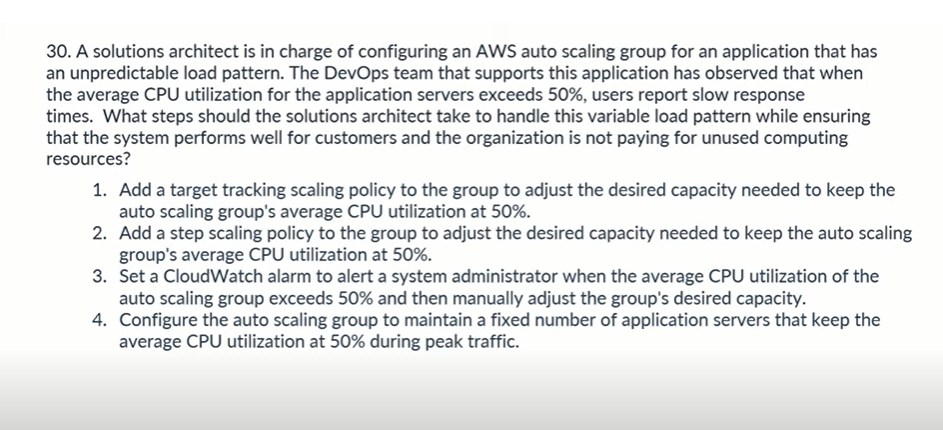
Question1: Please confirm the correct option



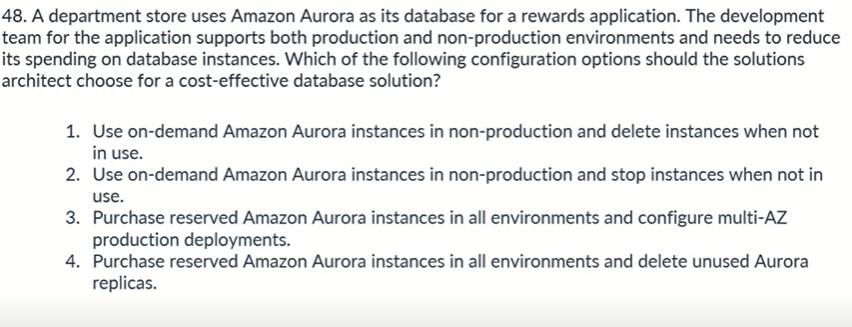
Option 3: This option involves setting a CloudWatch alarm to monitor CPU utilization and alerting a system administrator when the average CPU utilization exceeds 50%. The administrator can then manually adjust the desired capacity of the auto scaling group as needed. This approach requires manual intervention, which can lead to delays in scaling up or down to meet the demands of the application, potentially impacting performance.

Option 4:

This option involves configuring the auto scaling group to maintain a fixed number of application servers to keep the average CPU utilization at 50% during peak traffic. This approach may be less cost-effective than the other options, as it involves maintaining a fixed number of instances even when they are not needed.

The best option for handling the unpredictable load pattern of an application while ensuring that the system performs well for customers and the organization is not paying for unused computing resources is Option 1 or Option 2. These options provide an automated way to adjust the desired capacity of the auto scaling group based on the average CPU utilization, which can help ensure that the application is responsive and performing well while also minimizing costs by scaling up and down as needed. The choice between these options may depend on the specific needs and usage patterns of the application.

Question 2: Please confirm the correct option



Option 1: This option allows the development team to spin up non-production instances as needed and delete them when they are no longer required, which can help reduce costs. However, it requires manual management of the instances and may result in longer setup times when new instances are needed.

Option 2:

This option allows the development team to stop non-production instances when they are not in use, which can help reduce costs. However, it still requires manual management of the instances and may result in longer setup times when new instances are needed.

