one Availability Zone. Aurora automatically fails over to a replica in another Availability Zone.

Aurora's Compatibility: Aurora is compatible with MySQL and PostgreSQL, allowing for easier migration from the existing MySQL database.

Other options are suboptimal because:

A. Amazon Redshift is a data warehouse service optimized for analytical workloads, not transactional workloads.

B. Amazon RDS with a Single-AZ deployment does not guarantee high availability. Although reader instances can be added, a single-AZ configuration for the primary instance presents a single point of failure.

D. Amazon ElastiCache for Memcached is an in-memory caching service, not a database. It is not suitable for storing transactional data.

Supporting Links:

Amazon Aurora:https://aws.amazon.com/rds/aurora/
Aurora Auto Scaling:
https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Managing.Performance.html
Aurora Replicas:

https://docs.aws.amazon.com/AmazonRDS/latest/AuroraUserGuide/Aurora.Managing.ReadReplicas.html **Amazon RDS Multi-AZ:**https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Concepts.MultiAZ.html

Question: 15

A company recently migrated to AWS and wants to implement a solution to protect the traffic that flows in and out of the production VPC. The company had an inspection server in its on-premises data center. The inspection server performed specific operations such as traffic flow inspection and traffic filtering. The company wants to have the same functionalities in the AWS Cloud.
Which solution will meet these requirements?

A. Use Amazon GuardDuty for traffic inspection and traffic filtering in the production VPC.

B. Use Traffic Mirroring to mirror traffic from the production VPC for traffic inspection and filtering.

C. Use AWS Network Firewall to create the required rules for traffic inspection and traffic filtering for the production VPC.

D. Use AWS Firewall Manager to create the required rules for traffic inspection and traffic filtering for the production VPC.

Answer: C

Explanation:

The correct answer is C, using AWS Network Firewall. Here's a detailed justification:

The requirement is to replicate on-premises traffic inspection and filtering functionalities within AWS for a production VPC. Amazon GuardDuty (A) is a threat detection service that monitors for malicious activity and unauthorized behavior but doesn't provide inline traffic inspection and filtering based on custom rules. Traffic Mirroring (B) duplicates network traffic for analysis, but it doesn't provide filtering capabilities itself. You would need a separate inspection appliance to receive and process the mirrored traffic, adding complexity and cost. AWS Firewall Manager (D) centrally manages firewall rules across multiple accounts and VPCs; it doesn't inspect traffic directly or provide inspection capabilities. It works with AWS WAF, AWS Shield Advanced, and AWS Network Firewall.

AWS Network Firewall (C), on the other hand, is a managed network firewall service that provides essential protection for your VPCs. It allows you to define rules for inspecting and filtering network traffic based on criteria such as source and destination IP addresses, ports, and protocols. This precisely aligns with the company's need to implement traffic flow inspection and filtering, mimicking the capabilities of their on-premises inspection server. Network Firewall operates at the network layer (Layer 3 and Layer 4) and application layer (Layer 7), allowing for deep packet inspection (DPI) and fine-grained control. It also scales automatically to meet fluctuating traffic demands without requiring manual intervention.

Therefore, AWS Network Firewall is the best fit for the company's requirements because it provides a fully managed, scalable, and customizable solution for traffic inspection and filtering within the production VPC.

Authoritative links:

AWS Network Firewall:https://aws.amazon.com/network-firewall/
AWS Network Firewall Documentation:https://docs.aws.amazon.com/network-firewall/latest/developerguide/what-is-aws-network-firewall.html

Question: 16

A company hosts a data lake on AWS. The data lake consists of data in Amazon S3 and Amazon RDS for PostgreSQL. The company needs a reporting solution that provides data visualization and includes all the data sources within the data lake. Only the company's management team should have full access to all the

visualizations. The rest of the company should have only limited access. Which
solution will meet these requirements?

A. Create an analysis in Amazon QuickSight. Connect all the data sources and create new datasets. Publish dashboards to visualize the data. Share the dashboards with the appropriate IAM roles.

B. Create an analysis in Amazon QuickSight. Connect all the data sources and create new datasets. Publish dashboards to visualize the data. Share the dashboards with the appropriate users and groups.

C. Create an AWS Glue table and crawler for the data in Amazon S3. Create an AWS Glue extract, transform, and load (ETL) job to produce reports. Publish the reports to Amazon S3. Use S3 bucket policies to limit access to the reports.

D. Create an AWS Glue table and crawler for the data in Amazon S3. Use Amazon Athena Federated Query to access data within Amazon RDS for PostgreSQL. Generate reports by using Amazon Athena. Publish the reports to Amazon S3. Use S3 bucket policies to limit access to the reports.

Answer: B

Explanation:

The correct answer is B. Let's break down why:

Requirement 1: Data Visualization and Data Lake Integration: The company needs a data visualization tool that can ingest data from both S3 (data lake) and RDS for PostgreSQL. Amazon QuickSight excels at this. It can directly connect to both S3 (using manifests or direct paths) and RDS databases. Options C and D primarily focus on AWS Glue and Athena, which are more suited for data processing and querying, not direct visualization.

Requirement 2: Different Access Levels (Management vs. Rest of Company): QuickSight has a robust sharing mechanism. Dashboards can be shared with individual users or groups (created within QuickSight itself), allowing for granular control over who sees what. IAM roles (option A) are primarily used for controlling access to AWS resources, not for fine-grained access to dashboards within QuickSight. S3 bucket policies (options C and D) would only control access to the underlying report files, not the dynamic visualizations.

QuickSight Analysis, Datasets, and Dashboards: QuickSight works by creating an analysis from one or more datasets. The datasets are based on the connected data sources. The analysis is then used to build one or more dashboards. Publishing the dashboards makes them available to others.

Direct Connection to Data Sources: QuickSight's ability to directly connect to the data sources (S3 and RDS) eliminates the need for complex ETL processes just for reporting (which would be implied by option C). While ETL might be part of the broader data lake architecture, it's not the primary function for a reporting solution.

Athena Federated Query is overkill: Athena Federated Query (option D) allows querying across different data sources. Although possible, using QuickSight's direct connectors is a simpler and more suitable approach for this scenario.

In summary, option B is the most straightforward solution, using QuickSight's built-in capabilities to connect to data sources, create visualizations, and share dashboards with user-level and group-level access control.

Authoritative Links:

Amazon QuickSight:https://aws.amazon.com/quicksight/
Connecting to Data Sources in QuickSight:https://docs.aws.amazon.com/quicksight/latest/user/data-sources.html
Sharing QuickSight Dashboards:https://docs.aws.amazon.com/quicksight/latest/user/sharing-a-dashboard.html

Question: 17

A company is implementing a new business application. The application runs on two Amazon EC2 instances and uses an Amazon S3 bucket for document storage. A solutions architect needs to ensure that the EC2 instances can access the S3 bucket.
What should the solutions architect do to meet this requirement?

A. Create an IAM role that grants access to the S3 bucket. Attach the role to the EC2 instances.
B. Create an IAM policy that grants access to the S3 bucket. Attach the policy to the EC2 instances.
C. Create an IAM group that grants access to the S3 bucket. Attach the group to the EC2 instances.
D. Create an IAM user that grants access to the S3 bucket. Attach the user account to the EC2 instances.

Answer: A

Explanation:

The correct answer is A: Create an IAM role that grants access to the S3 bucket and attach the role to the EC2 instances. Here's why:

IAM Roles are the recommended approach for granting permissions to AWS services like EC2 to access other AWS resources such as S3. A role provides temporary security credentials to the EC2 instances, eliminating the need to store long-term credentials directly on the instances. This enhances security because the credentials are automatically rotated and managed by AWS.

IAM Policies define the permissions granted. While policies are fundamental, they are attached to IAM entities. Attaching an IAM policy directly to an EC2 instance is not a standard practice. Instead, the policy is attached to an IAM role, which is then associated with the EC2 instance. The instance assumes the role, gaining the permissions defined in the policy.

IAM Groups are collections of IAM users. While groups simplify permission management for users, they are not designed to be associated with EC2 instances for granting access to resources. Groups are a user-centric concept and not appropriate for service-to-service access control.

IAM Users represent individuals or applications that interact with AWS. Creating an IAM user and embedding its credentials on an EC2 instance is a security risk. These credentials would be long-term and static, making them vulnerable if the instance is compromised. Hardcoding or storing credentials on an EC2 instance is a bad practice. IAM Roles provide a more secure and manageable solution.

Therefore, the most secure and best practice approach is to create an IAM role with the necessary S3 access permissions and then attach that role to the EC2 instances. This allows the EC2 instances to assume the role and obtain temporary credentials to access the S3 bucket.

Further Reading:

IAM Roles for Amazon EC2
IAM Policies
Security best practices in IAM

Question: 18

An application development team is designing a microservice that will convert large images to smaller, compressed images. When a user uploads an image through the web interface, the microservice should store the image in an Amazon S3 bucket, process and compress the image with an AWS Lambda function, and store the image in its compressed form in a different S3 bucket.
A solutions architect needs to design a solution that uses durable, stateless components to process the images automatically.
Which combination of actions will meet these requirements? (Choose two.)

A. Create an Amazon Simple Queue Service (Amazon SQS) queue. Configure the S3 bucket to send a notification to the SQS queue when an image is uploaded to the S3 bucket.

B. Configure the Lambda function to use the Amazon Simple Queue Service (Amazon SQS) queue as the invocation source. When the SQS message is successfully processed, delete the message in the queue.

C. Configure the Lambda function to monitor the S3 bucket for new uploads. When an uploaded image is detected, write the file name to a text file in memory and use the text file to keep track of the images that were processed.

D. Launch an Amazon EC2 instance to monitor an Amazon Simple Queue Service (Amazon SQS) queue. When items are added to the queue, log the file name in a text file on the EC2 instance and invoke the Lambda function.

E. Configure an Amazon EventBridge (Amazon CloudWatch Events) event to monitor the S3 bucket. When an image is uploaded, send an alert to an Amazon ample Notification Service (Amazon SNS) topic with the application owner's email address for further processing.

Answer: AB

Explanation:

The chosen solution, using SQS and Lambda triggered by SQS, effectively addresses the requirements for durable, stateless image processing.

Option A is correct because SQS provides a durable queuing mechanism. When an image is uploaded to S3, a notification is sent to the SQS queue. This ensures that even if the Lambda function is temporarily unavailable, the image processing request will persist in the queue until it can be processed. SQS decouples the S3 upload event from the Lambda function invocation, enabling asynchronous processing. This aligns with the need for a stateless and reliable processing pipeline.
https://aws.amazon.com/sqs/

Option B is correct because configuring the Lambda function to consume messages from the SQS queue directly addresses the need for automated processing. Lambda's SQS trigger automatically polls the queue and invokes the function when a message arrives. This eliminates the need for custom polling logic. By deleting the message upon successful processing, the queue is kept clean and ensures that each image is processed only once.
https://docs.aws.amazon.com/lambda/latest/dg/with-sqs.html

Option C is incorrect because storing filenames in a text file in memory within the Lambda function introduces statefulness. Lambda functions should be stateless, and relying on in-memory storage for tracking processed images is unreliable, especially with concurrent invocations.

Option D is incorrect because launching an EC2 instance to monitor SQS adds unnecessary complexity and cost. Lambda functions can directly integrate with SQS without requiring an intermediary EC2 instance.

Option E is incorrect because using SNS for further processing is vague and doesn't directly address the requirement of automated image processing. SNS is a notification service, not an event-driven compute service. The application owner would need to manually trigger the image processing, defeating the automation requirement.

Question: 19

A company has a three-tier web application that is deployed on AWS. The web servers are deployed in a public subnet in a VPC. The application servers and database servers are deployed in private subnets in the same VPC. The company has deployed a third-party virtual firewall appliance from AWS Marketplace in an inspection VPC. The appliance is configured with an IP interface that can accept IP packets.
A solutions architect needs to integrate the web application with the appliance to inspect all traffic to the application before the traffic reaches the web server.
Which solution will meet these requirements with the LEAST operational overhead?

A. Create a Network Load Balancer in the public subnet of the application's VPC to route the traffic to the appliance for packet inspection.

B. Create an Application Load Balancer in the public subnet of the application's VPC to route the traffic to the appliance for packet inspection.

C. Deploy a transit gateway in the inspection VPConfigure route tables to route the incoming packets through the transit gateway.

D. Deploy a Gateway Load Balancer in the inspection VPC. Create a Gateway Load Balancer endpoint to receive the incoming packets and forward the packets to the appliance.

Answer: D

Explanation:

The correct answer is **D: Deploy a Gateway Load Balancer in the inspection VPC. Create a Gateway Load Balancer endpoint to receive the incoming packets and forward the packets to the appliance.**

Here's a detailed justification:

Gateway Load Balancer (GWLB): A GWLB is specifically designed for inline inspection of network traffic. It provides a single entry point for traffic, scales automatically, and distributes traffic to virtual appliances like firewalls. This is exactly what the scenario requires: inspecting traffic before it reaches the web servers. **GWLB Endpoint:** A GWLB endpoint allows you to seamlessly integrate the GWLB (in the inspection VPC) with the web application's VPC. Traffic destined for the web application's public subnet is redirected to the GWLB endpoint.

Traffic Flow: Incoming traffic from the internet flows to the GWLB endpoint in the application's VPC. The endpoint forwards the traffic to the GWLB in the inspection VPC. The GWLB distributes the traffic to the firewall appliance for inspection. After inspection, the appliance sends the traffic back through the GWLB, back through the endpoint, and finally to the web servers.

Least Operational Overhead: Compared to other options, the GWLB solution offers the least operational overhead. It's a managed service, meaning AWS handles scaling, availability, and patching. You don't have to manage routing tables as extensively as with Transit Gateway.

Here's why the other options are less suitable:

A. Network Load Balancer (NLB): NLBs are designed for load balancing TCP, UDP, and TLS traffic to backend targets. They are not suitable for inspecting packets at the application layer, or generally for integration with virtual firewalls. NLBs do not have native features to forward traffic for inspection and then back to the original destination.

B. Application Load Balancer (ALB): ALBs operate at the application layer (HTTP/HTTPS) and are primarily used for load balancing web traffic based on content. They are not intended for routing packets to virtual appliances for general-purpose packet inspection.

C. Transit Gateway (TGW): TGW is a network transit hub that connects VPCs and on-premises networks.

While you could use TGW to route traffic through an inspection VPC, it involves more complex route table configurations and management. It's overkill and adds unnecessary operational complexity compared to the GWLB, which is designed specifically for this purpose.

Authoritative Links:

Gateway Load Balancer:https://aws.amazon.com/elasticloadbalancing/gateway-load-balancer/
AWS Network Firewall Integration with GWLB:https://aws.amazon.com/blogs/networking-and-content-delivery/centralized-inspection-architecture-with-aws-network-firewall-gateway-load-balancer-and-route-manager/

Question: 20

A company wants to improve its ability to clone large amounts of production data into a test environment in the same AWS Region. The data is stored in Amazon EC2 instances on Amazon Elastic Block Store (Amazon EBS)

volumes. Modifications to the cloned data must not affect the production environment. The software that accesses this data requires consistently high I/O performance.
A solutions architect needs to minimize the time that is required to clone the production data into the test environment.
Which solution will meet these requirements?

A. Take EBS snapshots of the production EBS volumes. Restore the snapshots onto EC2 instance store volumes in the test environment.

B. Configure the production EBS volumes to use the EBS Multi-Attach feature. Take EBS snapshots of the production EBS volumes. Attach the production EBS volumes to the EC2 instances in the test environment.

C. Take EBS snapshots of the production EBS volumes. Create and initialize new EBS volumes. Attach the new EBS volumes to EC2 instances in the test environment before restoring the volumes from the production EBS snapshots.

D. Take EBS snapshots of the production EBS volumes. Turn on the EBS fast snapshot restore feature on the EBS snapshots. Restore the snapshots into new EBS volumes. Attach the new EBS volumes to EC2 instances in the test environment.

Answer: D

Explanation:

Here's a detailed justification for why option D is the correct answer, and why other options are incorrect:

Why Option D is Correct: "Take EBS snapshots of the production EBS volumes. Turn on the EBS fast snapshot restore feature on the EBS snapshots. Restore the snapshots into new EBS volumes. Attach the new EBS volumes to EC2 instances in the test environment."

This solution directly addresses the requirement of minimizing the cloning time while maintaining data isolation.

1. **EBS Snapshots for Data Duplication:** EBS snapshots are the standard and efficient way to create point-in-time copies of EBS volumes. This ensures a consistent and reliable clone of the production data. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/EBSSnapshots.html
2. **Fast Snapshot Restore (FSR) for Speed:** FSR significantly reduces the time required to restore EBS volumes from snapshots. Normally, restoring from a snapshot involves lazily loading the data blocks as they are accessed. FSR pre-initializes the volume in the background, so it's immediately ready for high I/O performance upon attachment. This satisfies the "minimize the time that is required to clone" requirement. https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ebs-fast-snapshot-restore.html
3. **New EBS Volumes for Isolation:** By restoring the snapshots into new EBS volumes, the test environment operates on independent copies of the data. Modifications in the test environment will not affect the production data, meeting the "Modifications to the cloned data must not affect the production environment" requirement.
4. **Attaching to EC2 Instances:** Finally, attaching the new EBS volumes to EC2 instances in the test environment makes the cloned data accessible to the applications.

Why Other Options Are Incorrect:

Option A: Restore to EC2 Instance Store: EC2 instance store volumes are ephemeral, meaning the data is lost when the instance is stopped or terminated. This makes them unsuitable for a persistent test environment where data needs to be retained. Further, instance store volumes aren't suitable for large datasets and generally have poorer durability.

Option B: EBS Multi-Attach and Attaching Production Volumes: EBS Multi-Attach allows you to attach a single EBS volume to multiple EC2 instances within the same Availability Zone. However, directly attaching production volumes to test instances creates a significant risk of data corruption in the production environment if the test environment makes unintended modifications. This also requires coordinating access and locking which is far from ideal. More over, the question states that modifications to cloned data should

NOT affect production environment. This choice violates it.

Option C: Creating Volumes and Restoring Without FSR: While this solution does provide data isolation by creating new EBS volumes, it does not address the requirement of minimizing the cloning time. Without FSR, restoring large EBS volumes from snapshots can take a significant amount of time, especially when high I/O performance is immediately required.

In summary, option D provides the best balance of data isolation, cloning speed, and performance for the test environment, leveraging EBS snapshots and FSR for efficient and safe data duplication.

Question: 21

An ecommerce company wants to launch a one-deal-a-day website on AWS. Each day will feature exactly one product on sale for a period of 24 hours. The company wants to be able to handle millions of requests each hour with millisecond latency during peak hours.
Which solution will meet these requirements with the LEAST operational overhead?

A. Use Amazon S3 to host the full website in different S3 buckets. Add Amazon CloudFront distributions. Set the S3 buckets as origins for the distributions. Store the order data in Amazon S3.

B. Deploy the full website on Amazon EC2 instances that run in Auto Scaling groups across multiple Availability Zones. Add an Application Load Balancer (ALB) to distribute the website traffic. Add another ALB for the backend APIs. Store the data in Amazon RDS for MySQL.

C. Migrate the full application to run in containers. Host the containers on Amazon Elastic Kubernetes Service (Amazon EKS). Use the Kubernetes Cluster Autoscaler to increase and decrease the number of pods to process bursts in traffic. Store the data in Amazon RDS for MySQL.

D. Use an Amazon S3 bucket to host the website's static content. Deploy an Amazon CloudFront distribution. Set the S3 bucket as the origin. Use Amazon API Gateway and AWS Lambda functions for the backend APIs. Store the data in Amazon DynamoDB.

Answer: D

Explanation:

Option D is the most suitable solution because it leverages serverless technologies and static content hosting for optimal scalability, performance, and minimal operational overhead. Hosting the static website content (HTML, CSS, JavaScript, images) on Amazon S3 and distributing it via Amazon CloudFront provides global content delivery with low latency and high availability. CloudFront's caching capabilities further reduce load on the origin S3 bucket.

Using Amazon API Gateway and AWS Lambda for backend APIs enables on-demand execution of code without managing servers. API Gateway handles request routing, authorization, and throttling, while Lambda functions execute the business logic. This serverless approach scales automatically with traffic, ensuring millisecond latency during peak hours.

Storing order data in Amazon DynamoDB is ideal because DynamoDB is a fully managed NoSQL database service that provides consistent, single-digit millisecond latency at any scale. DynamoDB's serverless nature eliminates the need for database administration tasks.

Options A, B, and C involve managing infrastructure (EC2 instances, containers, databases), which increases operational overhead. Option A's approach of storing order data directly in S3 isn't suitable for transactional data. Option B utilizes ALBs and RDS for MySQL, requiring instance management and database administration. Option C introduces the complexity of container orchestration with EKS. Therefore, Option D presents the least operational overhead while meeting the performance requirements.

Here are some authoritative links for further research:

Amazon S3:https://aws.amazon.com/s3/

Amazon CloudFront:https://aws.amazon.com/cloudfront/
Amazon API Gateway:https://aws.amazon.com/api-gateway/
AWS Lambda:https://aws.amazon.com/lambda/
Amazon DynamoDB:https://aws.amazon.com/dynamodb/

Question: 22

A solutions architect is using Amazon S3 to design the storage architecture of a new digital media application. The media files must be resilient to the loss of an Availability Zone. Some files are accessed frequently while other files are rarely accessed in an unpredictable pattern. The solutions architect must minimize the costs of storing and retrieving the media files.
Which storage option meets these requirements?

A. S3 Standard
B. S3 Intelligent-Tiering
C. S3 Standard-Infrequent Access (S3 Standard-IA)
D. S3 One Zone-Infrequent Access (S3 One Zone-IA)

Answer: B

Explanation:

The correct answer is **B. S3 Intelligent-Tiering**.

Here's why:

Resilience to Availability Zone Loss: The application requires resilience to the loss of an Availability Zone. This immediately rules out S3 One Zone-IA (Option D) because it stores data in a single Availability Zone. S3 Standard, S3 Intelligent-Tiering, and S3 Standard-IA all replicate data across multiple Availability Zones, providing the necessary resilience.

Frequently and Infrequently Accessed Files: The application has both frequently and infrequently accessed files with an unpredictable access pattern. S3 Standard-IA (Option C) is suitable for data accessed less frequently, but using it for frequently accessed data would be unnecessarily expensive. S3 Standard (Option A) would be suitable for all objects but is the most expensive option for objects that are infrequent to access.

Cost Minimization: S3 Intelligent-Tiering automatically moves data between frequent and infrequent access tiers based on access patterns, without any operational overhead or retrieval fees. It delivers automatic cost savings when access patterns change. It monitors access patterns and moves objects that have not been accessed for 30 consecutive days to the infrequent access tier and then after another 60 days of inactivity to the archive access tier. This ensures the optimal storage cost for both frequently and infrequently accessed files.