Option 3:

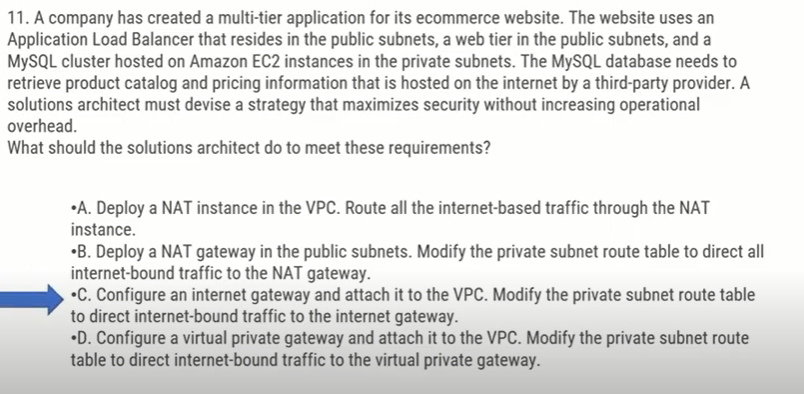
This option involves purchasing reserved instances for all environments, which can provide cost savings over on-demand instances. Additionally, using multi-AZ deployments can provide additional availability and durability benefits. However, this option requires upfront investment in reserved instances and may not be the most cost-effective option for non-production environments.

Option 4:

This option is similar to option 3 but involves deleting unused Aurora replicas to reduce costs. This can provide additional cost savings, but may require additional management overhead to ensure that unused replicas are properly identified and deleted.

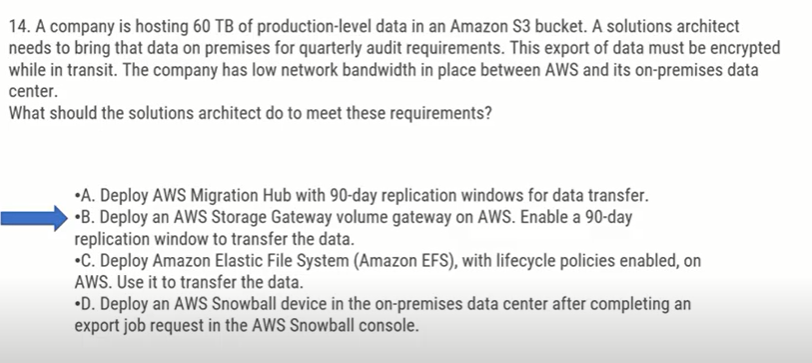
Overall, the most cost-effective option for a department store using Amazon Aurora as its database for a rewards application may depend on the specific needs and usage patterns of the development team. Option 1 or Option 2 may be the most cost-effective for non-production environments, while Option 3 or Option 4 may be the most cost-effective for production environments. The solutions architect should carefully evaluate the tradeoffs and costs of each option to determine the best approach.

Question3: should be option B rather than C as we can’t expose private subnets to Internet Gateway.. Please confirm



The best option to maximize security without increasing operational overhead would be to use a NAT gateway. Option B, which deploys a NAT gateway in the public subnets and modifies the private subnet route table to direct all internet-bound traffic to the NAT gateway, is the correct choice in this case.

Question 4: Since low network bandwidth is an issue as per question, can’t it be Option D rather than B. Please confirm.



Given the large amount of data to be transferred, the low network bandwidth between AWS and the on-premises data center, and the requirement for encrypted transit, deploying an AWS Snowball device is the most suitable solution for this scenario.

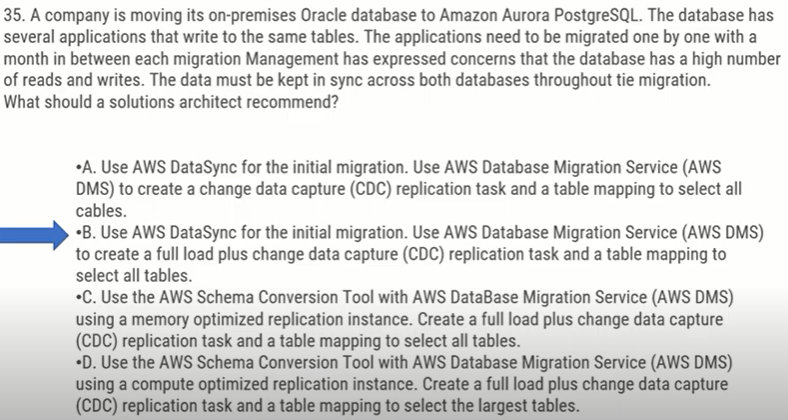
Option A, deploying AWS Migration Hub, and Option B, deploying an AWS Storage Gateway volume gateway, are not suitable for this scenario as they do not provide a secure and efficient way to transfer large amounts of data over a low-bandwidth network. Option C, deploying Amazon EFS, is not suitable for this scenario because it is not designed for data transfer to an on-premises data center.