Benefits of Intelligent-Tiering: S3 Intelligent-Tiering optimizes storage costs by automatically moving data to the most cost-effective access tier without performance impact or operational overhead. This approach is ideal when access patterns are unpredictable or change over time.

In summary, S3 Intelligent-Tiering balances the requirements of Availability Zone resilience, handling both frequently and infrequently accessed files, and minimizing storage costs, making it the most suitable storage option.

Relevant Links:

Amazon S3 Storage Classes:
S3 Intelligent-Tiering:

Question: 23

A company is storing backup files by using Amazon S3 Standard storage. The files are accessed frequently for 1 month. However, the files are not accessed after 1 month. The company must keep the files indefinitely. Which storage solution will meet these requirements MOST cost-effectively?

A. Configure S3 Intelligent-Tiering to automatically migrate objects.

B. Create an S3 Lifecycle configuration to transition objects from S3 Standard to S3 Glacier Deep Archive after 1 month.

C. Create an S3 Lifecycle configuration to transition objects from S3 Standard to S3 Standard-Infrequent Access (S3 Standard-IA) after 1 month.

D. Create an S3 Lifecycle configuration to transition objects from S3 Standard to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 1 month.

Answer: B

Explanation:

The question requires the most cost-effective storage solution for backup files frequently accessed for one month and then indefinitely archived with infrequent access.

Option B, creating an S3 Lifecycle configuration to transition objects from S3 Standard to S3 Glacier Deep Archive after 1 month, is the most cost-effective solution. S3 Glacier Deep Archive offers the lowest storage cost among all S3 storage classes, making it ideal for long-term archival. The lifecycle policy automates the transition, eliminating manual intervention. S3 Standard is used for the first month of frequent access. This approach balances initial accessibility with long-term cost optimization for rarely accessed data.

Option A is less suitable. S3 Intelligent-Tiering automatically moves data between frequent, infrequent, and archive access tiers based on usage patterns. While convenient, the cost of monitoring and frequent tier transitions might exceed the cost-effectiveness of directly moving to Glacier Deep Archive after the initial active period. The assumption is that the files are never accessed after the initial month, making Intelligent-Tiering an unnecessary expense.

Option C involves transitioning to S3 Standard-IA, which is designed for data accessed less frequently but still requires rapid retrieval. Given that the backup files will likely not be accessed after the first month, using Standard-IA is more expensive than Glacier Deep Archive because Standard-IA has higher storage costs.

Option D uses S3 One Zone-IA, which is even less suitable than Standard-IA, although cheaper than Standard-IA. It stores data in a single Availability Zone, making it cheaper, but it's inherently less durable than Standard-IA, Standard or Glacier. Because these are backup files the durability provided by the storage is of utmost importance, and a loss of data due to a single availability zone failure is unacceptable. Moreover, it's more expensive than Glacier Deep Archive.

Therefore, transitioning to Glacier Deep Archive through a Lifecycle policy offers the best combination of cost-effectiveness and long-term archival needs when data will not be accessed for the foreseeable future after the first month.

Relevant documentation:

S3 Glacier Deep Archive: Describes the features and cost benefits of S3 Glacier Deep Archive. S3 Lifecycle Management: Explains how to automate transitions between storage classes.

Question: 24

A company observes an increase in Amazon EC2 costs in its most recent bill. The billing team notices unwanted

vertical scaling of instance types for a couple of EC2 instances. A solutions architect needs to create a graph comparing the last 2 months of EC2 costs and perform an in-depth analysis to identify the root cause of the vertical scaling.
How should the solutions architect generate the information with the LEAST operational overhead?

A. Use AWS Budgets to create a budget report and compare EC2 costs based on instance types.

B. Use Cost Explorer's granular filtering feature to perform an in-depth analysis of EC2 costs based on instance types.

C. Use graphs from the AWS Billing and Cost Management dashboard to compare EC2 costs based on instance types for the last 2 months.

D. Use AWS Cost and Usage Reports to create a report and send it to an Amazon S3 bucket. Use Amazon QuickSight with Amazon S3 as a source to generate an interactive graph based on instance types.

Answer: B

Explanation:

The correct answer is **B. Use Cost Explorer's granular filtering feature to perform an in-depth analysis of EC2 costs based on instance types.**

Here's why:

Cost Explorer is designed specifically for cost analysis and provides a user-friendly interface for exploring AWS costs. It allows for granular filtering by service (EC2), usage type (instance type), and time range, enabling the solutions architect to easily compare EC2 costs for the last two months and identify the instances that have experienced unwanted vertical scaling. Cost Explorer's built-in charting capabilities make it straightforward to visualize these cost trends without requiring additional tools or setup. The "group by" feature within Cost Explorer lets users easily aggregate costs by instance type. This allows for direct identification of cost increases associated with specific instance types, streamlining the analysis. The goal is to analyze past costs, and Cost Explorer is built to efficiently do just that.

Option A, AWS Budgets, is primarily for setting cost thresholds and receiving alerts when those thresholds are exceeded. While it can provide cost information, it's not designed for in-depth historical analysis and doesn't offer the granular filtering capabilities needed to pinpoint specific instance type cost increases as readily as Cost Explorer.

Option C, the AWS Billing and Cost Management dashboard, provides a high-level overview of costs but lacks the granular filtering and analysis features available in Cost Explorer. It would require more manual effort to isolate and analyze the cost increase associated with specific EC2 instance types.

Option D, AWS Cost and Usage Reports (CUR) with Amazon QuickSight, is powerful but involves significantly more operational overhead. It requires configuring and delivering the CUR to an S3 bucket, then setting up QuickSight with S3 as a data source, and finally creating the interactive graph. This process is much more complex and time-consuming compared to using Cost Explorer's built-in features. While CUR provides comprehensive data, for a focused analysis as described in the question, Cost Explorer is far more efficient.

Therefore, Cost Explorer provides the least operational overhead and is the most efficient way to achieve the required in-depth analysis of EC2 costs based on instance types.

Here's an authoritative link for further research:

AWS Cost Explorer

Question: 25

A company is designing an application. The application uses an AWS Lambda function to receive information through Amazon API Gateway and to store the information in an Amazon Aurora PostgreSQL database.

During the proof-of-concept stage, the company has to increase the Lambda quotas significantly to handle the high volumes of data that the company needs to load into the database. A solutions architect must recommend a new design to improve scalability and minimize the configuration effort.
Which solution will meet these requirements?

A. Refactor the Lambda function code to Apache Tomcat code that runs on Amazon EC2 instances. Connect the database by using native Java Database Connectivity (JDBC) drivers.

B. Change the platform from Aurora to Amazon DynamoDProvision a DynamoDB Accelerator (DAX) cluster. Use the DAX client SDK to point the existing DynamoDB API calls at the DAX cluster.

C. Set up two Lambda functions. Configure one function to receive the information. Configure the other function to load the information into the database. Integrate the Lambda functions by using Amazon Simple Notification Service (Amazon SNS).

D. Set up two Lambda functions. Configure one function to receive the information. Configure the other function to load the information into the database. Integrate the Lambda functions by using an Amazon Simple Queue Service (Amazon SQS) queue.

Answer: D

Explanation:

The best solution to improve scalability and minimize configuration effort when dealing with high volumes of data between a Lambda function receiving data and an Aurora PostgreSQL database is to use an Amazon SQS queue to decouple the two functions.

Here's why:

Decoupling with SQS: SQS acts as a buffer between the data receiving Lambda function and the database loading Lambda function. The first Lambda function places messages containing the information into the SQS queue. The second Lambda function retrieves messages from the queue and loads the data into Aurora. This decoupling allows each function to scale independently based on its specific needs.

Scalability: SQS provides virtually unlimited scalability. It can handle large volumes of messages, allowing the system to absorb bursts of data without overwhelming the database. The database loading Lambda function can then process the messages at a rate the database can handle, preventing overload.

Minimized Configuration Effort: SQS is a managed service, which reduces the operational burden. Setting up the queue and integrating it with the Lambda functions requires minimal configuration.

Resilience: SQS provides message durability. If the database loading Lambda function fails, the messages remain in the queue until they are successfully processed, ensuring data is not lost. This contrasts with SNS, which is better suited for fan-out scenarios and doesn't guarantee message delivery if no subscribers are available.

Why other options are less suitable:

A: Refactoring to EC2 and Tomcat increases operational complexity. EC2 requires managing servers, including patching, scaling, and ensuring high availability. This defeats the goal of minimizing configuration effort.

B: Changing to DynamoDB with DAX is a significant architectural change and involves substantial code modification. DAX is for DynamoDB, not Aurora PostgreSQL.

C: Using SNS doesn't provide the queuing and buffering capabilities of SQS. If the database loading Lambda function is unavailable, messages sent via SNS might be lost, potentially impacting data consistency. SNS is not suitable for reliable, asynchronous processing.

In summary, using SQS for decoupling provides the best combination of scalability, minimized configuration, and reliability for this use case.

Relevant Documentation:

Amazon SQS:https://aws.amazon.com/sqs/
AWS Lambda:https://aws.amazon.com/lambda/
Amazon Aurora:https://aws.amazon.com/rds/aurora/

Question: 26

A company needs to review its AWS Cloud deployment to ensure that its Amazon S3 buckets do not have unauthorized configuration changes.
What should a solutions architect do to accomplish this goal?

A. Turn on AWS Config with the appropriate rules.
B. Turn on AWS Trusted Advisor with the appropriate checks.
C. Turn on Amazon Inspector with the appropriate assessment template.
D. Turn on Amazon S3 server access logging. Configure Amazon EventBridge (Amazon Cloud Watch Events).

Answer: A

Explanation:

The correct answer is A, turning on AWS Config with the appropriate rules. Here's why:

AWS Config is a service that enables you to assess, audit, and evaluate the configurations of your AWS resources. It continuously monitors and records your AWS resource configurations, allowing you to automate the evaluation of recorded configurations against desired configurations. In this scenario, the goal is to ensure S3 bucket configurations remain authorized, which directly aligns with Config's capabilities. You can define AWS Config rules that specify the desired state of your S3 buckets (e.g., encryption enabled, public access blocked). If a bucket drifts from this desired state, Config will flag it as non-compliant. This makes it perfect for detecting unauthorized configuration changes.

Option B, AWS Trusted Advisor, primarily provides recommendations on cost optimization, performance, security, fault tolerance, and service limits. While it offers security checks, it is more high-level and may not provide the detailed configuration change detection needed for S3 buckets specifically.

Option C, Amazon Inspector, focuses on security vulnerabilities and deviations from security best practices within your EC2 instances and container images. It's not designed to monitor and audit configuration changes to S3 buckets.

Option D, turning on Amazon S3 server access logging and configuring Amazon EventBridge (CloudWatch Events), would provide logs of who accessed the bucket. EventBridge could trigger alerts on specific events.
However, it requires significant custom parsing and rule building to determine if the configuration of the bucket itself has changed (e.g., a change to bucket policy or encryption). This is a much more complex and less direct method than using AWS Config. Config provides managed rules specifically designed for configuration compliance.

In summary, AWS Config is the most suitable service because it's designed to continuously monitor and evaluate the configurations of AWS resources, allowing you to proactively identify and remediate unauthorized configuration changes in S3 buckets.

Authoritative Links:

AWS Config:https://aws.amazon.com/config/
AWS Config Rules:https://docs.aws.amazon.com/config/latest/developerguide/evaluate-config.html

Question: 27

A company is launching a new application and will display application metrics on an Amazon CloudWatch dashboard. The company's product manager needs to access this dashboard periodically. The product manager does not have an AWS account. A solutions architect must provide access to the product manager by following the principle of least privilege. Which solution will meet these requirements?

A. Share the dashboard from the CloudWatch console. Enter the product manager's email address, and complete the sharing steps. Provide a shareable link for the dashboard to the product manager.

B. Create an IAM user specifically for the product manager. Attach the CloudWatchReadOnlyAccess AWS managed policy to the user. Share the new login credentials with the product manager. Share the browser URL of the correct dashboard with the product manager.

C. Create an IAM user for the company's employees. Attach the ViewOnlyAccess AWS managed policy to the IAM user. Share the new login credentials with the product manager. Ask the product manager to navigate to the CloudWatch console and locate the dashboard by name in the Dashboards section.

D. Deploy a bastion server in a public subnet. When the product manager requires access to the dashboard, start the server and share the RDP credentials. On the bastion server, ensure that the browser is configured to open the dashboard URL with cached AWS credentials that have appropriate permissions to view the dashboard.

Answer: A

Explanation:

The best solution is A because it leverages CloudWatch's built-in dashboard sharing feature, providing the most direct and least privileged access for the product manager without requiring an AWS account. CloudWatch's sharing feature allows you to create a shareable link to the dashboard that can be accessed without AWS credentials. This satisfies the requirement of allowing the product manager to view the dashboard.

Option B is less ideal. While it technically grants the product manager access, creating an IAM user solely for dashboard viewing is an over-provisioning of access. It also necessitates managing another AWS user and distributing credentials, increasing administrative overhead and security risks. The product manager would gain access to the entire AWS console with read-only access to CloudWatch, exceeding the least privilege principle.

Option C is also unsuitable. The ViewOnlyAccess AWS managed policy is overly broad and doesn't specifically target CloudWatch. Sharing credentials among employees introduces significant security risks and hinders accountability. Sharing an IAM user's credentials violates AWS best practices and security guidelines.

Option D is the least efficient and most complex option. Deploying a bastion server just to view a CloudWatch dashboard is excessive and unnecessary. It involves managing infrastructure, configuring security groups, and sharing RDP credentials, adding significant operational overhead. It's also not the least privileged approach.

Therefore, option A directly addresses the requirement by using CloudWatch's built-in feature, adhering to the principle of least privilege, and minimizing operational overhead.

Relevant Documentation:

Sharing CloudWatch dashboards

Question: 28

A company is migrating applications to AWS. The applications are deployed in different accounts. The company manages the accounts centrally by using AWS Organizations. The company's security team needs a single sign-on (SSO) solution across all the company's accounts. The company must continue managing the users and groups in

its on-premises self-managed Microsoft Active Directory.
Which solution will meet these requirements?

A. Enable AWS Single Sign-On (AWS SSO) from the AWS SSO console. Create a one-way forest trust or a one-way domain trust to connect the company's self-managed Microsoft Active Directory with AWS SSO by using AWS Directory Service for Microsoft Active Directory.

B. Enable AWS Single Sign-On (AWS SSO) from the AWS SSO console. Create a two-way forest trust to connect the company's self-managed Microsoft Active Directory with AWS SSO by using AWS Directory Service for Microsoft Active Directory.

C. Use AWS Directory Service. Create a two-way trust relationship with the company's self-managed Microsoft Active Directory.

D. Deploy an identity provider (IdP) on premises. Enable AWS Single Sign-On (AWS SSO) from the AWS SSO console.

Answer: B

Explanation:

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

The requirement is to integrate an existing on-premises Microsoft Active Directory (AD) with AWS SSO to provide centralized single sign-on across multiple AWS accounts managed by AWS Organizations. To achieve this while maintaining user and group management within the on-premises AD, a trust relationship must be established between the on-premises AD and AWS.

AWS SSO doesn't directly integrate with self-managed AD. Instead, AWS Directory Service for Microsoft Active Directory (AWS Managed Microsoft AD) acts as an intermediary. AWS Managed Microsoft AD is a fully managed service allowing you to run actual Active Directory in AWS. To link your on-premises AD, you need to establish a trust relationship.

A **two-way forest trust** is the appropriate type of trust when you need users in both the on-premises AD forest and the AWS Managed Microsoft AD forest to authenticate to resources in the other forest. In this scenario, on-premises users need to access AWS resources, and potentially, AWS-based users (if you later create them) might need to access on-premises resources. Therefore, a two-way trust ensures seamless
authentication in both directions. AWS SSO leverages this trust relationship via AWS Managed Microsoft AD to authenticate users from the on-premises AD. The AWS SSO console is used to enable and configure SSO across the AWS Organization.

Option A is incorrect because a one-way trust is insufficient. A one-way trust only allows authentication in one direction, failing to meet the requirement where AWS SSO needs to authenticate users against the on-premises AD.

Option C is incorrect because while creating a two-way trust relationship using AWS Directory Service is necessary, it doesn't by itself enable SSO. AWS SSO is the service required to centralize and manage single sign-on to multiple AWS accounts.

Option D is incorrect because deploying an on-premises IdP adds unnecessary complexity. AWS SSO is designed to integrate with existing identity providers, especially Microsoft Active Directory through AWS Directory Service.
Using a third-party IdP on-premises duplicates functionality that AWS SSO already provides and increases the management overhead. Moreover, AWS SSO has direct integration capabilities with AWS Managed Microsoft AD, and no further IdP on-premises is needed.

In summary, creating a two-way forest trust between the on-premises Active Directory and AWS Directory Service for Microsoft Active Directory, then enabling and configuring AWS SSO, allows the organization to centralize SSO while maintaining user management in the on-premises environment.

Relevant Links:

AWS SSO:https://aws.amazon.com/single-sign-on/
AWS Directory Service:https://aws.amazon.com/directoryservice/
How AWS SSO works with Active Directory:https://docs.aws.amazon.com/singlesignon/latest/userguide/ad-connector.html
Trust Relationships:https://learn.microsoft.com/en-us/windows-server/identity/ad-ds/plan/how-forest-trusts-work

Question: 29

A company provides a Voice over Internet Protocol (VoIP) service that uses UDP connections. The service consists of Amazon EC2 instances that run in an Auto Scaling group. The company has deployments across multiple AWS Regions. The company needs to route users to the Region with the lowest latency. The company also needs automated failover between Regions.
Which solution will meet these requirements?

A. Deploy a Network Load Balancer (NLB) and an associated target group. Associate the target group with the Auto Scaling group. Use the NLB as an AWS Global Accelerator endpoint in each Region.

B. Deploy an Application Load Balancer (ALB) and an associated target group. Associate the target group with the Auto Scaling group. Use the ALB as an AWS Global Accelerator endpoint in each Region.

C. Deploy a Network Load Balancer (NLB) and an associated target group. Associate the target group with the Auto Scaling group. Create an Amazon Route 53 latency record that points to aliases for each NLB. Create an Amazon CloudFront distribution that uses the latency record as an origin.

D. Deploy an Application Load Balancer (ALB) and an associated target group. Associate the target group with the Auto Scaling group. Create an Amazon Route 53 weighted record that points to aliases for each ALB. Deploy an Amazon CloudFront distribution that uses the weighted record as an origin.

Answer: A

Explanation:

The correct answer is A. Here's why:

UDP Requirement: The VoIP service relies on UDP connections. Application Load Balancers (ALB) only support HTTP and HTTPS protocols (TCP). Network Load Balancers (NLB) are designed to handle TCP, UDP, and TLS traffic. Therefore, NLB is the appropriate choice for this scenario.

Lowest Latency Routing and Automated Failover: AWS Global Accelerator leverages the AWS global network to route traffic to the optimal endpoint (closest Region) based on user location and network conditions, minimizing latency. It also provides automatic failover; if a Region becomes unavailable, Global Accelerator will automatically redirect traffic to a healthy Region.

Integration with Auto Scaling Groups: Both NLB and ALB can be integrated with Auto Scaling groups. By associating the target group with the Auto Scaling group, the load balancer automatically registers and deregisters instances as they are launched or terminated by the Auto Scaling group.

Route 53 Alternatives: While Route 53 latency-based routing can provide similar functionality, Global Accelerator offers performance advantages due to its use of the AWS global network and intelligent traffic routing. CloudFront is typically used for caching static content, not for routing UDP-based traffic based on latency.

Why other options are incorrect:

Option B: ALB is unsuitable because it does not support UDP.

Option C: Route 53 latency records do not provide the same level of optimized routing as Global Accelerator. Using CloudFront as a distribution for this purpose is unnecessary and not the best practice.

Option D: ALB is unsuitable because it does not support UDP. Route 53 weighted records are used to distribute traffic based on predefined weights, not real-time latency measurements.

Supporting Links:

AWS Global Accelerator: https://aws.amazon.com/global-accelerator/
Network Load Balancer: https://aws.amazon.com/elasticloadbalancing/network-load-balancer/ Application Load Balancer: https://aws.amazon.com/elasticloadbalancing/application-load-balancer/ Amazon Route 53: https://aws.amazon.com/route53/

Question: 30

A development team runs monthly resource-intensive tests on its general purpose Amazon RDS for MySQL DB instance with Performance Insights enabled. The testing lasts for 48 hours once a month and is the only process that uses the database. The team wants to reduce the cost of running the tests without reducing the compute and memory attributes of the DB instance.
Which solution meets these requirements MOST cost-effectively?

A. Stop the DB instance when tests are completed. Restart the DB instance when required.

B. Use an Auto Scaling policy with the DB instance to automatically scale when tests are completed.

C. Create a snapshot when tests are completed. Terminate the DB instance and restore the snapshot when required.

D. Modify the DB instance to a low-capacity instance when tests are completed. Modify the DB instance again when required.

Answer: C

Explanation:

The most cost-effective solution is **C. Create a snapshot when tests are completed. Terminate the DB instance and restore the snapshot when required.**

Here's why:

Cost Optimization: The primary goal is cost reduction. Terminating the DB instance when it's not needed completely eliminates the compute costs associated with running the instance. AWS charges for RDS instances per hour/day they are running.

Preserving Data: Snapshots preserve the database state at the end of the tests. This ensures no data loss, and the team can easily restore the database to its previous state.

Compute & Memory Retention: Restoring from a snapshot creates a new DB instance with the same specifications (compute, memory) as the original. This preserves the performance characteristics required during the test.

Alternative A (Stop/Start): Stopping the instance saves on compute costs, but you still incur costs for storage (database volume) which won't be as significant as the cost of a running instance, but there is a cost. Stopping and starting instances can take a significant amount of time as well.

Alternative B (Auto Scaling): Auto Scaling doesn't apply to RDS instance types. Auto Scaling is for automatically scaling the number of EC2 instances in your application, not for vertically scaling an RDS instance.

Alternative D (Modify Instance): Modifying the instance size involves downtime and does not completely eliminate costs. While it reduces costs, it doesn't achieve the same level of savings as terminating the instance.

Performance Insights: Performance Insights data is associated with the DB instance. Terminating and recreating the instance will lead to losing data that the testing team may want to review later on. The team can export the insights before terminating the instance.

Restore Time: While restoring from a snapshot does take time, it's a reasonable trade-off for significant cost savings, given the monthly testing frequency.

Therefore, creating a snapshot, terminating the instance, and restoring when needed provides the optimal balance between cost savings, data preservation, and performance retention.

Supporting Links:

Amazon RDS Pricing:https://aws.amazon.com/rds/pricing/
Creating a DB Snapshot:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_CreateSnapshot.html **Restoring from a DB Snapshot:**
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_RestoreFromSnapshot.html

Question: 31

A company that hosts its web application on AWS wants to ensure all Amazon EC2 instances. Amazon RDS DB instances. and Amazon Redshift clusters are configured with tags. The company wants to minimize the effort of configuring and operating this check.
What should a solutions architect do to accomplish this?

A. Use AWS Config rules to define and detect resources that are not properly tagged.

B. Use Cost Explorer to display resources that are not properly tagged. Tag those resources manually.

C. Write API calls to check all resources for proper tag allocation. Periodically run the code on an EC2 instance.

D. Write API calls to check all resources for proper tag allocation. Schedule an AWS Lambda function through Amazon CloudWatch to periodically run the code.

Answer: A

Explanation:

The correct answer is A because AWS Config provides a managed service to assess, audit, and evaluate the configurations of your AWS resources. With AWS Config, you can define rules that check if your resources are compliant with desired configurations, including the presence of specific tags. When a resource is found to be non-compliant (missing tags in this case), AWS Config can flag it, allowing for easy identification and remediation. This minimizes the operational burden since AWS Config continuously monitors resource configurations without requiring custom code or manual checks.

Option B is incorrect because Cost Explorer is primarily used for cost management and visualization and does not provide a built-in mechanism for identifying resources without proper tags. While Cost Explorer can utilize tags for cost allocation, it won't proactively alert you to missing tags or enforce their existence. Tagging resources manually is a tedious and error-prone approach for an environment where continuous monitoring is required.

Options C and D involve writing custom API calls and managing infrastructure to execute them, either on an EC2 instance or through Lambda. While technically feasible, these approaches introduce operational overhead associated with code development, deployment, maintenance, and scaling. AWS Config offers a managed and declarative way to achieve the same goal, thus reducing complexity and operational effort. Lambda adds unnecessary complexity for this task as AWS Config Rules can be executed and maintained without code.

In summary, AWS Config's pre-built rules, continuous monitoring capabilities, and managed service nature make it the optimal solution for ensuring resources are properly tagged while minimizing effort and operational overhead.

Refer to the following AWS documentation for further understanding:

AWS Config:https://aws.amazon.com/config/
AWS Config Rules:https://docs.aws.amazon.com/config/latest/developerguide/evaluate-config.html

Question: 32

A development team needs to host a website that will be accessed by other teams. The website contents consist of HTML, CSS, client-side JavaScript, and images.
Which method is the MOST cost-effective for hosting the website?

A. Containerize the website and host it in AWS Fargate.
B. Create an Amazon S3 bucket and host the website there.
C. Deploy a web server on an Amazon EC2 instance to host the website.
D. Configure an Application Load Balancer with an AWS Lambda target that uses the Express.js framework.

Answer: B

Explanation:

The most cost-effective solution for hosting a static website composed of HTML, CSS, JavaScript, and images is to use Amazon S3. S3 offers low storage costs and efficient content delivery through its integration with Amazon CloudFront.

Option A, containerizing the website and hosting it on AWS Fargate, involves running a container orchestration service, which is overkill for static content. Fargate is designed for applications that require compute resources, introducing unnecessary complexity and cost.

Option C, deploying a web server on an Amazon EC2 instance, incurs costs for the EC2 instance itself, including compute, storage, and potentially bandwidth. Maintaining and managing the EC2 instance also adds operational overhead. While EC2 provides flexibility, it's less cost-effective than S3 for static content.

Option D, configuring an Application Load Balancer with an AWS Lambda target using Express.js, is unnecessarily complex and expensive. This setup is suitable for dynamic content generation and serverless applications. Using Lambda for serving static files is inefficient and introduces additional latency. An Application Load Balancer is not necessary for static content delivery.

S3's static website hosting feature allows you to serve the files directly from an S3 bucket, eliminating the need for a web server. Furthermore, you can leverage CloudFront to cache the website content globally, reducing latency and further decreasing costs. S3 buckets also provide scalability and high availability by default. S3's pay-as-you-go pricing model makes it a very cost-effective option for this use case.

Therefore, the low cost, inherent scalability, and ease of use of Amazon S3 make it the most cost-effective solution for hosting a simple static website.

Relevant links:

Amazon S3 Static Website Hosting
Amazon CloudFront

Question: 33

A company runs an online marketplace web application on AWS. The application serves hundreds of thousands of users during peak hours. The company needs a scalable, near-real-time solution to share the details of millions of financial transactions with several other internal applications. Transactions also need to be processed to remove sensitive data before being stored in a document database for low-latency retrieval.
What should a solutions architect recommend to meet these requirements?

A. Store the transactions data into Amazon DynamoDB. Set up a rule in DynamoDB to remove sensitive data from every transaction upon write. Use DynamoDB Streams to share the transactions data with other applications.

B. Stream the transactions data into Amazon Kinesis Data Firehose to store data in Amazon DynamoDB and Amazon S3. Use AWS Lambda integration with Kinesis Data Firehose to remove sensitive data. Other applications can consume the data stored in Amazon S3.

C. Stream the transactions data into Amazon Kinesis Data Streams. Use AWS Lambda integration to remove sensitive data from every transaction and then store the transactions data in Amazon DynamoDB. Other applications can consume the transactions data off the Kinesis data stream.

D. Store the batched transactions data in Amazon S3 as files. Use AWS Lambda to process every file and remove sensitive data before updating the files in Amazon S3. The Lambda function then stores the data in Amazon DynamoDB. Other applications can consume transaction files stored in Amazon S3.

Answer: C

Explanation:

Here's a detailed justification for why option C is the best solution and why the others are less suitable:

Option C: Stream the transactions data into Amazon Kinesis Data Streams. Use AWS Lambda integration to remove sensitive data from every transaction and then store the transactions data in Amazon DynamoDB. Other applications can consume the transactions data off the Kinesis data stream.

This is the optimal choice because it leverages a combination of AWS services that are specifically designed for real-time data processing, transformation, and sharing.

1. **Real-time Ingestion:** Amazon Kinesis Data Streams is designed for ingesting and processing high-volume, real-time data streams. It is perfectly suited for capturing millions of financial transactions.
2. **Data Transformation:** The Lambda function provides a serverless and scalable way to remove sensitive data from each transaction as it flows through the Kinesis stream. This ensures compliance and data security before storage and sharing.
3. **Low-Latency Storage:** DynamoDB is a NoSQL database known for its low-latency read and write capabilities, making it ideal for storing the processed transactions and enabling fast retrieval.
4. **Data Sharing:** Other applications can subscribe to the Kinesis Data Stream and receive the transformed transactions data in near real-time. This provides a decoupled and scalable mechanism for sharing data with other internal applications.

Why other options are less optimal:

Option A: DynamoDB Streams are primarily intended for auditing or replicating data changes within DynamoDB itself, and not optimized for sharing data with a multitude of other applications in near real-time. Removing sensitive data using DynamoDB rules would happen after the transaction is already stored, possibly violating compliance requirements.

Option B: Kinesis Data Firehose is designed for loading data into data lakes or analytics services, and doesn't readily support direct consumption of data by other real-time applications. Using S3 as a central point can create latency.

Option D: Batching and storing in S3, processing with Lambda, and then storing in DynamoDB introduces significant latency. This approach isn't near real-time. Lambda processing S3 objects is more suited for larger files and infrequent updates, unlike this requirement.

In Conclusion: Option C provides a scalable, near-real-time, secure, and decoupled solution for processing financial transactions and sharing them with other applications, aligning perfectly with the requirements.

Authoritative Links:

Amazon Kinesis Data Streams:https://aws.amazon.com/kinesis/data-streams/
AWS Lambda:https://aws.amazon.com/lambda/

Amazon DynamoDB:https://aws.amazon.com/dynamodb/

Question: 34

A company hosts its multi-tier applications on AWS. For compliance, governance, auditing, and security, the company must track configuration changes on its AWS resources and record a history of API calls made to these resources.
What should a solutions architect do to meet these requirements?

A. Use AWS CloudTrail to track configuration changes and AWS Config to record API calls.
B. Use AWS Config to track configuration changes and AWS CloudTrail to record API calls.
C. Use AWS Config to track configuration changes and Amazon CloudWatch to record API calls.
D. Use AWS CloudTrail to track configuration changes and Amazon CloudWatch to record API calls.

Answer: B

Explanation:

The correct answer is B: Use AWS Config to track configuration changes and AWS CloudTrail to record API calls. This is because AWS Config and AWS CloudTrail are designed for specific and distinct purposes related to compliance, governance, security, and auditing in the AWS cloud.

AWS Config continuously monitors and records AWS resource configurations, enabling you to automate the evaluation of recorded configurations against desired configurations. It provides a detailed view of the configuration of AWS resources, allowing you to assess, audit, and evaluate the configurations of your resources. AWS Config Rules can automatically check whether resources comply with defined standards. This is essential for tracking configuration changes. You can explore more about AWS Config at
https://aws.amazon.com/config/.

AWS CloudTrail, on the other hand, tracks API calls made to AWS resources. It logs account activity, including actions taken through the AWS Management Console, AWS SDKs, command-line tools, and other AWS services. CloudTrail provides an event history of your AWS account activity, including who took what action, the resources that were acted upon, and when the event occurred. This is crucial for security analysis, resource change tracking, and compliance auditing. Comprehensive information about AWS CloudTrail can be found at https://aws.amazon.com/cloudtrail/.

Option A is incorrect because it reverses the roles of CloudTrail and Config. CloudTrail is not designed to track configuration changes directly, and Config is not designed to record API calls. Options C and D are incorrect as Amazon CloudWatch is primarily a monitoring service for metrics and logs, rather than a configuration tracking or API call recording service. While CloudWatch can be used to monitor logs generated by CloudTrail, it isn't a substitute for CloudTrail itself for API call logging. Therefore, AWS Config for configuration changes and AWS CloudTrail for API calls provides a robust solution for compliance, governance, auditing, and security as required by the company.

Question: 35

A company is preparing to launch a public-facing web application in the AWS Cloud. The architecture consists of Amazon EC2 instances within a VPC behind an Elastic Load Balancer (ELB). A third-party service is used for the DNS. The company's solutions architect must recommend a solution to detect and protect against large-scale DDoS attacks.
Which solution meets these requirements?

A. Enable Amazon GuardDuty on the account.

B. Enable Amazon Inspector on the EC2 instances.

C. Enable AWS Shield and assign Amazon Route 53 to it.

D. Enable AWS Shield Advanced and assign the ELB to it.

Answer: D

Explanation:

Here's a detailed justification for why option D is the correct answer:

The problem requires protection against large-scale DDoS attacks for a public-facing web application. Let's analyze each option:

A. Enable Amazon GuardDuty on the account: GuardDuty is a threat detection service that monitors for malicious activity and unauthorized behavior. While it's good for security, it's not specifically designed for DDoS protection. It primarily detects threats after they've potentially impacted the application. It won't automatically mitigate large-scale DDoS attacks in real-time.

B. Enable Amazon Inspector on the EC2 instances: Inspector is a vulnerability assessment service that analyzes the security posture of EC2 instances. It identifies software vulnerabilities and unintended network configurations. While it's useful for general security, it doesn't directly address large-scale DDoS attacks at the network layer.

C. Enable AWS Shield and assign Amazon Route 53 to it: AWS Shield Standard is automatically enabled for all AWS customers and provides basic DDoS protection against common, frequently occurring network and transport layer DDoS attacks. While it's a good starting point, it doesn't offer the customized protection and advanced mitigation techniques required for large-scale, sophisticated DDoS attacks. Assigning Route 53 would protect the DNS layer, but the question focuses on the web application behind the ELB.

D. Enable AWS Shield Advanced and assign the ELB to it: AWS Shield Advanced provides enhanced DDoS protection tailored to your specific application. By assigning the ELB to Shield Advanced, you get 24/7 access to the AWS DDoS Response Team (DRT), customized protection rules, and cost protection during DDoS events. Shield Advanced can detect and automatically mitigate sophisticated DDoS attacks, ensuring the availability of your web application. Since the application is behind an ELB, protecting the ELB directly protects the underlying EC2 instances. This is the only option that provides comprehensive DDoS protection for the described architecture.

In summary, AWS Shield Advanced, specifically protecting the ELB, provides the best solution to detect and mitigate large-scale DDoS attacks targeted at the public-facing web application.

Authoritative Links:

AWS Shield: https://aws.amazon.com/shield/
AWS Shield Advanced: https://aws.amazon.com/shield/advanced/ Elastic
Load Balancing: https://aws.amazon.com/elasticloadbalancing/ Amazon
GuardDuty: https://aws.amazon.com/guardduty/
Amazon Inspector: https://aws.amazon.com/inspector/

Question: 36

A company is building an application in the AWS Cloud. The application will store data in Amazon S3 buckets in two AWS Regions. The company must use an AWS Key Management Service (AWS KMS) customer managed key to encrypt all data that is stored in the S3 buckets. The data in both S3 buckets must be encrypted and decrypted with the same KMS key. The data and the key must be stored in each of the two Regions.
Which solution will meet these requirements with the LEAST operational overhead?

A. Create an S3 bucket in each Region. Configure the S3 buckets to use server-side encryption with Amazon S3 managed encryption keys (SSE-S3). Configure replication between the S3 buckets.

B. Create a customer managed multi-Region KMS key. Create an S3 bucket in each Region. Configure replication between the S3 buckets. Configure the application to use the KMS key with client-side encryption.

C. Create a customer managed KMS key and an S3 bucket in each Region. Configure the S3 buckets to use server-side encryption with Amazon S3 managed encryption keys (SSE-S3). Configure replication between the S3 buckets.

D. Create a customer managed KMS key and an S3 bucket in each Region. Configure the S3 buckets to use server-side encryption with AWS KMS keys (SSE-KMS). Configure replication between the S3 buckets.

Answer: B

Explanation:

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

B. Create a customer managed multi-Region KMS key. Create an S3 bucket in each Region. Configure replication between the S3 buckets. Configure the application to use the KMS key with client-side encryption.

This solution meets all the requirements with the least operational overhead:

1. **Customer Managed Key:** It utilizes a customer managed key, fulfilling the requirement to use a KMS key that the company controls. Critically, it uses a multi-Region KMS key.
2. **Same Key in Both Regions:** A multi-Region key ensures that the same logical key is available in both Regions for encryption and decryption. This is crucial for seamless data access regardless of which Region the data resides in.
3. **Data and Key Stored in Each Region:** Multi-Region keys guarantee the key material exists within both specified Regions, addressing the regional storage requirement.
4. **Encryption with KMS Key:** The solution specifies client-side encryption. This means the application encrypts the data before sending it to S3 using the multi-Region KMS key. This ensures encryption at rest using the KMS key, as the data is already encrypted when stored in S3.
5. **Least Operational Overhead:** This approach involves a relatively straightforward setup. The application manages encryption/decryption, which might add some development complexity initially, but it simplifies key management significantly compared to server-side encryption and key replication.

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

A & C: These options utilize SSE-S3. SSE-S3 uses encryption keys managed by AWS, failing the requirement to use a customer managed KMS key.

D: This solution proposes SSE-KMS with a KMS key. While it uses a customer managed key, it does not use a multi-Region key. Without a multi-Region key, replicating data between Regions encrypted with KMS becomes complex. You'd have to either:

Change the key used during replication, which adds significant operational overhead.
Re-encrypt the data after replication, again adding complexity.Furthermore, even if the same key was manually replicated to two different regions, the key ID would be different, and it would be treated as two distinct keys.

In summary, option B provides the most efficient and manageable way to meet all the requirements. It leverages the benefits of a multi-Region KMS key for seamless data encryption and decryption across both AWS Regions with minimal operational burden.

Supporting Documentation:

AWS KMS Multi-Region Keys:https://docs.aws.amazon.com/kms/latest/developerguide/multi-region-keys-overview.html
Amazon S3 Encryption:https://docs.aws.amazon.com/AmazonS3/latest/userguide/serv-side-encryption.html

Question: 37

A company recently launched a variety of new workloads on Amazon EC2 instances in its AWS account. The company needs to create a strategy to access and administer the instances remotely and securely. The company needs to implement a repeatable process that works with native AWS services and follows the AWS Well-Architected Framework.
Which solution will meet these requirements with the LEAST operational overhead?

A. Use the EC2 serial console to directly access the terminal interface of each instance for administration.

B. Attach the appropriate IAM role to each existing instance and new instance. Use AWS Systems Manager Session Manager to establish a remote SSH session.

C. Create an administrative SSH key pair. Load the public key into each EC2 instance. Deploy a bastion host in a public subnet to provide a tunnel for administration of each instance.

D. Establish an AWS Site-to-Site VPN connection. Instruct administrators to use their local on-premises machines to connect directly to the instances by using SSH keys across the VPN tunnel.

Answer: B

Explanation:

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

B. Attach the appropriate IAM role to each existing instance and new instance. Use AWS Systems Manager Session Manager to establish a remote SSH session.

This solution is the most aligned with the prompt's requirements of security, repeatability, native AWS services, and minimal operational overhead. AWS Systems Manager (SSM) Session Manager allows you to securely manage EC2 instances without needing to open inbound ports (like SSH 22) or manage SSH keys. By using IAM roles, you can precisely define the permissions granted to SSM, ensuring only authorized users and instances can initiate sessions. This approach minimizes the attack surface and reduces the risk of
unauthorized access. SSM Session Manager also integrates with AWS CloudTrail for auditing purposes, providing a clear record of all session activities. Furthermore, it leverages the AWS global infrastructure and eliminates the need to manage bastion hosts or VPNs, thus minimizing operational overhead. The "repeatable process" is achieved by consistently applying the same IAM roles across all instances.

Here's a breakdown of why the other options are less ideal:

A. Use the EC2 serial console to directly access the terminal interface of each instance for administration. While useful for troubleshooting boot issues, the serial console isn't intended for routine administration. It lacks the auditability and security features of SSM Session Manager and requires direct AWS Management Console access. It also does not scale well for a large number of instances.

C. Create an administrative SSH key pair. Load the public key into each EC2 instance. Deploy a bastion host in a public subnet to provide a tunnel for administration of each instance. This solution introduces significant operational overhead. It requires managing SSH keys, securing the bastion host, and configuring routing rules. Maintaining the security and availability of a bastion host adds complexity and ongoing management. SSH keys also present a security risk if compromised.

D. Establish an AWS Site-to-Site VPN connection. Instruct administrators to use their local on-premises machines to connect directly to the instances by using SSH keys across the VPN tunnel. This introduces reliance on on-premises infrastructure and the VPN connection. It increases complexity, maintenance overhead, and network latency. Additionally, managing SSH keys and distributing them securely to administrators becomes an operational burden and security risk.

In summary, **Option B** leverages native AWS services (IAM and SSM) to provide secure, auditable, and centrally managed access to EC2 instances with minimal operational burden, which is a best practice according to the AWS Well-Architected Framework.

Supporting Links:

AWS Systems Manager Session Manager:https://docs.aws.amazon.com/systems-manager/latest/userguide/session-manager.html
AWS IAM Roles:https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html **AWS Well-Architected Framework:**https://aws.amazon.com/architecture/well-architected/

Question: 38

A company is hosting a static website on Amazon S3 and is using Amazon Route 53 for DNS. The website is experiencing increased demand from around the world. The company must decrease latency for users who access the website.
Which solution meets these requirements MOST cost-effectively?

A. Replicate the S3 bucket that contains the website to all AWS Regions. Add Route 53 geolocation routing entries.

B. Provision accelerators in AWS Global Accelerator. Associate the supplied IP addresses with the S3 bucket. Edit the Route 53 entries to point to the IP addresses of the accelerators.

C. Add an Amazon CloudFront distribution in front of the S3 bucket. Edit the Route 53 entries to point to the CloudFront distribution.

D. Enable S3 Transfer Acceleration on the bucket. Edit the Route 53 entries to point to the new endpoint.

Answer: C

Explanation:

Here's a detailed justification for why option C is the most cost-effective solution for reducing latency for a static website hosted on S3 using Route 53, while dealing with increased global demand:

Option C, adding an Amazon CloudFront distribution in front of the S3 bucket and updating Route 53 records, leverages the power of Content Delivery Networks (CDNs). CloudFront caches the static website content at edge locations around the world. When a user requests content, CloudFront serves it from the nearest edge location, significantly reducing latency compared to fetching it from the origin S3 bucket, which might be geographically distant. This caching mechanism distributes the load and minimizes the impact of increased global demand on the origin server. CloudFront's pay-as-you-go pricing model makes it cost-effective as you only pay for the data transferred and requests served.

Option A is less cost-effective. Replicating the S3 bucket to all AWS Regions is an overkill for a static website. Moreover, managing replication across all regions and configuring Route 53 geolocation routing can be complex and expensive without a significant value add. While geolocation routing would direct users to the closest replica, the replication costs would likely be excessive.

Option B, using AWS Global Accelerator, is more appropriate for dynamic content or applications that require TCP or UDP connections. While it could improve performance, it's designed for accelerating dynamic content and is generally more expensive than CloudFront for static content. Global Accelerator also introduces complexity in terms of provisioning and managing accelerators.

Option D, enabling S3 Transfer Acceleration, improves the speed of uploads and downloads to S3 using optimized network paths. While beneficial for data transfer to S3, it does not directly address the need to reduce latency for users accessing the static website content. It speeds up transfers to S3, but users still retrieve data directly from the bucket unless a CDN is used. Therefore, it is not suitable.

In summary, CloudFront offers the most efficient and cost-effective solution for caching static content and delivering it to users with low latency globally, perfectly aligning with the requirements of the question.

Relevant links for further research:

Amazon CloudFront:https://aws.amazon.com/cloudfront/
Amazon S3:https://aws.amazon.com/s3/
Amazon Route 53:https://aws.amazon.com/route53/
AWS Global Accelerator:https://aws.amazon.com/global-accelerator/ **S3 Transfer Acceleration:**https://aws.amazon.com/s3/transfer-acceleration/

Question: 39

A company maintains a searchable repository of items on its website. The data is stored in an Amazon RDS for MySQL database table that contains more than 10 million rows. The database has 2 TB of General Purpose SSD storage. There are millions of updates against this data every day through the company's website.
The company has noticed that some insert operations are taking 10 seconds or longer. The company has determined that the database storage performance is the problem.
Which solution addresses this performance issue?

A. Change the storage type to Provisioned IOPS SSD.
B. Change the DB instance to a memory optimized instance class.
C. Change the DB instance to a burstable performance instance class.
D. Enable Multi-AZ RDS read replicas with MySQL native asynchronous replication.

Answer: A

Explanation:

The question highlights slow insert operations in an Amazon RDS for MySQL database due to database storage performance limitations. General Purpose SSD (gp2/gp3) storage, while suitable for many workloads, might not be sufficient for high-write workloads with millions of daily updates and large tables. Option A, changing the storage type to Provisioned IOPS SSD (io1/io2), directly addresses the performance bottleneck.
Provisioned IOPS SSD allows you to specify a consistent IOPS (Input/Output Operations Per Second) rate, guaranteeing a predictable performance level. This is crucial for applications requiring low latency and high throughput, especially those experiencing latency due to storage I/O.