Question 5:

AWS DataSync is a secure, online service that automates and accelerates moving data between on premises and AWS Storage services.

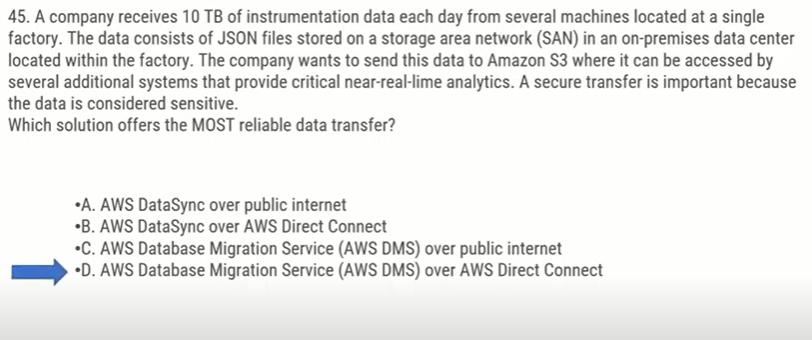
The AWS Schema Conversion Tool (AWS SCT) helps convert your existing database schema from one database engine to another.

As per above explanation, should be either option C or D and NOT B. Please advise the correct option.



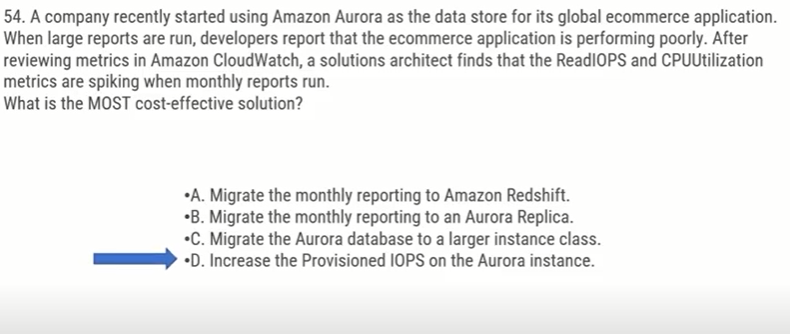
Since the database has a high number of reads and writes, the best option would be to use AWS Database Migration Service (AWS DMS) to create a full load plus change data capture (CDC) replication task and a table mapping to select the largest tables. Option D is the correct answer as it suggests using the AWS Schema Conversion Tool with AWS DMS using a compute-optimized replication instance. This option ensures that data is kept in sync across both databases throughout the migration process. Using a memory optimized replication instance, as suggested in option C, may not provide the required performance for a database with a high number of reads and writes. Options A and B suggest using AWS DataSync for the initial migration, which is not necessary for this scenario.

Question 6: Is D the correct answer?? Should be B .. Please confirm



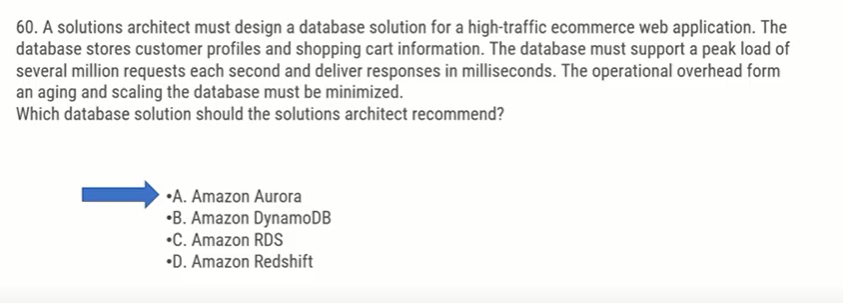
The MOST reliable data transfer option for this scenario is D. AWS Database Migration Service (AWS DMS) over AWS Direct Connect. Direct Connect provides a dedicated, private network connection between on-premises infrastructure and AWS, ensuring a secure and reliable transfer of sensitive data. Using AWS DMS, the data can be migrated to Amazon S3 in a fully-managed, reliable, and secure way. AWS DataSync over public internet is less reliable and secure, and AWS DataSync over AWS Direct Connect is not the best choice since it is typically used for transferring data between on-premises storage and AWS storage services

Question 7: Is D the correct option? If yes, why is it so?