Option B, changing the DB instance to a memory-optimized instance class, primarily addresses CPU and memory limitations, not storage performance. While more memory can improve caching, it doesn't solve underlying storage bottlenecks. Option C, changing to a burstable performance instance class, is unsuitable because burstable instances rely on credit accumulation and can experience performance degradation when credits are exhausted under sustained high workloads like millions of daily updates. Option D, enabling Multi-AZ RDS read replicas, improves read scalability and high availability but doesn't address the write
performance issues on the primary database instance causing the slow insert operations. Replication adds overhead to the primary instance.

Therefore, switching to Provisioned IOPS SSD is the optimal solution as it provides consistent and guaranteed I/O performance, mitigating the latency issues related to storage operations. For further reading:

Amazon RDS Storage Types:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_Storage.html

Provisioned IOPS SSD (io1/io2):https://aws.amazon.com/rds/pricing/ (Refer to the storage section)

Question: 40

A company has thousands of edge devices that collectively generate 1 TB of status alerts each day. Each alert is approximately 2 KB in size. A solutions architect needs to implement a solution to ingest and store the alerts for future analysis.
The company wants a highly available solution. However, the company needs to minimize costs and does not want to manage additional infrastructure. Additionally, the company wants to keep 14 days of data available for immediate analysis and archive any data older than 14 days.
What is the MOST operationally efficient solution that meets these requirements?

A. Create an Amazon Kinesis Data Firehose delivery stream to ingest the alerts. Configure the Kinesis Data Firehose stream to deliver the alerts to an Amazon S3 bucket. Set up an S3 Lifecycle configuration to transition data to Amazon S3 Glacier after 14 days.

B. Launch Amazon EC2 instances across two Availability Zones and place them behind an Elastic Load Balancer to ingest the alerts. Create a script on the EC2 instances that will store the alerts in an Amazon S3 bucket. Set up an S3 Lifecycle configuration to transition data to Amazon S3 Glacier after 14 days.

C. Create an Amazon Kinesis Data Firehose delivery stream to ingest the alerts. Configure the Kinesis Data Firehose stream to deliver the alerts to an Amazon OpenSearch Service (Amazon Elasticsearch Service) cluster.
Set up the Amazon OpenSearch Service (Amazon Elasticsearch Service) cluster to take manual snapshots every day and delete data from the cluster that is older than 14 days.

D. Create an Amazon Simple Queue Service (Amazon SQS) standard queue to ingest the alerts, and set the message retention period to 14 days. Configure consumers to poll the SQS queue, check the age of the message, and analyze the message data as needed. If the message is 14 days old, the consumer should copy the message to an Amazon S3 bucket and delete the message from the SQS queue.

Answer: A

Explanation:

The most operationally efficient solution is A because it leverages fully managed services that require minimal infrastructure management and are cost-effective.

Why A is the best solution:

Kinesis Data Firehose: It is a fully managed service designed for streaming data ingestion into destinations like S3. It handles scaling, buffering, and data transformation (if needed) automatically, removing the operational overhead of managing servers or clusters.

Amazon S3: It provides highly durable and scalable storage. The pay-as-you-go model and low cost makes it cost-effective.

S3 Lifecycle Management: This feature automates the process of transitioning data between different storage tiers. Configuring a lifecycle policy to move data to Glacier after 14 days automatically handles archiving, reducing storage costs for older data without manual intervention.

Why other options are less suitable:

B: Involves manually managing EC2 instances and load balancers, which increases operational overhead. Scripting the data storage process also requires manual configuration and maintenance.

C: Amazon OpenSearch Service (successor to Elasticsearch Service) is not the best choice for long-term archival storage, as it is more suited for search and analysis of recent data. OpenSearch Service can be expensive.

D: SQS is not designed for storing large amounts of data long-term. Consumers would need to actively process the data as it arrives and manage the archival process. This approach introduces complexity and potential points of failure and it does not allow the full 1TB of daily alerts to be retained in SQS due to the limits of its data retention policies. Also, SQS's limit message size of 256KB makes it unsuitable for 2KB status alerts if thousands of edge devices are sending them.

Supporting Concepts:

Serverless Computing: Kinesis Data Firehose and S3 are serverless services, which minimize the operational burden.

Data Lifecycle Management: S3 Lifecycle Policies are designed to automate data tiering based on age or other criteria, optimizing costs and storage management.

Cost Optimization: By using S3 Glacier for long-term archival, the solution minimizes storage costs.

Authoritative Links:

Amazon Kinesis Data Firehose:https://aws.amazon.com/kinesis/data-firehose/
Amazon S3 Lifecycle Management:https://docs.aws.amazon.com/AmazonS3/latest/userguide/lifecycle-configuration-concept.html
Amazon S3 Glacier:https://aws.amazon.com/glacier/

Question: 41

A company's application integrates with multiple software-as-a-service (SaaS) sources for data collection. The company runs Amazon EC2 instances to receive the data and to upload the data to an Amazon S3 bucket for analysis. The same EC2 instance that receives and uploads the data also sends a notification to the user when an upload is complete. The company has noticed slow application performance and wants to improve the performance as much as possible.
Which solution will meet these requirements with the LEAST operational overhead?

A. Create an Auto Scaling group so that EC2 instances can scale out. Configure an S3 event notification to send events to an Amazon Simple Notification Service (Amazon SNS) topic when the upload to the S3 bucket is complete.

B. Create an Amazon AppFlow flow to transfer data between each SaaS source and the S3 bucket. Configure an S3 event notification to send events to an Amazon Simple Notification Service (Amazon SNS) topic when the upload to the S3 bucket is complete.

C. Create an Amazon EventBridge (Amazon CloudWatch Events) rule for each SaaS source to send output data. Configure the S3 bucket as the rule's target. Create a second EventBridge (Cloud Watch Events) rule to send events when the upload to the S3 bucket is complete. Configure an Amazon Simple Notification Service (Amazon SNS) topic as the second rule's target.

D. Create a Docker container to use instead of an EC2 instance. Host the containerized application on Amazon Elastic Container Service (Amazon ECS). Configure Amazon CloudWatch Container Insights to send events to an Amazon Simple Notification Service (Amazon SNS) topic when the upload to the S3 bucket is complete.

Answer: B

Explanation:

The most efficient solution to improve application performance with minimal operational overhead is to use Amazon AppFlow for data transfer and S3 event notifications with SNS.

Option B is the best because:

AppFlow for Data Transfer: AppFlow directly transfers data from SaaS applications to S3 without the need for EC2 instances. This eliminates the EC2 instance bottleneck and reduces the operational burden of managing those instances. AppFlow is specifically designed for SaaS data integration and automates much of the process.
https://aws.amazon.com/appflow/
S3 Event Notifications & SNS: S3 event notifications trigger an SNS topic when an upload completes. This is a serverless and highly scalable approach to notify users without burdening the data transfer process. SNS provides a simple and reliable way to publish messages to subscribers. https://aws.amazon.com/sns/

Why other options are not optimal:

Option A (Auto Scaling): Scaling EC2 instances might improve throughput, but it doesn't address the fundamental inefficiency of using EC2 for simple data transfer from SaaS sources. It also increases operational complexity.

Option C (EventBridge): EventBridge can handle events, but using it directly for SaaS data integration isn't its primary purpose. It's better suited for routing events within AWS services, not directly interfacing with external SaaS applications. AppFlow is purpose-built for SaaS integrations. Furthermore, using EventBridge as a target to directly write into S3 is not a best practice.

Option D (Docker/ECS): Containerizing the application doesn't fundamentally change the data transfer bottleneck. EC2 (or ECS) is still involved in receiving and uploading data. Also, CloudWatch Container Insights is for monitoring container performance, not for triggering notifications based on S3 uploads.

In conclusion, the solution involving AppFlow and S3 event notifications/SNS offers the most direct and efficient way to address the slow application performance while minimizing operational overhead. This aligns with the principle of using managed services to reduce manual intervention and improve scalability.

Question: 42

A company runs a highly available image-processing application on Amazon EC2 instances in a single VPC. The EC2 instances run inside several subnets across multiple Availability Zones. The EC2 instances do not communicate with each other. However, the EC2 instances download images from Amazon S3 and upload images to Amazon S3 through a single NAT gateway. The company is concerned about data transfer charges.
What is the MOST cost-effective way for the company to avoid Regional data transfer charges?

A. Launch the NAT gateway in each Availability Zone.

B. Replace the NAT gateway with a NAT instance.

C. Deploy a gateway VPC endpoint for Amazon S3.

D. Provision an EC2 Dedicated Host to run the EC2 instances.

Answer: C

Explanation:

The question is focused on minimizing Regional data transfer costs for EC2 instances downloading from and uploading to S3 through a NAT gateway.

Option C: Deploy a gateway VPC endpoint for Amazon S3 is the most cost-effective solution. Gateway endpoints provide connectivity to S3 without using the internet gateway or NAT gateway. This avoids data transfer charges associated with routing traffic through the NAT gateway. Data transferred between EC2 instances in the VPC and S3 via the gateway endpoint stays within the AWS network, eliminating Regional data transfer charges.

Option A: Launch the NAT gateway in each Availability Zone does not avoid Regional data transfer charges, although it improves availability. Data still traverses the NAT gateway, and the associated costs remain. It primarily prevents a single NAT gateway failure from impacting all AZs.

Option B: Replace the NAT gateway with a NAT instance offers no cost benefit and introduces operational overhead. NAT instances are not managed by AWS and require manual configuration, patching, and scaling. Moreover, data transfer charges still apply.

Option D: Provision an EC2 Dedicated Host to run the EC2 instances is irrelevant to the problem. Dedicated Hosts provide hardware-level isolation but do not affect data transfer charges associated with S3 access.

Therefore, leveraging a gateway VPC endpoint is the most efficient way to eliminate Regional data transfer charges for S3 access from EC2 instances within a VPC.

Supporting resources:

VPC Endpoints
NAT Gateways
Understanding AWS Data Transfer Costs

Question: 43

A company has an on-premises application that generates a large amount of time-sensitive data that is backed up to Amazon S3. The application has grown and there are user complaints about internet bandwidth limitations. A solutions architect needs to design a long-term solution that allows for both timely backups to Amazon S3 and with minimal impact on internet connectivity for internal users.
Which solution meets these requirements?

A. Establish AWS VPN connections and proxy all traffic through a VPC gateway endpoint.

B. Establish a new AWS Direct Connect connection and direct backup traffic through this new connection.

C. Order daily AWS Snowball devices. Load the data onto the Snowball devices and return the devices to AWS each day.

D. Submit a support ticket through the AWS Management Console. Request the removal of S3 service limits from the account.

Answer: B

Explanation:

The best solution is **B. Establish a new AWS Direct Connect connection and direct backup traffic through this new connection.**

Here's why:

Bandwidth Bottleneck: The core issue is limited internet bandwidth affecting users due to large data backups to S3.

AWS Direct Connect: Direct Connect establishes a dedicated, private network connection between your on-premises infrastructure and AWS. This bypasses the public internet, providing consistent and often higher bandwidth, lower latency, and more predictable network performance.

Minimal Impact on Internet: By routing the backup traffic through Direct Connect, it won't compete for bandwidth with other internet-bound traffic, alleviating the strain on internal users' connectivity.

Timely Backups: The dedicated connection ensures backups can complete efficiently and on time, as it's not subject to internet congestion.

Long-term Solution: Direct Connect is a durable infrastructure solution, ideal for continuous data transfer needs.

Why other options are less ideal:

A. AWS VPN and VPC Gateway Endpoint: While VPN provides secure connectivity, it still relies on the existing internet bandwidth. VPC gateway endpoints allow access to S3 from within a VPC without using public IPs, but the data still travels over the internet to the on-premises network. It does not solve the bandwidth issue.

C. AWS Snowball: Snowball is useful for initial large-scale data migrations. Using it daily for backups is impractical, expensive, and introduces significant delays in data availability.

D. Support Ticket for S3 Limits: S3 service limits are designed for the overall health and availability of the service, not for solving bandwidth problems. Requesting their removal is unlikely to be successful and isn't a proper architectural solution.

Authoritative Links:

AWS Direct Connect:https://aws.amazon.com/directconnect/
AWS VPN:https://aws.amazon.com/vpn/
AWS Snowball:https://aws.amazon.com/snowball/

Question: 44

A company has an Amazon S3 bucket that contains critical data. The company must protect the data from accidental deletion.
Which combination of steps should a solutions architect take to meet these requirements? (Choose two.)

A. Enable versioning on the S3 bucket.
B. Enable MFA Delete on the S3 bucket.
C. Create a bucket policy on the S3 bucket.
D. Enable default encryption on the S3 bucket.
E. Create a lifecycle policy for the objects in the S3 bucket.

Answer: AB

Explanation:

The correct answer is AB.

Justification:

The primary requirement is to protect the data from accidental deletion. Two features in S3 directly address this: Versioning and MFA Delete.

A. Enable versioning on the S3 bucket: S3 Versioning ensures that every version of an object is preserved, even if it's deleted or overwritten. When an object is deleted, it doesn't truly disappear; instead, a delete marker is created. The previous version remains accessible, allowing for easy recovery from accidental deletions. This is crucial for data protection.
https://docs.aws.amazon.com/AmazonS3/latest/userguide/versioning-workflows.html

B. Enable MFA Delete on the S3 bucket: MFA Delete adds an extra layer of security to prevent accidental or malicious deletions. When MFA Delete is enabled, deleting a versioned object or permanently deleting the S3 bucket itself requires multi-factor authentication. This significantly reduces the risk of unauthorized or unintentional data loss. It enforces the principle of "something you know" (password) and "something you have" (MFA device), strengthening the deletion process.
https://docs.aws.amazon.com/AmazonS3/latest/userguide/Versioning.html#MultiFactorAuthenticationDelete

Let's examine why the other options are incorrect:

C. Create a bucket policy on the S3 bucket: While bucket policies control access to the bucket, they don't prevent accidental deletions. You can restrict who can delete objects, but if someone with the delete permission makes a mistake, the policy won't prevent it.

D. Enable default encryption on the S3 bucket: Default encryption protects data at rest, addressing security and compliance concerns related to data confidentiality. However, encryption does not prevent accidental deletion.

E. Create a lifecycle policy for the objects in the S3 bucket: Lifecycle policies manage the object lifecycle, automatically transitioning them to cheaper storage classes or deleting them after a specified period. If misconfigured, a lifecycle policy could actually cause accidental deletions. It's for automating data archival/deletion based on pre-defined rules, not for general protection against accidental deletion.

Question: 45

A company has a data ingestion workflow that consists of the following:
• An Amazon Simple Notification Service (Amazon SNS) topic for notifications about new data deliveries• An AWS Lambda function to process the data and record metadata
The company observes that the ingestion workflow fails occasionally because of network connectivity issues.
When such a failure occurs, the Lambda function does not ingest the corresponding data unless the company manually reruns the job.
Which combination of actions should a solutions architect take to ensure that the Lambda function ingests all data in the future? (Choose two.)

A. Deploy the Lambda function in multiple Availability Zones.
B. Create an Amazon Simple Queue Service (Amazon SQS) queue, and subscribe it to the SNS topic.
C. Increase the CPU and memory that are allocated to the Lambda function.
D. Increase provisioned throughput for the Lambda function.
E. Modify the Lambda function to read from an Amazon Simple Queue Service (Amazon SQS) queue.

Answer: BE

Explanation:

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

B. Create an Amazon Simple Queue Service (Amazon SQS) queue, and subscribe it to the SNS topic.

This is a crucial step for building a reliable and resilient data ingestion pipeline. When an SNS topic is connected to an SQS queue via subscription, messages published to the SNS topic are automatically queued in the SQS queue. SQS provides a mechanism to buffer the messages, ensuring that they are not lost even if the Lambda function is temporarily unavailable due to network issues or other transient failures. This buffering helps decoupling the data producers (SNS) from the data consumers (Lambda).

E. Modify the Lambda function to read from an Amazon Simple Queue Service (Amazon SQS) queue.

By having the Lambda function consume messages from the SQS queue, the function becomes more fault-tolerant. If a processing failure occurs during the initial Lambda execution (due to network issue), the message remains in the SQS queue. SQS can be configured to automatically retry the message delivery to the Lambda function after a visibility timeout. This ensures that the Lambda function eventually processes all messages, providing at-least-once delivery semantics. The queue acts as a buffer and retry mechanism, guaranteeing data ingestion despite intermittent failures.https://docs.aws.amazon.com/lambda/latest/dg/services-sqs.html

Why other options are not suitable:

A. Deploy the Lambda function in multiple Availability Zones. While deploying Lambda in multiple AZs improves availability, it doesn't address message loss caused by transient network issues during the initial invocation. Lambda inherently runs in multiple AZs managed by AWS.

C. Increase the CPU and memory that are allocated to the Lambda function. Increasing CPU and memory might help with performance if the Lambda function is resource-constrained, but it won't prevent message loss due to network issues.

D. Increase provisioned throughput for the Lambda function. Lambda does not use provisioned throughput, and this action is not applicable. Also, it does not prevent message loss due to network issues.

Question: 46

A company has an application that provides marketing services to stores. The services are based on previous purchases by store customers. The stores upload transaction data to the company through SFTP, and the data is processed and analyzed to generate new marketing offers. Some of the files can exceed 200 GB in size. Recently, the company discovered that some of the stores have uploaded files that contain personally identifiable information (PII) that should not have been included. The company wants administrators to be alerted if PII is shared again. The company also wants to automate remediation.
What should a solutions architect do to meet these requirements with the LEAST development effort?

A. Use an Amazon S3 bucket as a secure transfer point. Use Amazon Inspector to scan the objects in the bucket. If objects contain PII, trigger an S3 Lifecycle policy to remove the objects that contain PII.

B. Use an Amazon S3 bucket as a secure transfer point. Use Amazon Macie to scan the objects in the bucket. If objects contain PII, use Amazon Simple Notification Service (Amazon SNS) to trigger a notification to the administrators to remove the objects that contain PII.

C. Implement custom scanning algorithms in an AWS Lambda function. Trigger the function when objects are loaded into the bucket. If objects contain PII, use Amazon Simple Notification Service (Amazon SNS) to trigger a notification to the administrators to remove the objects that contain PII.

D. Implement custom scanning algorithms in an AWS Lambda function. Trigger the function when objects are loaded into the bucket. If objects contain PII, use Amazon Simple Email Service (Amazon SES) to trigger a notification to the administrators and trigger an S3 Lifecycle policy to remove the meats that contain PII.

Answer: B

Explanation:

The correct answer is B because it leverages purpose-built AWS services for data security and PII detection with minimal development effort. Here's a detailed justification:

S3 as a Secure Transfer Point: Using S3 provides a scalable and secure storage location for the uploaded files, replacing the potentially less secure SFTP method. S3 integrates well with other AWS services, simplifying the overall workflow.

Amazon Macie for PII Detection: Amazon Macie is a fully managed data security and data privacy service that uses machine learning and pattern matching to discover sensitive data, including PII. Macie is designed specifically for this purpose and eliminates the need to develop custom scanning algorithms, adhering to the principle of "least development effort". It can automatically detect a wide range of PII types.

SNS for Notifications: Amazon SNS is a simple and cost-effective way to send notifications to administrators when PII is detected. This allows for immediate action and investigation.

Why other options are less suitable:

Option A suggests using Amazon Inspector, which primarily focuses on identifying security vulnerabilities and deviations from security best practices within EC2 instances and container images, and doesn't specialize in PII detection within S3 objects like Macie. Also S3 Lifecycle policy removing without investigation is not ideal. Options C and D propose implementing custom scanning algorithms in Lambda. Developing and maintaining custom scanning algorithms would require significant development effort and expertise in PII identification, which contradicts the "least development effort" requirement. Option D uses SES which is similar to SNS, however, SES requires more configuration and isn't as straightforward for simple notification use cases. Option D also suggests removing "meats" which is not an option from the question.

In summary, option B provides a pre-built and managed solution for PII detection and notification, ensuring minimal development effort and a robust security posture.

Supporting Links:

Amazon Macie:https://aws.amazon.com/macie/
Amazon S3:https://aws.amazon.com/s3/

Amazon SNS:https://aws.amazon.com/sns/

Question: 47

A company needs guaranteed Amazon EC2 capacity in three specific Availability Zones in a specific AWS Region for an upcoming event that will last 1 week.
What should the company do to guarantee the EC2 capacity?

A. Purchase Reserved Instances that specify the Region needed.

B. Create an On-Demand Capacity Reservation that specifies the Region needed.

C. Purchase Reserved Instances that specify the Region and three Availability Zones needed.

D. Create an On-Demand Capacity Reservation that specifies the Region and three Availability Zones needed.

Answer: D

Explanation:

The correct answer is D: Create an On-Demand Capacity Reservation that specifies the Region and three Availability Zones needed.

Here's why:

On-Demand Capacity Reservations (ODCR): ODCRs provide a way to reserve EC2 instance capacity in a specific Availability Zone for a specified duration. This guarantees that the required EC2 capacity will be available when needed. Crucially, they allow you to target specific Availability Zones.

AWS Documentation on Capacity Reservations

Reserved Instances (RI): While RIs offer a billing discount in exchange for a term commitment, they do not guarantee capacity. They provide discounted pricing for instances used but do not explicitly reserve resources in specific Availability Zones. RIs help lower costs, they do not fulfill the requirement for guaranteed capacity. Regional RIs apply to any AZ in the region.

AWS Documentation on Reserved Instances

Why other options are incorrect:

A & C: Purchasing Reserved Instances: As stated above, RIs do not guarantee EC2 capacity. Purchasing RIs only reduces the cost of running instances; it doesn't reserve physical resources.

B: Create an On-Demand Capacity Reservation that specifies the Region needed: This is not precise enough.

The company requires capacity reservations in three specific Availability Zones. Specifying only the region would not guarantee the resources would be in the desired AZs.

Given the short duration (1 week) and the need for a guarantee the On-Demand Capacity Reservation is a perfect match.

Question: 48

A company's website uses an Amazon EC2 instance store for its catalog of items. The company wants to make sure that the catalog is highly available and that the catalog is stored in a durable location.
What should a solutions architect do to meet these requirements?

A. Move the catalog to Amazon ElastiCache for Redis.

B. Deploy a larger EC2 instance with a larger instance store.

C. Move the catalog from the instance store to Amazon S3 Glacier Deep Archive.

D. Move the catalog to an Amazon Elastic File System (Amazon EFS) file system.

Answer: D

Explanation:

The correct answer is D. Here's why:

The question highlights two key requirements: high availability and durable storage for the website's catalog. Instance store volumes are ephemeral, meaning data is lost when the instance stops, terminates, or fails. This makes them unsuitable for durable storage.