The most cost-effective solution to address the performance issues when large reports are run in an Amazon Aurora database is to increase the Provisioned IOPS on the Aurora instance. This will provide more IOPS to handle the spikes in Read IOPS during the reporting period and improve CPU utilization, ensuring that the e commerce application performance remains stable. Migrating the monthly reporting to Amazon Redshift would involve additional costs, and migrating to an Aurora Replica would not address the root cause of the performance issues. Migrating the Aurora database to a larger instance class would also involve additional costs and may not be necessary if increasing Provisioned IOPS can address the performance issues. Therefore, option C and D are the best solutions.

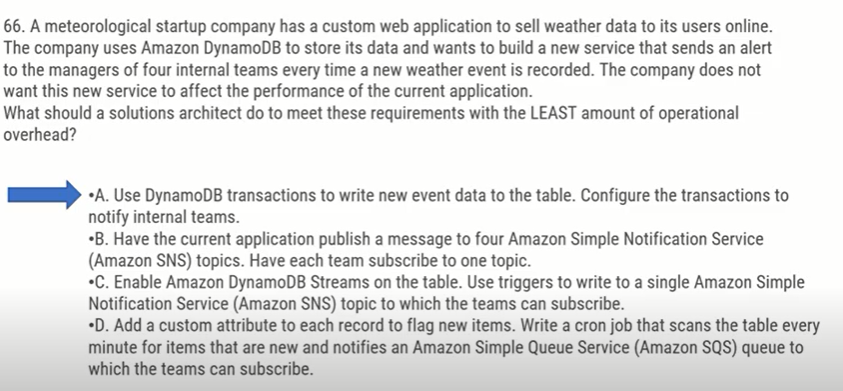
Question 8: For storing shopping cart info, should be DynamoDB rather than Aurora. Please confirm



Amazon DynamoDB would be the best option for this use case, as it is designed for high-traffic web applications that require millisecond response times and can handle several million requests per second. DynamoDB is also fully managed, meaning that operational overhead for scaling and aging the database is minimized. Amazon Aurora and Amazon RDS are also good database solutions, but they may not be able to handle the peak load of several million requests per second without significant scaling and operational overhead. Amazon Redshift is a data warehousing solution, and is not well-suited for transactional workloads such as customer profiles and shopping cart information.

Question 9: As per my understanding, we can use Dynamo stream with Lambda to trigger a notification whenever a new update is made to the DB. Therefore Option C should be the correct one as per my understanding. However, A is the correct one as per website.

Please confirm



Option B is a possible solution. Instead of using DynamoDB transactions or triggers, the current application can publish a message to four Amazon SNS topics, with each team subscribing to one topic. This solution reduces operational overhead because it leverages an existing mechanism for publishing messages, and the teams can receive notifications without any additional setup. Therefore, option B is the best choice.

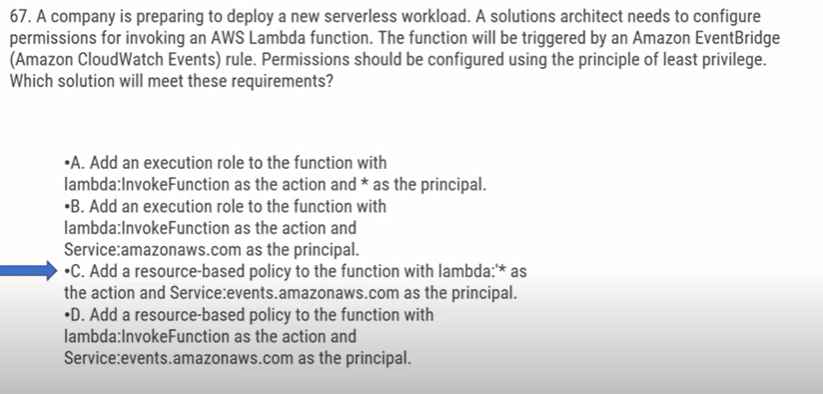
Option A is not an ideal solution because it requires setting up DynamoDB transactions to write new event data and notify internal teams. This would require additional overhead to ensure that the transactions are executed correctly and that notifications are sent to the correct teams.