Option A, moving to ElastiCache for Redis, isn't ideal for durable storage of a full catalog. Redis is an in-memory data store, primarily used for caching to improve performance, not long-term storage. While data persistence can be configured, it's not the primary use case and introduces complexity compared to other options.

Option B, deploying a larger EC2 instance with a larger instance store, doesn't address the fundamental problem of data loss upon instance failure. It merely provides more ephemeral storage, failing to meet the durability requirement.

Option C, moving to S3 Glacier Deep Archive, provides durable storage but is designed for infrequent access, like archiving. It's unsuitable for a website catalog that requires frequent reads and writes. Retrieval times are measured in hours, making it impractical for serving web content.

Option D, moving to Amazon EFS, is the best solution. EFS provides a scalable, highly available, and durable file system that can be mounted by multiple EC2 instances simultaneously. This allows the catalog to be stored persistently and accessed by the web servers, ensuring high availability and durability. EFS replicates data across multiple Availability Zones, providing resilience against failures.

In summary, EFS addresses both requirements: high availability through shared access and data replication, and durability through persistent storage. It's designed for file-based storage that can be easily accessed by applications, making it a good fit for a website catalog.

Further Reading:

Amazon EFS:https://aws.amazon.com/efs/
Instance Store Volumes:https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/InstanceStorage.html

Question: 49

A company stores call transcript files on a monthly basis. Users access the files randomly within 1 year of the call, but users access the files infrequently after 1 year. The company wants to optimize its solution by giving users the ability to query and retrieve files that are less than 1-year-old as quickly as possible. A delay in retrieving older files is acceptable.
Which solution will meet these requirements MOST cost-effectively?

A. Store individual files with tags in Amazon S3 Glacier Instant Retrieval. Query the tags to retrieve the files from S3 Glacier Instant Retrieval.

B. Store individual files in Amazon S3 Intelligent-Tiering. Use S3 Lifecycle policies to move the files to S3 Glacier Flexible Retrieval after 1 year. Query and retrieve the files that are in Amazon S3 by using Amazon Athena. Query and retrieve the files that are in S3 Glacier by using S3 Glacier Select.

C. Store individual files with tags in Amazon S3 Standard storage. Store search metadata for each archive in Amazon S3 Standard storage. Use S3 Lifecycle policies to move the files to S3 Glacier Instant Retrieval after 1 year. Query and retrieve the files by searching for metadata from Amazon S3.

D. Store individual files in Amazon S3 Standard storage. Use S3 Lifecycle policies to move the files to S3 Glacier Deep Archive after 1 year. Store search metadata in Amazon RDS. Query the files from Amazon RDS. Retrieve the files from S3 Glacier Deep Archive.

Answer: B

Explanation:

Option B is the most cost-effective solution because it leverages S3 Intelligent-Tiering to automatically optimize storage costs based on access patterns, while also utilizing S3 Glacier for long-term archival.

Here's a breakdown:

S3 Intelligent-Tiering: This storage class automatically moves data between frequent, infrequent, and archive access tiers based on changing access patterns. This means frequently accessed files (within the first year) reside in the more expensive but faster access tiers, ensuring quick retrieval. Infrequently accessed files are moved to cheaper tiers automatically.

S3 Lifecycle Policies: These policies automate the transition of objects between storage classes. By moving files older than 1 year to S3 Glacier Flexible Retrieval (formerly S3 Glacier), the company benefits from significantly lower storage costs for infrequently accessed data.

Amazon Athena and S3 Glacier Select: Athena enables querying data directly in S3 using standard SQL, while S3 Glacier Select allows querying data stored in S3 Glacier without needing to restore the entire object. This allows the company to query and retrieve data irrespective of its storage class (S3 Intelligent-Tiering or S3 Glacier).

The other options are less ideal:

Option A: S3 Glacier Instant Retrieval is the most expensive S3 Glacier tier. Using it for frequently accessed files in the first year defeats the purpose of cost optimization. Also, querying based on object tags isn't as efficient as using Athena for structured querying.

Option C: Storing all files in S3 Standard for the first year is more expensive than using S3 Intelligent-Tiering, which dynamically adjusts storage costs. Storing search metadata in S3 Standard storage alongside the data is also redundant, as Athena can be used for that purpose.

Option D: Storing search metadata in Amazon RDS is more complex and expensive than using Athena for querying data in S3. S3 Glacier Deep Archive offers the lowest storage cost but has the highest retrieval times, which might be acceptable for older data but is not necessary when Glacier Flexible Retrieval meets the requirement.

In summary, Option B provides the best balance of performance and cost-effectiveness by using S3 Intelligent-Tiering for frequent access, S3 Lifecycle policies for automated archival, and Athena for querying data across different storage classes.

Authoritative Links:

S3 Intelligent-Tiering
S3 Lifecycle Policies
Amazon Athena
S3 Glacier Select
S3 Storage Classes

Question: 50

A company has a production workload that runs on 1,000 Amazon EC2 Linux instances. The workload is powered by third-party software. The company needs to patch the third-party software on all EC2 instances as quickly as possible to remediate a critical security vulnerability.
What should a solutions architect do to meet these requirements?

A. Create an AWS Lambda function to apply the patch to all EC2 instances.

B. Configure AWS Systems Manager Patch Manager to apply the patch to all EC2 instances.

C. Schedule an AWS Systems Manager maintenance window to apply the patch to all EC2 instances.

D. Use AWS Systems Manager Run Command to run a custom command that applies the patch to all EC2 instances.

Answer: D

Explanation:

The correct answer is **D. Use AWS Systems Manager Run Command to run a custom command that applies the patch to all EC2 instances.**

Here's a detailed justification:

The primary requirement is to apply a third-party software patch to 1,000 EC2 instances as quickly as possible to remediate a critical security vulnerability. Speed and wide distribution are key.

Why Run Command is Best: AWS Systems Manager (SSM) Run Command allows you to remotely and securely manage the configuration of your managed instances. It lets you execute shell scripts or commands on a large number of EC2 instances simultaneously. For rapidly deploying a patch to a known vulnerability, Run Command offers the most immediate and direct route. You create a shell script (or similar) containing the patching logic and then use Run Command to execute it on all instances. The entire operation can be executed within minutes.

Why other options are less suitable:

A. Create an AWS Lambda function: While Lambda is useful for event-driven tasks, it is less suited for directly patching 1,000 EC2 instances. Lambda functions have execution time limits (up to 15 minutes), making it challenging to complete patching on a large number of instances reliably. Moreover, you'd still need to manage the EC2 instance interaction, adding unnecessary complexity. The lambda function itself will need to call SSM to run commands on the instances.

B. Configure AWS Systems Manager Patch Manager: Patch Manager is designed for automated OS-level patching based on patch baselines. It's not ideal for deploying third-party software patches quickly, especially in an emergency situation. Setting up and configuring patch baselines and ensuring they are applied to 1000 machines will take longer than using Run Command directly.

C. Schedule an AWS Systems Manager maintenance window: Maintenance Windows are valuable for scheduled tasks, but not for immediate, emergency patching. They introduce a delay before the patch can be applied. The requirement emphasizes quickly as possible, which Maintenance Windows don't fulfill.

In summary, Run Command is designed for ad-hoc operations and rapid execution of commands across fleets of instances, making it the optimal solution for patching a critical vulnerability as quickly as possible. It provides direct control and immediate deployment capabilities, crucial for addressing security concerns promptly.

Here are authoritative links for further research:

AWS Systems Manager Run Command:https://docs.aws.amazon.com/systems-manager/latest/userguide/execute-remote-commands.html
AWS Systems Manager Patch Manager:https://docs.aws.amazon.com/systems-manager/latest/userguide/patch-manager.html
AWS Systems Manager Maintenance Windows:https://docs.aws.amazon.com/systems-manager/latest/userguide/maintenance-windows.html

Question: 51

A company is developing an application that provides order shipping statistics for retrieval by a REST API. The company wants to extract the shipping statistics, organize the data into an easy-to-read HTML format, and send the report to several email addresses at the same time every morning.
Which combination of steps should a solutions architect take to meet these requirements? (Choose two.)

A. Configure the application to send the data to Amazon Kinesis Data Firehose.

B. Use Amazon Simple Email Service (Amazon SES) to format the data and to send the report by email.

C. Create an Amazon EventBridge (Amazon CloudWatch Events) scheduled event that invokes an AWS Glue job to query the application's API for the data.

D. Create an Amazon EventBridge (Amazon CloudWatch Events) scheduled event that invokes an AWS Lambda function to query the application's API for the data.

E. Store the application data in Amazon S3. Create an Amazon Simple Notification Service (Amazon SNS) topic as an S3 event destination to send the report by email.

Answer: BD

Explanation:

The correct answer is BD. Here's why:

D: Create an Amazon EventBridge (Amazon CloudWatch Events) scheduled event that invokes an AWS Lambda function to query the application's API for the data. Amazon EventBridge allows for scheduled, event-driven automation. In this scenario, it's ideal for triggering the data extraction and report generation process every morning. Lambda offers a serverless compute environment where code can be executed without managing underlying infrastructure. A Lambda function can be coded to query the REST API, retrieve the shipping statistics, and format the data into HTML. This addresses the requirement of extracting data and organizing it.
https://aws.amazon.com/eventbridge/, https://aws.amazon.com/lambda/

B: Use Amazon Simple Email Service (Amazon SES) to format the data and to send the report by email.
Amazon SES is a cost-effective, scalable email service that provides a reliable way to send emails. The Lambda function, after formatting the data into HTML, can leverage SES to send the report to the specified email addresses. SES handles email delivery and can be configured to handle bounces and complaints. This satisfies the requirement of sending the report to multiple email addresses in an easy-to-read HTML format. The Lambda function would take on the additional task of the HTML formatting. https://aws.amazon.com/ses/

Why other options are incorrect:

A: Configure the application to send the data to Amazon Kinesis Data Firehose. Kinesis Data Firehose is used for streaming data to destinations like S3, Redshift, and Elasticsearch. It's not suitable for querying data from an API and generating reports.

C: Create an Amazon EventBridge (Amazon CloudWatch Events) scheduled event that invokes an AWS Glue job to query the application's API for the data. AWS Glue is an ETL (Extract, Transform, Load) service primarily designed for processing large datasets. Using Glue to directly query an API and generate a single HTML report is overkill and less efficient than using Lambda. Glue is usually used against a data lake, not an operational API.

E: Store the application data in Amazon S3. Create an Amazon Simple Notification Service (Amazon SNS) topic as an S3 event destination to send the report by email. S3 is for object storage and would only make sense if the source data (the shipping statistics) were already available within S3, which isn't indicated in the problem description. SNS is for simple notifications and not suited for complex HTML formatting or extracting data from APIs. S3 event notifications are also triggered by data changes, not scheduled events.

Question: 52

A company wants to migrate its on-premises application to AWS. The application produces output files that vary in size from tens of gigabytes to hundreds of terabytes. The application data must be stored in a standard file system structure. The company wants a solution that scales automatically. is highly available, and requires minimum operational overhead.
Which solution will meet these requirements?

A. Migrate the application to run as containers on Amazon Elastic Container Service (Amazon ECS). Use Amazon S3 for storage.

B. Migrate the application to run as containers on Amazon Elastic Kubernetes Service (Amazon EKS). Use Amazon Elastic Block Store (Amazon EBS) for storage.

C. Migrate the application to Amazon EC2 instances in a Multi-AZ Auto Scaling group. Use Amazon Elastic File System (Amazon EFS) for storage.

D. Migrate the application to Amazon EC2 instances in a Multi-AZ Auto Scaling group. Use Amazon Elastic Block Store (Amazon EBS) for storage.

Answer: C

Explanation:

The correct answer is C because it addresses all the requirements: scalability, high availability, standard file system structure, and minimal operational overhead.

Scalability: Amazon EFS scales automatically to petabytes without disrupting applications, handling the varied file sizes (tens of gigabytes to hundreds of terabytes). EC2 Auto Scaling ensures the application instances scale based on demand.

High Availability: Using a Multi-AZ Auto Scaling group ensures that the application is highly available because EC2 instances are spread across multiple Availability Zones. EFS is also designed for high availability and durability, replicating data across multiple AZs.

Standard File System Structure: EFS provides a standard file system interface (NFSv4.1) that allows applications to interact with storage in a familiar way.

Minimum Operational Overhead: EFS is a fully managed service, which eliminates the need for manual provisioning, patching, or backups. EC2 Auto Scaling automates the scaling of instances.

Option A is incorrect because S3 is an object storage service and does not provide a standard file system interface. Option B is incorrect because EBS is block storage and would not scale elastically for the varying size files, and it has more operational overhead compared to EFS. Option D is incorrect because EBS is block storage and not a shared file system. Mounting and sharing EBS volumes between multiple instances introduces complexities.Here are some authoritative links for further research:

Amazon EFS:https://aws.amazon.com/efs/
Amazon EC2 Auto Scaling:https://aws.amazon.com/autoscaling/
Amazon EC2:https://aws.amazon.com/ec2/

Question: 53

A company needs to store its accounting records in Amazon S3. The records must be immediately accessible for 1 year and then must be archived for an additional 9 years. No one at the company, including administrative users and root users, can be able to delete the records during the entire 10-year period. The records must be stored with maximum resiliency.
Which solution will meet these requirements?

A. Store the records in S3 Glacier for the entire 10-year period. Use an access control policy to deny deletion of the records for a period of 10 years.

B. Store the records by using S3 Intelligent-Tiering. Use an IAM policy to deny deletion of the records. After 10 years, change the IAM policy to allow deletion.

C. Use an S3 Lifecycle policy to transition the records from S3 Standard to S3 Glacier Deep Archive after 1 year. Use S3 Object Lock in compliance mode for a period of 10 years.

D. Use an S3 Lifecycle policy to transition the records from S3 Standard to S3 One Zone-Infrequent Access (S3 One Zone-IA) after 1 year. Use S3 Object Lock in governance mode for a period of 10 years.

Answer: C

Explanation:

Here's a detailed justification for why option C is the correct answer:

Option C effectively addresses all requirements: immediate accessibility for 1 year, archival for 9 years, immutability (preventing deletion), and maximum resiliency.

1. **Initial Accessibility:** S3 Standard offers immediate access to the records for the first year.

2. **Archival Storage:** Transitioning the data to S3 Glacier Deep Archive after one year via an S3 Lifecycle policy fulfills the archival requirement and is the most cost-effective storage option for long-term retention with infrequent access. S3 Glacier Deep Archive is designed for data that is rarely accessed.

3. **Immutability:** S3 Object Lock in compliance mode is crucial. Compliance mode ensures that no one, including administrative users or the root user, can delete the objects during the specified retention period (10 years in this case). This definitively meets the immutability requirement.

4. **Resiliency:** S3 Standard and S3 Glacier Deep Archive provide the highest levels of data durability and availability by storing data across multiple devices and facilities.

Why other options are incorrect:

A: S3 Glacier is not designed for immediate accessibility. It's for infrequent access with retrieval times ranging from minutes to hours, not the "immediately accessible" requirement of the first year. Access control policies and IAM Policies can be bypassed or modified, which means they cannot guarantee immutability.

B: S3 Intelligent-Tiering automatically moves data between frequent, infrequent, and archive access tiers based on access patterns. While useful for cost optimization, it doesn't guarantee immutability. IAM policies can be bypassed or modified.

D: S3 One Zone-IA offers lower availability (data is stored in a single Availability Zone), which does not meet the "maximum resiliency" requirement. Also, S3 Object Lock in governance mode can be overridden by users with specific IAM permissions, failing to guarantee immutability against administrative users or root.

Authoritative Links:

S3 Storage Classes:https://aws.amazon.com/s3/storage-classes/
S3 Lifecycle Policies:https://docs.aws.amazon.com/AmazonS3/latest/userguide/lifecycle-configuration-concept.html
S3 Object Lock:https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lock.html
Object Lock Compliance vs Governance Mode:
https://docs.aws.amazon.com/AmazonS3/latest/userguide/object-lock-overview.html

In summary, Option C provides the optimal balance of accessibility, cost-effectiveness, immutability, and resiliency to meet the stated requirements, making it the correct answer.

Question: 54

A company runs multiple Windows workloads on AWS. The company's employees use Windows file shares that are hosted on two Amazon EC2 instances. The file shares synchronize data between themselves and maintain duplicate copies. The company wants a highly available and durable storage solution that preserves how users currently access the files.
What should a solutions architect do to meet these requirements?

A. Migrate all the data to Amazon S3. Set up IAM authentication for users to access files.

B. Set up an Amazon S3 File Gateway. Mount the S3 File Gateway on the existing EC2 instances.

C. Extend the file share environment to Amazon FSx for Windows File Server with a Multi-AZ configuration. Migrate all the data to FSx for Windows File Server.

D. Extend the file share environment to Amazon Elastic File System (Amazon EFS) with a Multi-AZ configuration. Migrate all the data to Amazon EFS.

Answer: C

Explanation:

The correct answer is C. Here's why:

The requirement is to provide a highly available and durable file share solution for Windows workloads while preserving the existing user access patterns.

Option A (Amazon S3 with IAM): While S3 is highly available and durable, it doesn't directly support Windows file shares. Users would need to adapt to a completely different access method, breaking the "preserve how users currently access the files" requirement. IAM authentication, although secure, doesn't replicate the file share experience.

Option B (Amazon S3 File Gateway): S3 File Gateway provides a way to access S3 objects as files, but it doesn't replicate a traditional Windows file share. It adds complexity to the existing EC2 setup and doesn't inherently provide high availability for the gateway itself. Users would still potentially experience access disruptions if the gateway instance fails.

Option C (Amazon FSx for Windows File Server with Multi-AZ): Amazon FSx for Windows File Server is a fully managed service specifically designed for providing native Windows file server capabilities in the cloud. It's built on Windows Server, so it directly supports SMB file shares and integrates seamlessly with existing Active Directory environments for authentication. A Multi-AZ configuration provides high availability by automatically replicating data across multiple Availability Zones. This ensures that if one AZ fails, the file share remains accessible to users. Migrating the data to FSx for Windows File Server preserves the existing user access patterns as they continue to use familiar Windows file share protocols.

Option D (Amazon EFS with Multi-AZ): Amazon EFS is a network file system designed for Linux-based workloads. While it offers high availability and durability, it's not a native Windows file server solution and doesn't directly support SMB file shares. Integrating EFS with Windows workloads would require additional configuration and wouldn't provide the same seamless experience as FSx for Windows File Server.

In conclusion, Option C provides the best solution as it directly addresses all the requirements: high availability, durability, preservation of existing user access methods, and native support for Windows file shares through Amazon FSx for Windows File Server with a Multi-AZ configuration.

Authoritative Links for Further Research:

Amazon FSx for Windows File Server:https://aws.amazon.com/fsx/windows/
High Availability (HA) and Multi-AZ:https://docs.aws.amazon.com/fsx/latest/WindowsGuide/high-availability.html

Question: 55

A solutions architect is developing a VPC architecture that includes multiple subnets. The architecture will host applications that use Amazon EC2 instances and Amazon RDS DB instances. The architecture consists of six subnets in two Availability Zones. Each Availability Zone includes a public subnet, a private subnet, and a dedicated subnet for databases. Only EC2 instances that run in the private subnets can have access to the RDS databases.
Which solution will meet these requirements?

A. Create a new route table that excludes the route to the public subnets' CIDR blocks. Associate the route table with the database subnets.

B. Create a security group that denies inbound traffic from the security group that is assigned to instances in the public subnets. Attach the security group to the DB instances.

C. Create a security group that allows inbound traffic from the security group that is assigned to instances in the private subnets. Attach the security group to the DB instances.

D. Create a new peering connection between the public subnets and the private subnets. Create a different peering connection between the private subnets and the database subnets.

Answer: C

Explanation:

The correct solution (C) leverages security groups to control network access to the RDS instances. Security groups act as virtual firewalls at the instance level, controlling both inbound and outbound traffic. To meet the requirement that only EC2 instances in the private subnets can access the RDS databases, we create a security group that specifically allows inbound traffic only from the security group assigned to those EC2 instances in the private subnets.

Here's why the other options are not ideal:

A: Creating a route table that excludes routes to public subnets' CIDR blocks and associating it with the database subnets would primarily affect routing and not precisely control which instances can access the database. It might block general traffic from public subnets to database subnets, but it wouldn't isolate access to only EC2 instances associated with a specific security group in the private subnets. It also makes the configuration less flexible and scalable.

B: Denying inbound traffic from the public subnet's security group at the database instance is a step in the right direction, but it doesn't explicitly allow traffic from the private subnets. This approach might inadvertently block legitimate traffic originating from other sources within the private subnets that were intended to connect to the database. The best practice is to use a principle of least privilege by allowing only the necessary traffic.

D: Peering connections are for connecting VPCs, not subnets within the same VPC. The EC2 instances and RDS instances are already within the same VPC, so subnet-level peering is unnecessary and not a feasible solution. Peering creates network connectivity between entire VPCs, adding unnecessary complexity and cost for this specific requirement. Subnets within a VPC inherently communicate via the VPC's internal routing.

Therefore, option C is the most precise, secure, and efficient way to ensure that only EC2 instances in the private subnets can access the RDS instances, adhering to the principle of least privilege and utilizing the inherent security features provided by security groups.

Supporting Links:

Amazon VPC Security Groups:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-security-groups.html
Controlling Traffic to Resources Using Security Groups:
https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/USER_WorkingWithSecurityGroups.html

Question: 56

A company has registered its domain name with Amazon Route 53. The company uses Amazon API Gateway in the ca-central-1 Region as a public interface for its backend microservice APIs. Third-party services consume the APIs securely. The company wants to design its API Gateway URL with the company's domain name and corresponding certificate so that the third-party services can use HTTPS.
Which solution will meet these requirements?

A. Create stage variables in API Gateway with Name="Endpoint-URL" and Value="Company Domain Name" to overwrite the default URL. Import the public certificate associated with the company's domain name into AWS Certificate Manager (ACM).

B. Create Route 53 DNS records with the company's domain name. Point the alias record to the Regional API Gateway stage endpoint. Import the public certificate associated with the company's domain name into AWS Certificate Manager (ACM) in the us-east-1 Region.

C. Create a Regional API Gateway endpoint. Associate the API Gateway endpoint with the company's domain name. Import the public certificate associated with the company's domain name into AWS Certificate Manager (ACM) in the same Region. Attach the certificate to the API Gateway endpoint. Configure Route 53 to route traffic to the API Gateway endpoint.

D. Create a Regional API Gateway endpoint. Associate the API Gateway endpoint with the company's domain name. Import the public certificate associated with the company's domain name into AWS Certificate Manager (ACM) in the us-east-1 Region. Attach the certificate to the API Gateway APIs. Create Route 53 DNS records with the company's domain name. Point an A record to the company's domain name.

Answer: C

Explanation:

The correct answer is C because it aligns with the best practices for using custom domain names with API Gateway and ensuring secure HTTPS communication. Here's why:

Regional API Gateway Endpoint: Creating a Regional API Gateway endpoint is essential. It allows associating a custom domain name and certificate within a specific AWS Region (ca-central-1 in this case), providing lower latency and better control compared to edge-optimized endpoints in this scenario.

Custom Domain Association: Associating the API Gateway endpoint with the company's domain name is a key step in providing a user-friendly and branded URL for the APIs. This is the fundamental step toward using the company's domain instead of the default API Gateway URL.

ACM Certificate Import and Region Specificity: Importing the SSL/TLS certificate into AWS Certificate Manager (ACM) in the same Region as the API Gateway (ca-central-1) is crucial. ACM certificates are region-specific, and the API Gateway needs to be able to access the certificate to establish secure HTTPS connections. Answer D fails on this very important concept.

Certificate Attachment: Attaching the certificate to the API Gateway endpoint ensures that the API Gateway uses the certificate for SSL/TLS termination, enabling HTTPS for clients accessing the APIs.

Route 53 Configuration: Configuring Route 53 to route traffic to the API Gateway endpoint is the final step.
Route 53 maps the company's domain name to the API Gateway endpoint, ensuring that requests to the custom domain name are directed to the API Gateway.Option A is incorrect as stage variables are generally for API Gateway configuration values, not for custom domain mapping. Option B is incorrect because, while pointing a Route 53 record to an API Gateway endpoint is correct, the ACM certificate must be in the same Region as the API Gateway. Option D incorrectly states that ACM certificates need to be in us-east-1. They need to be in the same region as the API Gateway. It also incorrectly states that the certificate needs to be attached to the API Gateway APIs (plural). The certificate is attached to the API Gateway endpoint.

Supporting Links:

Custom Domain Names for API Gateway:

https://docs.aws.amazon.com/apigateway/latest/developerguide/apigateway-use-custom-domain-name.html **AWS Certificate Manager (ACM):**https://aws.amazon.com/certificate-manager/
Amazon Route 53:https://aws.amazon.com/route53/

Question: 57

A company is running a popular social media website. The website gives users the ability to upload images to share with other users. The company wants to make sure that the images do not contain inappropriate content. The company needs a solution that minimizes development effort.
What should a solutions architect do to meet these requirements?

A. Use Amazon Comprehend to detect inappropriate content. Use human review for low-confidence predictions.

B. Use Amazon Rekognition to detect inappropriate content. Use human review for low-confidence predictions.

C. Use Amazon SageMaker to detect inappropriate content. Use ground truth to label low-confidence predictions.

D. Use AWS Fargate to deploy a custom machine learning model to detect inappropriate content. Use ground truth to label low-confidence predictions.

Answer: B

Explanation:

The correct answer is B because it leverages Amazon Rekognition's pre-trained models specifically designed for image analysis, including the detection of explicit or suggestive content. This significantly reduces development effort compared to building a custom model.

Amazon Rekognition's Content Moderation feature can identify various types of inappropriate content, such as nudity, violence, and hate symbols. The service provides a confidence score for each detection. For detections with low confidence scores, a human review workflow can be implemented to ensure accuracy and prevent false positives or negatives. This hybrid approach balances automation with human oversight.

Option A is incorrect because Amazon Comprehend is a natural language processing (NLP) service, best suited for analyzing text, not images. Option C is incorrect because Amazon SageMaker is a machine learning platform for building, training, and deploying custom models. While it's possible to build a custom image moderation model with SageMaker, it requires significantly more development effort than using Rekognition's pre-built capabilities. Ground Truth is used for data labeling in SageMaker and isn't the optimal choice when pre-trained models with confidence scores and built in human review workflows are available. Option D is incorrect because deploying a custom machine learning model on AWS Fargate also requires significant development and operational overhead compared to utilizing a managed service like Rekognition. Fargate provides serverless compute for containers, not a machine learning image analysis service. Building custom models and infrastructure is unnecessary when a suitable managed service already exists. Rekognition provides a more straightforward and efficient solution for the stated requirements.Authoritative links:

Amazon Rekognition Content Moderation: https://aws.amazon.com/rekognition/content-moderation/

Question: 58

A company wants to run its critical applications in containers to meet requirements for scalability and availability.
The company prefers to focus on maintenance of the critical applications. The company does not want to be responsible for provisioning and managing the underlying infrastructure that runs the containerized workload. What should a solutions architect do to meet these requirements?

A. Use Amazon EC2 instances, and install Docker on the instances.

B. Use Amazon Elastic Container Service (Amazon ECS) on Amazon EC2 worker nodes.

C. Use Amazon Elastic Container Service (Amazon ECS) on AWS Fargate.

D. Use Amazon EC2 instances from an Amazon Elastic Container Service (Amazon ECS)-optimized Amazon Machine Image (AMI).

Answer: C

Explanation:

The best solution is to use Amazon ECS on AWS Fargate because it abstracts away the need to manage the underlying infrastructure. Fargate is a serverless compute engine for containers that lets you run containers without managing servers or clusters. This aligns perfectly with the company's preference to focus on application maintenance and avoid infrastructure management.

Option A requires manual installation and configuration of Docker on EC2 instances, placing the burden of server maintenance and scaling on the company. Option B, while using ECS, still involves managing EC2 worker nodes. This means patching, scaling, and ensuring the availability of the underlying EC2 instances. Option D, using an ECS-optimized AMI on EC2, simplifies some aspects of EC2 management, but the company is still responsible for the EC2 instances themselves.

Fargate eliminates these responsibilities by handling the provisioning, scaling, and patching of the infrastructure. The company simply defines the container image, CPU, and memory requirements, and Fargate handles the rest. This reduces operational overhead and allows the company to focus solely on their critical applications. Furthermore, Fargate integrates well with other AWS services and offers cost optimization benefits.https://aws.amazon.com/fargate/https://aws.amazon.com/ecs/

Question: 59

A company hosts more than 300 global websites and applications. The company requires a platform to analyze more than 30 TB of clickstream data each day.
What should a solutions architect do to transmit and process the clickstream data?

A. Design an AWS Data Pipeline to archive the data to an Amazon S3 bucket and run an Amazon EMR cluster with the data to generate analytics.

B. Create an Auto Scaling group of Amazon EC2 instances to process the data and send it to an Amazon S3 data lake for Amazon Redshift to use for analysis.

C. Cache the data to Amazon CloudFront. Store the data in an Amazon S3 bucket. When an object is added to the S3 bucket. run an AWS Lambda function to process the data for analysis.

D. Collect the data from Amazon Kinesis Data Streams. Use Amazon Kinesis Data Firehose to transmit the data to an Amazon S3 data lake. Load the data in Amazon Redshift for analysis.

Answer: D

Explanation:

The most effective solution for handling 30 TB of daily clickstream data from 300+ global websites and applications for analysis involves a scalable and robust data ingestion and processing pipeline using AWS's specialized services for big data. Option D is the best approach.

Here's a breakdown:

Data Ingestion (Amazon Kinesis Data Streams): Kinesis Data Streams is designed for real-time ingestion of high-volume data streams. It can handle the constant flow of clickstream data from numerous sources globally. It allows the data to be buffered and processed in manageable chunks.

Data Delivery (Amazon Kinesis Data Firehose): Kinesis Data Firehose is ideal for reliably loading data into data lakes and data warehouses. It can automatically scale to match the incoming data volume and deliver data to S3. It offers data transformation and batching capabilities.

Data Lake (Amazon S3): S3 provides a scalable and cost-effective storage solution for the large volume of clickstream data. It serves as a central repository for raw and processed data, forming the foundation of the data lake.

Data Warehousing (Amazon Redshift): Redshift is a fast, fully managed, petabyte-scale data warehouse service in the cloud. It is optimized for analytical queries and can efficiently analyze the large dataset stored in S3.

Option A is less desirable because while EMR can process large datasets, using Data Pipeline solely for archiving and triggering EMR jobs introduces unnecessary complexity. Data Pipeline is typically used for orchestration, and Kinesis Data Firehose is more streamlined for data delivery to S3. Option B involves managing EC2 instances, which adds operational overhead compared to using managed services like Kinesis and Redshift. While caching data with CloudFront, as suggested in Option C, can improve website
performance, it doesn't directly address the core problem of analyzing 30 TB of clickstream data daily. A Lambda function triggered by S3 objects may struggle with the sheer volume of data.

In summary, using Kinesis Data Streams and Firehose ensures real-time data ingestion and delivery to a data lake (S3), where it can be efficiently analyzed by Redshift, offering a scalable, managed, and cost-effective solution.

Relevant links:

Amazon Kinesis Data Streams:https://aws.amazon.com/kinesis/data-streams/
Amazon Kinesis Data Firehose:https://aws.amazon.com/kinesis/data-firehose/
Amazon S3:https://aws.amazon.com/s3/
Amazon Redshift:https://aws.amazon.com/redshift/

Question: 60

A company has a website hosted on AWS. The website is behind an Application Load Balancer (ALB) that is configured to handle HTTP and HTTPS separately. The company wants to forward all requests to the website so that the requests will use HTTPS.
What should a solutions architect do to meet this requirement?

A. Update the ALB's network ACL to accept only HTTPS traffic.
B. Create a rule that replaces the HTTP in the URL with HTTPS.
C. Create a listener rule on the ALB to redirect HTTP traffic to HTTPS.
D. Replace the ALB with a Network Load Balancer configured to use Server Name Indication (SNI).

Answer: C

Explanation:

The correct solution is to create a listener rule on the Application Load Balancer (ALB) to redirect HTTP traffic to HTTPS (Option C). Here's why:

An ALB listens for incoming traffic on specified ports and protocols. To redirect HTTP traffic to HTTPS, you configure a listener on port 80 (HTTP) and add a rule to redirect all incoming requests to port 443 (HTTPS). This is a standard practice for ensuring secure communication.

Option A (Updating the ALB's network ACL) is incorrect because network ACLs operate at the subnet level and control inbound and outbound traffic based on IP addresses and ports. While you can restrict HTTP traffic at the network ACL level, this would block HTTP traffic entirely rather than redirecting it to HTTPS. Network

ACLs are not designed for URL redirection.

Option B (Creating a rule to replace HTTP in the URL with HTTPS) is incorrect because ALBs don't directly manipulate URLs in that way. ALBs can perform content-based routing, but not URL rewriting in the context described. The primary function of the ALB in this scenario is to redirect traffic based on listener rules.

Option D (Replacing the ALB with a Network Load Balancer) is incorrect because Network Load Balancers (NLBs) operate at Layer 4 (TCP/UDP), while ALBs operate at Layer 7 (Application Layer). NLBs do not have the capability to inspect HTTP headers or redirect traffic based on the HTTP protocol. Server Name Indication (SNI) on NLBs handles TLS certificates for multiple domains but does not redirect HTTP to HTTPS. NLBs are typically used for high-performance, low-latency applications where protocol-level inspection is not required. An ALB is better suited for HTTP/HTTPS traffic management and URL redirection.

Therefore, the most efficient and appropriate method is to use an ALB listener rule to redirect HTTP traffic to HTTPS, ensuring all communication with the website is secure.

Further Research:

AWS Documentation on ALB Listeners:
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-listeners.html **AWS Documentation on ALB Rules:**
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/load-balancer-rules.html
AWS Documentation on Network ACLs:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-network-acls.html
AWS Documentation on Network Load Balancers:
https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html

Question: 61

A company is developing a two-tier web application on AWS. The company's developers have deployed the application on an Amazon EC2 instance that connects directly to a backend Amazon RDS database. The company must not hardcode database credentials in the application. The company must also implement a solution to automatically rotate the database credentials on a regular basis.
Which solution will meet these requirements with the LEAST operational overhead?

A. Store the database credentials in the instance metadata. Use Amazon EventBridge (Amazon CloudWatch Events) rules to run a scheduled AWS Lambda function that updates the RDS credentials and instance metadata at the same time.

B. Store the database credentials in a configuration file in an encrypted Amazon S3 bucket. Use Amazon EventBridge (Amazon CloudWatch Events) rules to run a scheduled AWS Lambda function that updates the RDS credentials and the credentials in the configuration file at the same time. Use S3 Versioning to ensure the ability to fall back to previous values.

C. Store the database credentials as a secret in AWS Secrets Manager. Turn on automatic rotation for the secret. Attach the required permission to the EC2 role to grant access to the secret.

D. Store the database credentials as encrypted parameters in AWS Systems Manager Parameter Store. Turn on automatic rotation for the encrypted parameters. Attach the required permission to the EC2 role to grant access to the encrypted parameters.

Answer: C

Explanation:

Option C is the most efficient and secure solution because it leverages AWS Secrets Manager for credential management and rotation.

Here's why:

1. **Secrets Manager:** AWS Secrets Manager is specifically designed for securely storing and managing sensitive information like database credentials. It eliminates the need to hardcode credentials in applications.

2. **Automatic Rotation:** Secrets Manager provides automatic secret rotation, significantly reducing the operational overhead associated with manually managing credentials. The service handles the complexities of updating the database and the stored secret.

3. **IAM Integration:** IAM roles assigned to EC2 instances can be granted permissions to access specific secrets stored in Secrets Manager. This allows the application running on the EC2 instance to retrieve the database credentials securely.

4. **Least Operational Overhead:** Solutions A, B, and D require custom Lambda functions and scheduling mechanisms to manage credential rotation. Option C automates this process, which results in the lowest operational overhead because Secrets Manager handles the complexity of rotating the credentials.

5. **Security Best Practices:** Using Secrets Manager aligns with security best practices by centralizing and securing sensitive information and using encryption.

Why other options are not ideal:

Option A: Storing credentials in instance metadata is not recommended for sensitive information like database credentials. Also, instance metadata is not designed for automated rotation.

Option B: Storing credentials in S3, even encrypted, is a less secure and operationally heavier approach than using Secrets Manager. Implementing the rotation logic and managing S3 versions introduces unnecessary complexity.

Option D: Parameter Store can store secrets, but automatic rotation of encrypted parameters requires custom configuration and Lambda functions. Secrets Manager is a more streamlined and secure solution for this specific use case.

Here are some authoritative links for further research:

AWS Secrets Manager Documentation
Rotating AWS Secrets Manager secrets automatically
AWS Identity and Access Management (IAM)

Question: 62

A company is deploying a new public web application to AWS. The application will run behind an Application Load Balancer (ALB). The application needs to be encrypted at the edge with an SSL/TLS certificate that is issued by an external certificate authority (CA). The certificate must be rotated each year before the certificate expires. What should a solutions architect do to meet these requirements?

A. Use AWS Certificate Manager (ACM) to issue an SSL/TLS certificate. Apply the certificate to the ALB. Use the managed renewal feature to automatically rotate the certificate.

B. Use AWS Certificate Manager (ACM) to issue an SSL/TLS certificate. Import the key material from the certificate. Apply the certificate to the ALUse the managed renewal feature to automatically rotate the certificate.

C. Use AWS Certificate Manager (ACM) Private Certificate Authority to issue an SSL/TLS certificate from the root CA. Apply the certificate to the ALB. Use the managed renewal feature to automatically rotate the certificate.

D. Use AWS Certificate Manager (ACM) to import an SSL/TLS certificate. Apply the certificate to the ALB. Use Amazon EventBridge (Amazon CloudWatch Events) to send a notification when the certificate is nearing expiration. Rotate the certificate manually.

Answer: D

Explanation:

The correct answer is D because it addresses the specific requirements of using a certificate issued by an external CA and the need to manually rotate it. Let's break down why the other options are incorrect and why D is the best fit:

A, B, and C are incorrect: These options involve using ACM to issue a certificate. The question explicitly states that the certificate is issued by an external CA. ACM's managed renewal feature only works for ACM-issued certificates. Therefore, relying on ACM to issue and automatically rotate the certificate is not possible when using an externally issued certificate. Option B contains the erroneous step of importing key material from a certificate (a certificate is the key material). Option C involves a private certificate authority, an unnecessary level of complexity when an existing public CA is being used.

D is correct: This option acknowledges the need to import the externally issued certificate into ACM. ACM serves as a central repository for managing certificates, regardless of where they are issued. Once imported, the certificate can be associated with the ALB. Because the certificate is externally issued, ACM cannot automatically renew it. Therefore, the solution leverages Amazon EventBridge (formerly CloudWatch Events) to monitor the certificate's expiration date. When the certificate is nearing expiry, EventBridge triggers a notification (e.g., via SNS to an operations team), prompting a manual rotation process. This process involves obtaining a new certificate from the external CA, importing it into ACM, and updating the ALB to use the new certificate.

In summary, option D correctly acknowledges the externally issued certificate, utilizes ACM for management, and implements a notification system to ensure manual rotation occurs before the certificate expires, thus meeting all requirements.

Relevant links:

AWS Certificate Manager (ACM): https://aws.amazon.com/certificate-manager/
Importing Certificates into ACM: https://docs.aws.amazon.com/acm/latest/userguide/import-certificate.html
Amazon EventBridge: https://aws.amazon.com/eventbridge/
Application Load Balancer (ALB): https://aws.amazon.com/elasticloadbalancing/application-load-balancer/

Question: 63

A company runs its infrastructure on AWS and has a registered base of 700,000 users for its document management application. The company intends to create a product that converts large .pdf files to .jpg image files. The .pdf files average 5 MB in size. The company needs to store the original files and the converted files. A solutions architect must design a scalable solution to accommodate demand that will grow rapidly over time. Which solution meets these requirements MOST cost-effectively?

A. Save the .pdf files to Amazon S3. Configure an S3 PUT event to invoke an AWS Lambda function to convert the files to .jpg format and store them back in Amazon S3.

B. Save the .pdf files to Amazon DynamoDUse the DynamoDB Streams feature to invoke an AWS Lambda function to convert the files to .jpg format and store them back in DynamoDB.

C. Upload the .pdf files to an AWS Elastic Beanstalk application that includes Amazon EC2 instances, Amazon Elastic Block Store (Amazon EBS) storage, and an Auto Scaling group. Use a program in the EC2 instances to convert the files to .jpg format. Save the .pdf files and the .jpg files in the EBS store.

D. Upload the .pdf files to an AWS Elastic Beanstalk application that includes Amazon EC2 instances, Amazon Elastic File System (Amazon EFS) storage, and an Auto Scaling group. Use a program in the EC2 instances to convert the file to .jpg format. Save the .pdf files and the .jpg files in the EBS store.

Answer: A

Explanation:

Here's why option A is the most cost-effective solution for the PDF to JPG conversion problem, along with supporting justifications and links:

Option A leverages the strengths of Amazon S3 and AWS Lambda for a serverless and scalable solution. S3 provides highly durable and cost-effective object storage for both the original PDFs and the converted JPGs. Its event notification system allows triggering the Lambda function directly upon PDF upload, automating the conversion process. Lambda offers pay-per-execution pricing, meaning you only pay for the compute time used during the conversion, making it exceptionally cost-efficient for variable workloads. This serverless approach eliminates the need to manage EC2 instances, reducing operational overhead and associated costs.

Option B is incorrect because DynamoDB is primarily designed for fast key-value or document data storage and retrieval, not for storing large binary files like PDFs and JPGs. Storing these files in DynamoDB would be significantly more expensive than using S3, and the read/write capacity units required would add to cost unnecessarily. Also, using DynamoDB streams for processing is less efficient than S3 event notifications in this scenario.

Options C and D involve using Elastic Beanstalk with EC2 instances and EBS/EFS for storage. While these options could work, they are less cost-effective because they require maintaining EC2 instances even during periods of low demand. EBS, being block storage, is primarily for persistent storage attached to EC2 instances, while EFS is better suited for shared file systems. Neither are as economical for storing large numbers of files as S3. The added operational complexity of managing EC2 instances, Auto Scaling, and storage volumes further increases the total cost.

In summary, the serverless architecture provided by S3 and Lambda offers the best combination of scalability, automation, and cost-effectiveness for this specific PDF-to-JPG conversion task.

Supporting Links:

Amazon S3:https://aws.amazon.com/s3/
AWS Lambda:https://aws.amazon.com/lambda/
S3 Event Notifications:https://docs.aws.amazon.com/AmazonS3/latest/userguide/event-notifications-overview.html
AWS Lambda Pricing:https://aws.amazon.com/lambda/pricing/

Question: 64

A company has more than 5 TB of file data on Windows file servers that run on premises. Users and applications interact with the data each day.
The company is moving its Windows workloads to AWS. As the company continues this process, the company requires access to AWS and on-premises file storage with minimum latency. The company needs a solution that minimizes operational overhead and requires no significant changes to the existing file access patterns. The company uses an AWS Site-to-Site VPN connection for connectivity to AWS.
What should a solutions architect do to meet these requirements?

A. Deploy and configure Amazon FSx for Windows File Server on AWS. Move the on-premises file data to FSx for Windows File Server. Reconfigure the workloads to use FSx for Windows File Server on AWS.

B. Deploy and configure an Amazon S3 File Gateway on premises. Move the on-premises file data to the S3 File Gateway. Reconfigure the on-premises workloads and the cloud workloads to use the S3 File Gateway.

C. Deploy and configure an Amazon S3 File Gateway on premises. Move the on-premises file data to Amazon S3. Reconfigure the workloads to use either Amazon S3 directly or the S3 File Gateway. depending on each workload's location.

D. Deploy and configure Amazon FSx for Windows File Server on AWS. Deploy and configure an Amazon FSx File Gateway on premises. Move the on-premises file data to the FSx File Gateway. Configure the cloud workloads to use FSx for Windows File Server on AWS. Configure the on-premises workloads to use the FSx File Gateway.

Answer: D

Explanation:

The correct answer is D because it directly addresses all requirements with a balanced approach to performance, operational overhead, and minimal disruption. Here's a breakdown:

Minimum Latency: Deploying FSx File Gateway on-premises allows the on-premises workloads to access the file data stored in FSx for Windows File Server in AWS with minimal latency. The FSx File Gateway acts as a local cache for frequently accessed files, ensuring quick access for on-premises users and applications.

Access to AWS and On-Premises: FSx for Windows File Server on AWS stores the data in the cloud, providing access to cloud workloads. The FSx File Gateway provides access to the same data for on-premises workloads.

Minimized Operational Overhead: FSx File Gateway is a fully managed service, reducing the operational burden of managing storage infrastructure on-premises. The management of the data is centralized on AWS.

No Significant Changes to File Access Patterns: Both FSx for Windows File Server and FSx File Gateway support the standard SMB protocol, allowing existing Windows applications to access files without significant code modifications.

Why other options are incorrect:

A is incorrect because moving all the data to AWS and accessing it over Site-to-Site VPN would likely increase the latency for on-premises workloads.

B and C are incorrect because S3 File Gateway replicates data to Amazon S3, which is object storage and doesn't natively support SMB protocol. Thus, the workloads would need to be reconfigured to use S3 APIs, which would be a significant change to existing file access patterns, and S3 File Gateway on-premises still might increase the latency for on-premises workloads compared to a dedicated solution like FSx File Gateway.

Here are some authoritative links for further research:

Amazon FSx for Windows File Server:https://aws.amazon.com/fsx/windows/
Amazon FSx File Gateway:https://aws.amazon.com/fsx/file-gateway/ **Amazon S3 File Gateway:**https://aws.amazon.com/storagegateway/file/

Question: 65

A hospital recently deployed a RESTful API with Amazon API Gateway and AWS Lambda. The hospital uses API Gateway and Lambda to upload reports that are in PDF format and JPEG format. The hospital needs to modify the Lambda code to identify protected health information (PHI) in the reports.
Which solution will meet these requirements with the LEAST operational overhead?

A. Use existing Python libraries to extract the text from the reports and to identify the PHI from the extracted text.

B. Use Amazon Textract to extract the text from the reports. Use Amazon SageMaker to identify the PHI from the extracted text.

C. Use Amazon Textract to extract the text from the reports. Use Amazon Comprehend Medical to identify the PHI from the extracted text.

D. Use Amazon Rekognition to extract the text from the reports. Use Amazon Comprehend Medical to identify the PHI from the extracted text.

Answer: C

Explanation:

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

The core requirement is identifying PHI (Protected Health Information) within PDF and JPEG reports uploaded via API Gateway and Lambda. The solution must minimize operational overhead.

Option A (Python Libraries): While feasible, this approach requires significant custom coding to handle PDF and JPEG parsing (potentially complex) and implementing a PHI identification algorithm. This increases development effort, testing requirements, and ongoing maintenance, leading to higher operational overhead. **Option B (Textract and SageMaker):** Textract is excellent for text extraction. However, using SageMaker for PHI identification introduces unnecessary complexity. SageMaker is designed for building, training, and deploying custom machine learning models. For a specialized task like PHI detection (which is already addressed by a dedicated service), SageMaker is overkill and introduces significant operational overhead related to model management, infrastructure, and training data.

Option C (Textract and Comprehend Medical): Textract efficiently extracts text from both PDF and JPEG formats. Comprehend Medical is a HIPAA-eligible natural language processing (NLP) service specifically designed for analyzing medical text and identifying PHI. This combination minimizes custom code and leverages AWS-managed services, resulting in the least operational overhead. It offers built-in PHI detection capabilities.

Option D (Rekognition and Comprehend Medical): Rekognition is primarily designed for image analysis, including object detection and facial recognition. While it can extract text through OCR (Optical Character Recognition), Textract is optimized for document processing and will yield better results with PDF and complex document layouts. Also, Rekognition might not be as accurate as Textract in extracting text from these specific report formats. Comprehend Medical is suitable for the PHI extraction task, but Rekognition is not the ideal choice for text extraction in this scenario, creating additional overhead.

Therefore, using Amazon Textract to extract text and Amazon Comprehend Medical to identify PHI offers the best balance of functionality, accuracy, and minimal operational burden by utilizing purpose-built, managed AWS services tailored to these specific tasks. This solution aligns with the "less code is better" principle and simplifies the deployment and maintenance processes.

Authoritative Links:

Amazon Textract:https://aws.amazon.com/textract/
Amazon Comprehend Medical:https://aws.amazon.com/comprehend/medical/
Amazon SageMaker:https://aws.amazon.com/sagemaker/
Amazon Rekognition:https://aws.amazon.com/rekognition/

Question: 66

A company has an application that generates a large number of files, each approximately 5 MB in size. The files are stored in Amazon S3. Company policy requires the files to be stored for 4 years before they can be deleted. Immediate accessibility is always required as the files contain critical business data that is not easy to reproduce. The files are frequently accessed in the first 30 days of the object creation but are rarely accessed after the first 30 days.
Which storage solution is MOST cost-effective?

A. Create an S3 bucket lifecycle policy to move files from S3 Standard to S3 Glacier 30 days from object creation. Delete the files 4 years after object creation.

B. Create an S3 bucket lifecycle policy to move files from S3 Standard to S3 One Zone-Infrequent Access (S3 One Zone-IA) 30 days from object creation. Delete the files 4 years after object creation.

C. Create an S3 bucket lifecycle policy to move files from S3 Standard to S3 Standard-Infrequent Access (S3 Standard-IA) 30 days from object creation. Delete the files 4 years after object creation.

D. Create an S3 bucket lifecycle policy to move files from S3 Standard to S3 Standard-Infrequent Access (S3

Standard-IA) 30 days from object creation. Move the files to S3 Glacier 4 years after object creation.

Answer: C

Explanation:

The correct answer is C because it offers the most cost-effective solution while adhering to the stated requirements of immediate accessibility and long-term retention.

Here's a detailed breakdown:

S3 Standard: Initially, files are stored in S3 Standard for the first 30 days. This ensures fast and frequent access when the files are actively used. S3 Standard provides high availability (99.99%) and durability (99.999999999%) for frequently accessed data.

S3 Standard-IA: After 30 days, the lifecycle policy automatically transitions the files to S3 Standard-IA. This storage class is designed for data that is infrequently accessed but requires rapid retrieval when needed. It offers lower storage costs compared to S3 Standard while maintaining similar performance characteristics. Since the question specifies immediate accessibility is always required, S3 Standard-IA is a suitable choice for infrequently accessed data.

4-Year Retention & Deletion: The lifecycle policy is configured to permanently delete the files after 4 years, fulfilling the company's retention policy.

Now, let's examine why the other options are less optimal:

Option A (S3 Glacier): Moving files directly to S3 Glacier after 30 days is not ideal. S3 Glacier is designed for archival storage and retrieval can take several hours. The requirement of "immediate accessibility" makes Glacier an unsuitable option for data that may need to be accessed at any moment.

Option B (S3 One Zone-IA): While S3 One Zone-IA offers lower storage costs than S3 Standard-IA, it stores data in a single Availability Zone. This exposes the data to a higher risk of data loss in the event of an Availability Zone failure. S3 One Zone-IA is appropriate when data is easily reproducible. However, the question states that data is not easy to reproduce.

Option D (S3 Standard-IA followed by S3 Glacier): Moving to Glacier after 4 years is unnecessary, since deletion is required at that point. It adds complexity and cost without providing any benefit. Moreover, files can be stored in S3 Standard-IA for 4 years and then be deleted.

In summary, option C provides the best balance between cost, accessibility, and durability, meeting all of the company's requirements in the most cost-effective way. Using S3 Standard for immediate needs, transitioning to S3 Standard-IA for infrequent access, and deleting after 4 years aligns with the best practices for S3 lifecycle management.

Authoritative Links:

Amazon S3 Storage Classes:https://aws.amazon.com/s3/storage-classes/
S3 Lifecycle Management:https://docs.aws.amazon.com/AmazonS3/latest/userguide/lifecycle-configuration-concept.html

Question: 67

A company hosts an application on multiple Amazon EC2 instances. The application processes messages from an Amazon SQS queue, writes to an Amazon RDS table, and deletes the message from the queue. Occasional duplicate records are found in the RDS table. The SQS queue does not contain any duplicate messages. What should a solutions architect do to ensure messages are being processed once only?

A. Use the CreateQueue API call to create a new queue.

B. Use the AddPermission API call to add appropriate permissions.

C. Use the ReceiveMessage API call to set an appropriate wait time.

D. Use the ChangeMessageVisibility API call to increase the visibility timeout.

Answer: D

Explanation:

The most appropriate solution to address the issue of duplicate records in the RDS table, despite the absence of duplicate messages in the SQS queue, is **D. Use the ChangeMessageVisibility API call to increase the visibility timeout.**

Here's a detailed justification:

The problem indicates that the same message is being processed multiple times, leading to duplicate entries in the RDS database. Since the queue itself isn't the source of duplicates, the issue likely lies in how the EC2 instances are handling message processing and acknowledging completion to SQS.

SQS employs a visibility timeout mechanism. When an EC2 instance retrieves a message from the queue, the message becomes invisible to other consumers for a specified duration (the visibility timeout). This prevents other instances from processing the same message simultaneously. The receiving instance is then expected to delete the message from the queue upon successful processing.

If an EC2 instance fails to process a message within the visibility timeout (e.g., due to a crash, network issue, or slow processing), the message becomes visible again in the queue. This allows another instance (or even the original instance after recovery) to pick up the message and re-process it, hence leading to duplicates.

Increasing the visibility timeout using the ChangeMessageVisibility API call provides more time for the EC2 instance to process the message and delete it from the queue before it becomes visible again. This reduces the likelihood of another instance grabbing the same message and creating a duplicate record in the RDS table.

Options A, B, and C are not relevant to addressing the identified problem.

A (CreateQueue): Creating a new queue doesn't solve the underlying issue of messages being processed multiple times.

B (AddPermission): Adding permissions is about access control, not about preventing duplicate processing. **C (ReceiveMessage Wait Time):** While adjusting wait time can improve polling efficiency, it doesn't prevent duplicate processing if an instance fails to process a message within the visibility timeout.

By increasing the visibility timeout to a value that comfortably exceeds the maximum expected processing time for a message, the solution architect can minimize the risk of duplicate processing and ensure messages are processed only once. In cases where processing time can vary greatly, a combination of increased visibility timeout and implementing idempotent message processing logic in the application can provide a robust solution against duplicates. Idempotency ensures that even if a message is processed multiple times, the result is the same as processing it once, preventing data inconsistencies.Consider using Dead Letter Queues to manage message that exceeds retry count.

Relevant Documentation:

Amazon SQS Visibility Timeout:
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-visibility-timeout.html
Amazon SQS ChangeMessageVisibility:
https://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/API_ChangeMessageVisibility.html

Question: 68

A solutions architect is designing a new hybrid architecture to extend a company's on-premises infrastructure to AWS. The company requires a highly available connection with consistent low latency to an AWS Region. The company needs to minimize costs and is willing to accept slower traffic if the primary connection fails. What should the solutions architect do to meet these requirements?

A. Provision an AWS Direct Connect connection to a Region. Provision a VPN connection as a backup if the primary Direct Connect connection fails.

B. Provision a VPN tunnel connection to a Region for private connectivity. Provision a second VPN tunnel for private connectivity and as a backup if the primary VPN connection fails.

C. Provision an AWS Direct Connect connection to a Region. Provision a second Direct Connect connection to the same Region as a backup if the primary Direct Connect connection fails.

D. Provision an AWS Direct Connect connection to a Region. Use the Direct Connect failover attribute from the AWS CLI to automatically create a backup connection if the primary Direct Connect connection fails.

Answer: A

Explanation:

The correct answer is A because it provides a cost-effective solution for high availability and consistent low latency. Direct Connect (DX) offers a dedicated network connection from the on-premises environment to AWS, fulfilling the requirement for high availability and consistent low latency under normal circumstances. This dedicated connection bypasses the public internet, resulting in more predictable and lower latency than VPN connections.

The requirement to minimize costs while accepting slower traffic during failover is met by using a VPN connection as a backup. VPN connections are established over the internet and are generally less expensive than a redundant Direct Connect connection. When the primary Direct Connect link fails, traffic can be routed through the VPN connection, ensuring business continuity, although at a reduced speed and potentially higher latency.

Option B is incorrect because VPN connections, while cost-effective, do not offer the consistently low latency required during normal operation. VPN connections are susceptible to internet traffic fluctuations.

Option C is incorrect as it proposes a redundant Direct Connect connection. While this would offer higher availability, it significantly increases costs. The company has already stated a willingness to accept slower traffic if the primary connection fails, making the cost of a second Direct Connect circuit unnecessary.

Option D is incorrect because the Direct Connect failover attribute within the AWS CLI doesn't automatically create a backup connection. The Direct Connect failover attribute allows you to influence which path traffic takes but does not instantiate backup connections. It influences the traffic routing, given the primary connection is already established. The failover attribute helps prioritize connections and adjust routing preferences but will not create connections from scratch.

In conclusion, a primary Direct Connect connection coupled with a VPN as a fallback mechanism delivers the best balance between high performance and cost efficiency, aligning perfectly with the company's requirements.

Further Research:

AWS Direct Connect: https://aws.amazon.com/directconnect/ AWS
VPN: https://aws.amazon.com/vpn/

Question: 69

A company is running a business-critical web application on Amazon EC2 instances behind an Application Load Balancer. The EC2 instances are in an Auto Scaling group. The application uses an Amazon Aurora PostgreSQL database that is deployed in a single Availability Zone. The company wants the application to be highly available with minimum downtime and minimum loss of data.
Which solution will meet these requirements with the LEAST operational effort?

A. Place the EC2 instances in different AWS Regions. Use Amazon Route 53 health checks to redirect traffic. Use Aurora PostgreSQL Cross-Region Replication.

B. Configure the Auto Scaling group to use multiple Availability Zones. Configure the database as Multi-AZ. Configure an Amazon RDS Proxy instance for the database.

C. Configure the Auto Scaling group to use one Availability Zone. Generate hourly snapshots of the database. Recover the database from the snapshots in the event of a failure.

D. Configure the Auto Scaling group to use multiple AWS Regions. Write the data from the application to Amazon S3. Use S3 Event Notifications to launch an AWS Lambda function to write the data to the database.

Answer: B

Explanation:

Option B is the most appropriate solution because it achieves high availability for both the application and the database with minimal operational overhead.

Here's a detailed breakdown:

Auto Scaling Group with Multiple Availability Zones (AZs): Spreading EC2 instances across multiple AZs ensures that if one AZ fails, the application remains available in other AZs. This provides redundancy and fault tolerance at the application tier.

Multi-AZ Aurora PostgreSQL: Configuring the database as Multi-AZ creates a synchronous standby replica in a different AZ. If the primary database instance fails, Aurora automatically fails over to the standby, minimizing downtime and data loss. This addresses the high availability requirement for the database.

Amazon RDS Proxy: RDS Proxy is an optional service that further enhances availability and scalability. It manages database connections, reducing the load on the database and protecting it from connection storms. While not strictly required for basic Multi-AZ failover, it improves connection management, especially under heavy load and failover situations.

Other options are less efficient or introduce unnecessary complexity:

Option A (Cross-Region Replication): Using cross-region replication for the database adds complexity and potential latency. It is also overkill since the application is running in a single region. Also, failover between regions generally introduces more downtime than a Multi-AZ failover.

Option C (Hourly Snapshots): Relying on snapshots for recovery introduces a significant recovery time objective (RTO) and recovery point objective (RPO). Data loss is possible up to the last snapshot. This is not suitable for a business-critical application requiring minimal downtime and data loss.

Option D (Multi-Region ASG and S3/Lambda integration): This introduces substantial architectural complexity for data persistence compared to using Aurora Multi-AZ. The S3/Lambda pathway for database writes introduces latency and increased operational overhead.

Therefore, configuring the ASG across multiple AZs, using Aurora Multi-AZ, and leveraging RDS Proxy provide the best balance between high availability, minimal downtime/data loss, and operational simplicity, making it the most suitable option.

Supporting Links:

Amazon Aurora High Availability:https://aws.amazon.com/rds/aurora/features/
Amazon RDS Proxy:https://aws.amazon.com/rds/proxy/
Auto Scaling Groups:https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-groups.html

Question: 70

A company's HTTP application is behind a Network Load Balancer (NLB). The NLB's target group is configured to use an Amazon EC2 Auto Scaling group with multiple EC2 instances that run the web service.
The company notices that the NLB is not detecting HTTP errors for the application. These errors require a manual restart of the EC2 instances that run the web service. The company needs to improve the application's availability without writing custom scripts or code.
What should a solutions architect do to meet these requirements?

A. Enable HTTP health checks on the NLB, supplying the URL of the company's application.

B. Add a cron job to the EC2 instances to check the local application's logs once each minute. If HTTP errors are detected. the application will restart.

C. Replace the NLB with an Application Load Balancer. Enable HTTP health checks by supplying the URL of the company's application. Configure an Auto Scaling action to replace unhealthy instances.

D. Create an Amazon Cloud Watch alarm that monitors the UnhealthyHostCount metric for the NLB. Configure an Auto Scaling action to replace unhealthy instances when the alarm is in the ALARM state.

Answer: C

Explanation:

The correct answer is **C**, which involves replacing the Network Load Balancer (NLB) with an Application Load Balancer (ALB) and configuring HTTP health checks and an Auto Scaling action. Here's why:

NLB Limitations: NLBs operate at Layer 4 (TCP/UDP), meaning they are unaware of the application-layer protocol (HTTP) and cannot interpret HTTP status codes. They can only check if a TCP connection can be established on the configured port, not if the application is functioning correctly.
https://docs.aws.amazon.com/elasticloadbalancing/latest/network/introduction.html

ALB's HTTP Awareness: ALBs, on the other hand, operate at Layer 7 (HTTP/HTTPS), enabling them to perform health checks based on HTTP status codes (e.g., 200 OK, 400 Bad Request, 500 Internal Server Error). This allows the ALB to determine if the application is truly healthy and route traffic accordingly.
https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html

HTTP Health Checks: By configuring HTTP health checks on the ALB with a specific URL, the load balancer will periodically send requests to that URL and verify the HTTP status code returned. If the status code indicates an error, the ALB will mark the instance as unhealthy.

Auto Scaling Integration: The Auto Scaling group can be configured with lifecycle hooks or be configured directly to respond to ALB health check failures. When the ALB marks an instance as unhealthy, Auto Scaling can be configured to automatically replace the unhealthy instance with a new one, thus improving application availability without manual intervention. https://docs.aws.amazon.com/autoscaling/ec2/userguide/as-scaling-process.html

Why other options are wrong:

A: NLBs cannot do HTTP health checks.
B: Requires custom scripting, and the company wants to avoid it. Also, it is less reliable than ALB health checks.
D: While CloudWatch can monitor the NLB, it cannot detect HTTP-level errors due to NLB's Layer 4 operation. Monitoring UnhealthyHostCount may only indicate that an EC2 instance is unreachable at the TCP level, not that the application is failing to respond to HTTP requests.

In summary, switching to an ALB enables HTTP health checks, which allows for the automated detection and replacement of unhealthy instances, fulfilling the company's requirements for improved application availability without custom scripting or code.

Question: 71

A company runs a shopping application that uses Amazon DynamoDB to store customer information. In case of data corruption, a solutions architect needs to design a solution that meets a recovery point objective (RPO) of 15 minutes and a recovery time objective (RTO) of 1 hour.
What should the solutions architect recommend to meet these requirements?

A. Configure DynamoDB global tables. For RPO recovery, point the application to a different AWS Region.

B. Configure DynamoDB point-in-time recovery. For RPO recovery, restore to the desired point in time.

C. Export the DynamoDB data to Amazon S3 Glacier on a daily basis. For RPO recovery, import the data from S3 Glacier to DynamoDB.

D. Schedule Amazon Elastic Block Store (Amazon EBS) snapshots for the DynamoDB table every 15 minutes. For RPO recovery, restore the DynamoDB table by using the EBS snapshot.

Answer: B

Explanation:

The correct answer is B because it aligns with the stated RPO and RTO requirements most effectively and efficiently.

Here's why:

DynamoDB Point-in-Time Recovery (PITR): DynamoDB PITR provides automatic backups of your table data. You can restore the table to any point in time within the past 35 days. This perfectly addresses the need for recovering data to a specific point within the last 15 minutes (RPO). The restoration process also falls within the 1-hour RTO.
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/PointInTimeRecovery.html

Why other options are incorrect:

A. DynamoDB Global Tables: While Global Tables provide replication for high availability and disaster recovery, their primary purpose is not granular point-in-time recovery for data corruption scenarios within a single table. Switching regions might help avoid regional failures, but if the corruption is replicated, it won't solve the problem. They do not meet the specific RPO requirement of 15 minutes for data corruption within a table.

C. Export to S3 Glacier: Exporting data to S3 Glacier is a suitable strategy for long-term archiving and compliance, but it is too slow for a 1-hour RTO. Glacier is designed for infrequently accessed data, and retrieving data from Glacier can take several hours. This does not satisfy the 1-hour RTO requirement.

D. EBS Snapshots for DynamoDB: DynamoDB does not reside on EBS volumes. It is a NoSQL database service that internally handles storage. Taking EBS snapshots won't capture the data or the configuration of your DynamoDB table, and it's not the correct way to back up DynamoDB data.

In summary, DynamoDB PITR is the most appropriate solution as it offers the required granularity for recovery, meeting both the 15-minute RPO and the 1-hour RTO. Other options do not directly address the specific requirements of recovering from data corruption with the given recovery objectives.

Question: 72

A company runs a photo processing application that needs to frequently upload and download pictures from Amazon S3 buckets that are located in the same AWS Region. A solutions architect has noticed an increased cost in data transfer fees and needs to implement a solution to reduce these costs.
How can the solutions architect meet this requirement?

A. Deploy Amazon API Gateway into a public subnet and adjust the route table to route S3 calls through it.

B. Deploy a NAT gateway into a public subnet and attach an endpoint policy that allows access to the S3 buckets.

C. Deploy the application into a public subnet and allow it to route through an internet gateway to access the S3 buckets.

D. Deploy an S3 VPC gateway endpoint into the VPC and attach an endpoint policy that allows access to the S3 buckets.

Answer: D

Explanation:

The correct solution is **D. Deploy an S3 VPC gateway endpoint into the VPC and attach an endpoint policy that allows access to the S3 buckets.**

Here's a detailed justification:

The core problem is the data transfer cost between the application and S3 within the same AWS Region. Data transfer to S3 is generally free. However, data transfer from S3 incurs costs, and these costs increase significantly when the data traverses the public internet.

Options A, B, and C all involve routing traffic via the public internet, defeating the purpose of cost reduction. Deploying API Gateway (A), NAT Gateway (B), or using an Internet Gateway directly (C) means the application is accessing S3 over the internet, incurring outbound data transfer fees. While API Gateway offers other benefits, in this scenario, it adds unnecessary complexity and cost. Similarly, NAT Gateways are for enabling instances in private subnets to access the internet, not to optimize S3 access.

Option D leverages a VPC endpoint, which establishes a direct, private connection between your VPC and S3, without traversing the public internet. A VPC endpoint specifically for S3 creates a gateway within your VPC that allows your application to access S3 buckets as if they were within the same network. This eliminates public internet data transfer costs, as traffic stays within the AWS network.

Furthermore, the endpoint policy attached to the S3 VPC gateway endpoint controls access to the S3 buckets.
This policy specifies which S3 buckets and actions are allowed for resources within the VPC, enhancing security.
It restricts access based on IAM roles, users, and conditions. Without the policy, the endpoint won't function.

Therefore, deploying an S3 VPC gateway endpoint and attaching an endpoint policy is the most efficient and cost-effective way to enable the photo processing application to access S3 buckets without incurring public internet data transfer costs. It keeps the data within the AWS internal network, reducing costs and improving security.

Here are authoritative links for further research:

AWS VPC Endpoints:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints.html
Gateway VPC Endpoints for S3:https://docs.aws.amazon.com/vpc/latest/userguide/vpc-endpoints-s3.html **S3 Pricing:**https://aws.amazon.com/s3/pricing/ (Focus on the "Data Transfer" section)

Question: 73

A company recently launched Linux-based application instances on Amazon EC2 in a private subnet and launched a Linux-based bastion host on an Amazon EC2 instance in a public subnet of a VPC. A solutions architect needs to connect from the on-premises network, through the company's internet connection, to the bastion host, and to the application servers. The solutions architect must make sure that the security groups of all the EC2 instances will allow that access.
Which combination of steps should the solutions architect take to meet these requirements? (Choose two.)

A. Replace the current security group of the bastion host with one that only allows inbound access from the application instances.

B. Replace the current security group of the bastion host with one that only allows inbound access from the internal IP range for the company.

C. Replace the current security group of the bastion host with one that only allows inbound access from the external IP range for the company.

D. Replace the current security group of the application instances with one that allows inbound SSH access from only the private IP address of the bastion host.

E. Replace the current security group of the application instances with one that allows inbound SSH access from only the public IP address of the bastion host.

Answer: CD

Explanation:

The correct answer is CD. Here's why:

C: The bastion host acts as a secure gateway to the application instances. To allow access from the on-premises network, the security group associated with the bastion host must permit inbound traffic from the company's external IP range. This restricts access to only traffic originating from the known and trusted company network, enhancing security. This is because all traffic from on-premise will come through the company's internet connection, which will have a specific, and usually static, external IP range.

D: The application instances in the private subnet should not be directly accessible from the internet. Instead, access is granted via the bastion host. Therefore, the application instances' security group should only allow inbound SSH traffic from the private IP address of the bastion host. This ensures that only traffic originating from the bastion host can connect to the application instances, restricting the attack surface. This implements a layered security approach.

Why the other options are incorrect:

A: Restricting inbound access to the bastion host to only application instances would prevent the on-premises network from accessing it.
B: Allowing the internal IP range of the company would not work, because traffic from the company will be routed through the internet, and therefore come from the external IP range of the company.
E: Using the public IP address of the bastion host in the application instance's security group is less secure because public IP addresses can sometimes change, and more importantly, defeats the purpose of using the bastion host as a secure intermediary, as the application servers are exposed to outside IPs.

Supporting Concepts and Links:

Bastion Host: A server whose purpose is to provide access to a private network from an external network, such as the internet. https://aws.amazon.com/quickstart/architecture/linux-bastion/
Security Groups: Act as a virtual firewall for your EC2 instances to control inbound and outbound traffic.
https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-security-groups.html
Principle of Least Privilege: Granting only the minimum necessary permissions. In this case, restricting access to the bastion host and application instances based on IP addresses aligns with this principle.

Question: 74

A solutions architect is designing a two-tier web application. The application consists of a public-facing web tier hosted on Amazon EC2 in public subnets. The database tier consists of Microsoft SQL Server running on Amazon EC2 in a private subnet. Security is a high priority for the company.
How should security groups be configured in this situation? (Choose two.)

A. Configure the security group for the web tier to allow inbound traffic on port 443 from 0.0.0.0/0.

B. Configure the security group for the web tier to allow outbound traffic on port 443 from 0.0.0.0/0.

C. Configure the security group for the database tier to allow inbound traffic on port 1433 from the security group for the web tier.

D. Configure the security group for the database tier to allow outbound traffic on ports 443 and 1433 to the security group for the web tier.

E. Configure the security group for the database tier to allow inbound traffic on ports 443 and 1433 from the security group for the web tier.

Answer: AC

Explanation:

Here's a detailed justification for why options A and C are the correct choices for configuring security groups in the given two-tier web application scenario:

Option A: Configure the security group for the web tier to allow inbound traffic on port 443 from 0.0.0.0/0.

This configuration allows HTTPS traffic (port 443) from any source (0.0.0.0/0) to reach the web servers. Since the web tier is public-facing, it needs to accept inbound traffic from users on the internet. HTTPS is the standard secure protocol for web communication, encrypting data in transit. Restricting inbound access to port 443 (and potentially port 80 for HTTP redirects to HTTPS) minimizes the attack surface. A more restrictive approach could involve limiting access to specific IP ranges of known users or employing a Web Application Firewall (WAF) for more granular control.

Option C: Configure the security group for the database tier to allow inbound traffic on port 1433 from the security group for the web tier.

This configuration allows traffic on port 1433 (the default port for Microsoft SQL Server) from the web tier's security group to reach the database tier. This is crucial for the web servers to connect to the database server. Instead of specifying IP addresses, using the web tier's security group ensures that only instances within that group can connect to the database. This dynamic association simplifies management, as new web servers added to the web tier automatically gain database access without manually updating IP addresses in the database security group. The database tier should only allow traffic from the web tier and no other sources.

Why other options are incorrect:

Option B: The web tier needs to make outbound calls, but likely not on port 443 to 0.0.0.0/0. The web tier might need outbound access to other AWS services or external APIs, but this would be governed by different rules and likely restricted to specific services or IP ranges rather than the entire internet on a particular port like 443.

Option D and E: The database tier only needs to accept inbound connections from the web tier on port 1433. It does not need to initiate outbound traffic to the web tier. Option E is also incorrect in that it allows inbound traffic on port 443, which is unneeded and increases the attack surface.

Key Cloud Computing Concepts Reinforced:

Security Groups: Acting as virtual firewalls controlling inbound and outbound traffic at the instance level. **Least Privilege Principle:** Granting only the necessary permissions to resources, minimizing potential damage from security breaches.

Two-Tier Architecture: A common application architecture separating the presentation tier (web tier) from the data tier (database tier).

Network Segmentation: Isolating resources (like the database tier in a private subnet) to reduce the blast radius of security incidents.

Authoritative Links for Further Research:

AWS Security Groups:https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html
Security Best Practices in AWS:https://aws.amazon.com/security/

Question: 75

A company wants to move a multi-tiered application from on premises to the AWS Cloud to improve the application's performance. The application consists of application tiers that communicate with each other by way of RESTful services. Transactions are dropped when one tier becomes overloaded. A solutions architect must design a solution that resolves these issues and modernizes the application.
Which solution meets these requirements and is the MOST operationally efficient?

A. Use Amazon API Gateway and direct transactions to the AWS Lambda functions as the application layer. Use Amazon Simple Queue Service (Amazon SQS) as the communication layer between application services.

B. Use Amazon CloudWatch metrics to analyze the application performance history to determine the servers' peak utilization during the performance failures. Increase the size of the application server's Amazon EC2 instances to meet the peak requirements.

C. Use Amazon Simple Notification Service (Amazon SNS) to handle the messaging between application servers running on Amazon EC2 in an Auto Scaling group. Use Amazon CloudWatch to monitor the SNS queue length and scale up and down as required.

D. Use Amazon Simple Queue Service (Amazon SQS) to handle the messaging between application servers running on Amazon EC2 in an Auto Scaling group. Use Amazon CloudWatch to monitor the SQS queue length and scale up when communication failures are detected.

Answer: A

Explanation:

Here's a breakdown of why option A is the best choice and why the others fall short, along with supporting concepts and links.

Option A (API Gateway + Lambda + SQS) offers the most operationally efficient and modern solution for the company's requirements. Let's dissect it:

Amazon API Gateway: Acts as a front door for the application. It allows the company to manage API access, authorization, throttling, and versioning in a centralized and scalable manner. It abstracts the underlying complexities of the backend services. https://aws.amazon.com/api-gateway/

AWS Lambda: Enables the company to modernize its application by leveraging serverless computing. Lambda functions can handle individual requests or transactions without the need to manage servers. This significantly reduces operational overhead and allows for automatic scaling based on demand. Lambda can also integrate directly with API Gateway. https://aws.amazon.com/lambda/

Amazon SQS: Introduces asynchronous communication between the application tiers. When one tier is overloaded, SQS acts as a buffer, preventing transaction drops. The application can decouple its components so that they can fail or go down at any time. SQS provides durability and fault tolerance.
https://aws.amazon.com/sqs/

Why other options are less suitable:

Option B: Increasing EC2 instance sizes (vertical scaling) might alleviate the problem in the short term, but it's not a modern or efficient solution. It does not address the root cause of the issue and does not allow you to decouple components. It involves unnecessary resource allocation even during periods of low demand.

Option C: While SNS can handle messaging, it's primarily designed for push notifications and fanout scenarios. It doesn't provide the same level of queuing and buffering capabilities as SQS, making it less suitable for preventing transaction drops during overload. Also, running application servers on EC2 within an Auto Scaling group is not as operationally efficient as Lambda, which eliminates server management.

Option D: Using SQS is beneficial, but continuing to run application servers on EC2 instances with Auto Scaling groups isn't as operationally efficient as using Lambda. Lambda reduces management overhead by abstracting the underlying infrastructure. EC2 instances can add management overhead in the long term.

Question: 76

A company receives 10 TB of instrumentation data each day from several machines located at a single factory. The data consists of JSON files stored on a storage area network (SAN) in an on-premises data center located within the factory. The company wants to send this data to Amazon S3 where it can be accessed by several additional systems that provide critical near-real-time analytics. A secure transfer is important because the data is considered sensitive.
Which solution offers the MOST reliable data transfer?

A. AWS DataSync over public internet
B. AWS DataSync over AWS Direct Connect
C. AWS Database Migration Service (AWS DMS) over public internet
D. AWS Database Migration Service (AWS DMS) over AWS Direct Connect

Answer: B

Explanation:

The correct answer is **B. AWS DataSync over AWS Direct Connect.** Here's a detailed justification:

Data Volume and Real-Time Analytics: The company generates a significant 10 TB of data daily and requires near-real-time analytics. This necessitates a robust and efficient transfer mechanism.
Reliability and Security: Secure transfer is paramount due to the sensitive nature of the data.
AWS DataSync: AWS DataSync is a purpose-built service designed for moving large amounts of data between on-premises storage and AWS services. It handles data encryption in transit and at rest, ensuring data security. DataSync is optimized for network utilization, maximizing transfer speeds and minimizing transfer times.
https://aws.amazon.com/datasync/
AWS Direct Connect: AWS Direct Connect establishes a dedicated network connection from the company's on-premises data center to AWS. This connection bypasses the public internet, providing a more reliable, secure, and consistent network experience. Direct Connect significantly reduces network latency and increases bandwidth compared to internet-based transfers. https://aws.amazon.com/directconnect/ **Why A is less ideal:** While AWS DataSync is appropriate, transferring over the public internet (option A) introduces vulnerabilities to security breaches and network congestion, making the transfer less reliable and secure than using a dedicated connection.

Why C and D are incorrect: AWS Database Migration Service (DMS) is primarily for migrating databases. While it could technically transfer JSON files stored as BLOBs, it's not designed or optimized for this purpose. DataSync is much more efficient and appropriate for large file transfers. Additionally, DMS adds unnecessary complexity and overhead for a simple file transfer scenario.

Combined benefits of DataSync and Direct Connect: By combining AWS DataSync with AWS Direct Connect, the company gets a secure, reliable, and high-throughput data transfer pipeline. Direct Connect minimizes latency and ensures consistent network performance, while DataSync optimizes the transfer process. The encryption during transfer provided by DataSync in conjunction with Direct Connect reduces exposure of the data.

Consistency: Consistent, reliable data transfer is crucial for supporting near-real-time analytics because interruptions or delays in the data transfer can result in outdated or incomplete analyses.

Cost considerations: While Direct Connect involves additional costs, the improved reliability, security, and throughput may justify the investment, especially considering the critical nature of the data and analytics.

Therefore, using AWS DataSync over AWS Direct Connect is the most reliable data transfer solution for transferring the instrumentation data to Amazon S3 because it offers enhanced security, dedicated bandwidth, and optimized transfer mechanisms.

Question: 77

A company needs to configure a real-time data ingestion architecture for its application. The company needs an API, a process that transforms data as the data is streamed, and a storage solution for the data.
Which solution will meet these requirements with the LEAST operational overhead?

A. Deploy an Amazon EC2 instance to host an API that sends data to an Amazon Kinesis data stream. Create an Amazon Kinesis Data Firehose delivery stream that uses the Kinesis data stream as a data source. Use AWS Lambda functions to transform the data. Use the Kinesis Data Firehose delivery stream to send the data to Amazon S3.

B. Deploy an Amazon EC2 instance to host an API that sends data to AWS Glue. Stop source/destination checking on the EC2 instance. Use AWS Glue to transform the data and to send the data to Amazon S3.

C. Configure an Amazon API Gateway API to send data to an Amazon Kinesis data stream. Create an Amazon Kinesis Data Firehose delivery stream that uses the Kinesis data stream as a data source. Use AWS Lambda functions to transform the data. Use the Kinesis Data Firehose delivery stream to send the data to Amazon S3.

D. Configure an Amazon API Gateway API to send data to AWS Glue. Use AWS Lambda functions to transform the data. Use AWS Glue to send the data to Amazon S3.

Answer: C

Explanation:

The correct answer is C because it offers a serverless, scalable, and managed solution for real-time data ingestion with minimal operational overhead.

Here's why:

API Gateway: Amazon API Gateway allows you to create and manage APIs without managing any servers. It handles tasks like traffic management, authorization, and monitoring, significantly reducing operational burden compared to hosting an API on an EC2 instance. (Reference: https://aws.amazon.com/api-gateway/)

Kinesis Data Streams: Kinesis Data Streams is a fully managed, scalable, and durable real-time data streaming service. It's designed to ingest high-velocity data streams. Using it directly as the target of the API ensures immediate ingestion.

Kinesis Data Firehose: Kinesis Data Firehose is designed for loading data into data lakes and data warehouses. Using it with Kinesis Data Streams as a source is a common pattern. Firehose automatically scales, buffers, and compresses data. (Reference: https://aws.amazon.com/kinesis/data-firehose/)

Lambda Functions: Lambda allows you to execute code without provisioning or managing servers. Using Lambda functions within the Kinesis Data Firehose delivery stream enables real-time data transformations without the operational overhead of managing EC2 instances. This is done during data loading to S3 by Firehose (Reference: https://docs.aws.amazon.com/firehose/latest/dev/data-transformation.html)

S3: Amazon S3 provides highly durable and scalable object storage, ideal for storing the transformed data.

Why other options are less suitable:

Option A: Hosting the API on an EC2 instance increases operational overhead due to server management, patching, scaling, and monitoring.

Option B: AWS Glue is primarily designed for ETL (Extract, Transform, Load) jobs, which are typically batch-oriented, and is not the optimal solution for real-time data ingestion. AWS Glue is also overkill for the simple transformation needed in this case. The "stop source/destination checking" is also a security concern and not needed with the API Gateway/Kinesis solution.

Option D: Similar to B, using AWS Glue for data transformation is generally better suited for batch-oriented ETL pipelines, and API Gateway doesn't natively integrate with it for real-time streaming.

Therefore, option C is the most efficient and cost-effective choice for real-time data ingestion with the least operational overhead by leveraging serverless services.

Question: 78

A company needs to keep user transaction data in an Amazon DynamoDB table. The company must retain the data for 7 years.
What is the MOST operationally efficient solution that meets these requirements?

A. Use DynamoDB point-in-time recovery to back up the table continuously.

B. Use AWS Backup to create backup schedules and retention policies for the table.

C. Create an on-demand backup of the table by using the DynamoDB console. Store the backup in an Amazon S3 bucket. Set an S3 Lifecycle configuration for the S3 bucket.

D. Create an Amazon EventBridge (Amazon CloudWatch Events) rule to invoke an AWS Lambda function. Configure the Lambda function to back up the table and to store the backup in an Amazon S3 bucket. Set an S3 Lifecycle configuration for the S3 bucket.

Answer: B

Explanation:

The correct answer is B: Use AWS Backup to create backup schedules and retention policies for the table. This is the most operationally efficient solution for retaining DynamoDB data for 7 years for several reasons:

1. **Centralized Backup Management:** AWS Backup provides a single pane of glass to manage backups across various AWS services, including DynamoDB. This simplifies backup management and reduces the operational overhead associated with managing backups using separate service-specific tools. https://aws.amazon.com/backup/

2. **Automated Scheduling and Retention:** AWS Backup allows defining backup schedules and retention policies that automatically handle the creation and deletion of backups based on the defined parameters. This eliminates the need for manual intervention and ensures compliance with the 7-year retention requirement.

3. **Compliance and Auditing:** AWS Backup offers features for compliance reporting and auditing, allowing you to demonstrate adherence to data retention policies. https://docs.aws.amazon.com/aws-backup/latest/devguide/whatis.html

4. **Cost Optimization:** AWS Backup optimizes backup storage by using incremental backups and data compression, which helps reduce storage costs.

Now, let's look at why other options are less efficient:

A. **DynamoDB Point-in-Time Recovery (PITR):** While PITR allows restoring the table to any point in time within

the past 35 days, it doesn't satisfy the 7-year retention requirement.
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/PointInTimeRecovery.html

C. **On-Demand Backups with S3 Lifecycle:** Creating on-demand backups manually is not operationally efficient because it requires manual intervention and scheduling. Also, managing backups in S3 requires configuring S3 Lifecycle rules, adding operational complexity. The responsibility of remembering to perform these tasks remains with the user.

D. **EventBridge, Lambda, and S3 Lifecycle:** While this approach can meet the retention requirement, it involves more complex configurations than using AWS Backup. Building and maintaining a custom Lambda function introduces operational overhead for development, testing, and maintenance. The management of the S3 lifecycle, though automated, adds to the solution's complexity compared to AWS Backup.

Question: 79

A company is planning to use an Amazon DynamoDB table for data storage. The company is concerned about cost optimization. The table will not be used on most mornings. In the evenings, the read and write traffic will often be unpredictable. When traffic spikes occur, they will happen very quickly.
What should a solutions architect recommend?

A. Create a DynamoDB table in on-demand capacity mode.
B. Create a DynamoDB table with a global secondary index.
C. Create a DynamoDB table with provisioned capacity and auto scaling.
D. Create a DynamoDB table in provisioned capacity mode, and configure it as a global table.

Answer: A

Explanation:

Here's a detailed justification for why option A (Create a DynamoDB table in on-demand capacity mode) is the most appropriate solution for the scenario described:

The scenario highlights two key cost optimization concerns: the table is largely unused during mornings, and traffic is unpredictable with rapid spikes during evenings. On-demand capacity mode is ideally suited for these situations. In on-demand capacity mode, DynamoDB automatically scales capacity in response to actual workload needs. This means you only pay for the reads and writes your application performs, without needing to provision capacity upfront. This aligns perfectly with the morning downtime where no costs are incurred.

Because traffic spikes happen rapidly, provisioned capacity with autoscaling (option C) might not react quickly enough. Autoscaling takes some time to adjust capacity, potentially leading to throttling during the initial moments of a spike. On-demand mode, on the other hand, scales instantly.

A global secondary index (option B) is useful for querying data using attributes other than the primary key but doesn't address the cost optimization and unpredictable traffic concerns. A global table (option D) is for multi-region replication for disaster recovery and low-latency access in different geographical areas, which is also irrelevant to the given problem. It would simply increase costs without providing a benefit.

Therefore, on-demand capacity mode provides the best balance of cost efficiency during periods of low activity and responsiveness to unpredictable, rapid traffic spikes by eliminating the need for upfront capacity planning and enabling automatic and instant scaling.

Here are some resources for further reading:

DynamoDB On-Demand Capacity:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/HowItWorks.ReadWriteCapacityMode.html

DynamoDB Auto Scaling:
https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/AutoScaling.html

Question: 80

A company recently signed a contract with an AWS Managed Service Provider (MSP) Partner for help with an application migration initiative. A solutions architect needs ta share an Amazon Machine Image (AMI) from an existing AWS account with the MSP Partner's AWS account. The AMI is backed by Amazon Elastic Block Store (Amazon EBS) and uses an AWS Key Management Service (AWS KMS) customer managed key to encrypt EBS volume snapshots.
What is the MOST secure way for the solutions architect to share the AMI with the MSP Partner's AWS account?

A. Make the encrypted AMI and snapshots publicly available. Modify the key policy to allow the MSP Partner's AWS account to use the key.

B. Modify the launchPermission property of the AMI. Share the AMI with the MSP Partner's AWS account only. Modify the key policy to allow the MSP Partner's AWS account to use the key.

C. Modify the launchPermission property of the AMI. Share the AMI with the MSP Partner's AWS account only. Modify the key policy to trust a new KMS key that is owned by the MSP Partner for encryption.

D. Export the AMI from the source account to an Amazon S3 bucket in the MSP Partner's AWS account, Encrypt the S3 bucket with a new KMS key that is owned by the MSP Partner. Copy and launch the AMI in the MSP Partner's AWS account.

Answer: B

Explanation:

The correct answer is B. Here's why:

A is incorrect: Making the AMI and snapshots publicly available is highly insecure. It exposes the company's data to anyone, violating confidentiality and compliance requirements.

B is correct: Modifying the launchPermission property of the AMI allows you to selectively share the AMI with the MSP Partner's AWS account. This approach limits access to only the intended recipient. Crucially, because the EBS volumes are encrypted with a KMS key, the MSP Partner's account needs permission to use that key. Modifying the key policy to grant the MSP Partner's AWS account the necessary permissions enables them to launch instances from the AMI. This balances security with functionality, allowing the MSP Partner to use the shared AMI while maintaining control over access to the underlying data.

C is incorrect: While sharing the AMI with the MSP's account is correct, modifying the key policy to trust a new KMS key owned by the MSP is less straightforward and potentially less secure. This would necessitate re-encrypting the EBS volumes using the new key, adding unnecessary complexity and potential for data loss or corruption. Also, it requires the original account to relinquish some control over the encryption process. It is generally best practice to maintain control of KMS keys used to encrypt your data whenever possible.

D is incorrect: Exporting the AMI and storing it in an S3 bucket, even if encrypted with the MSP's KMS key, introduces unnecessary complexity and potential vulnerabilities. It involves creating a copy of the AMI, which can increase storage costs and introduce inconsistencies. Furthermore, exporting and importing AMIs can be time-consuming and may require downtime. Sharing the AMI directly via launchPermission is a more efficient and secure approach.

In summary, option B provides the most secure and efficient way to share the encrypted AMI with the MSP Partner by granting the MSP Partner access to the existing KMS key used to encrypt the EBS volumes, without exposing the AMI publicly or needing to create copies.

Supporting links:

Sharing AMIs: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/sharingamis-intro.html KMS Key Policies: https://docs.aws.amazon.com/kms/latest/developerguide/key-policies.html Encryption at Rest with KMS: https://docs.aws.amazon.com/kms/latest/developerguide/services-supported.html