Option C is not the best choice because it requires enabling DynamoDB Streams on the table, setting up triggers to write to a single Amazon SNS topic, and ensuring that the correct teams are subscribed to the topic. This solution requires more overhead than option B.

Option D is not the best choice because it requires adding a custom attribute to each record to flag new items, writing a cron job to scan the table every minute, and notifying an Amazon SQS queue to which the teams can subscribe. This solution requires more overhead than option B, which can achieve the same result with fewer moving parts.

Question 10: As the questions talks about “LEAST” privilege, proving lambda as \* would give all permissions and hence couldn’t be the correct option here. Therefore D should be the correct option as per my understanding. But the correct one is C as per website.

Please confirm.



To configure permissions for invoking an AWS Lambda function that will be triggered by an Amazon EventBridge (Amazon CloudWatch Events) rule, using the principle of least privilege, a resource-based policy should be added to the function with lambda:InvokeFunction as the action and Service:events.amazonaws.com as the principal. This will allow only EventBridge to invoke the function and prevent other services or entities from invoking it unnecessarily, thus meeting the requirement of least privilege.

Therefore, the correct answer is D. "Add a resource-based policy to the function with lambda:InvokeFunction as the action and Service:events.amazonaws.com as the principal."

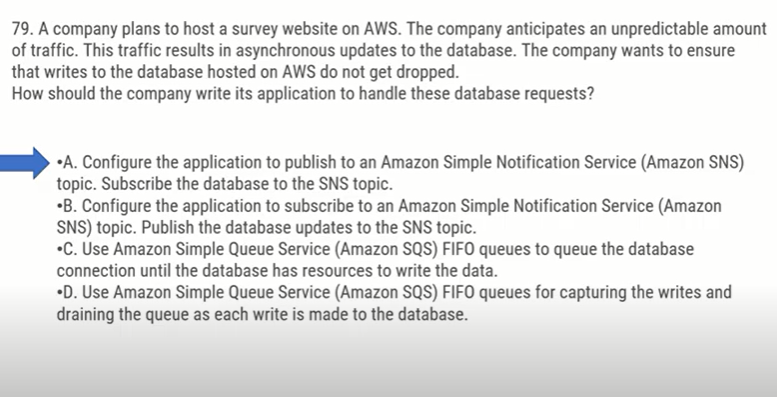
Option A is incorrect because specifying a wildcard (\*) as the principal will grant permission to any entity, which is not the principle of least privilege.

Option B is incorrect because it grants permission to all services in the AWS account, which is not the principle of least privilege.

Option C is incorrect because it uses a wildcard (\*\*) in the action, which is not recommended as it grants permission for all actions.

Question 11: Since the messages (writes) should not be dropped, it should be stored somewhere for processing, and I think SQS should be the correct option here. So my understanding is Option D.

However, the marked option is the correct one as per the website from where I picked up the question. Please confirm.



Option C and D are the possible solutions for handling asynchronous updates to the database without dropping writes.

Option C: Using Amazon Simple Queue Service (Amazon SQS) FIFO queues to queue the database connection until the database has resources to write the data.

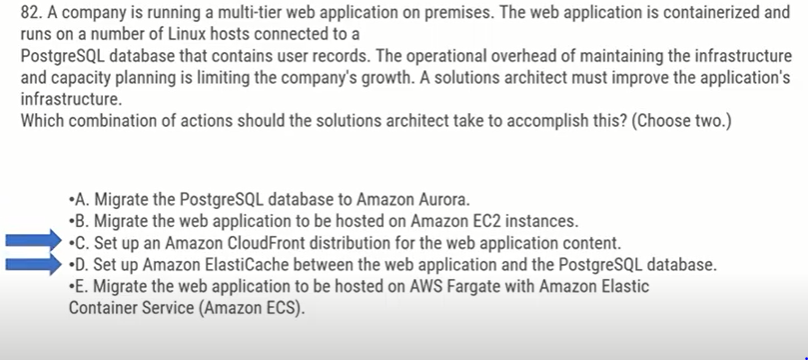
Option D: Using Amazon Simple Queue Service (Amazon SQS) FIFO queues for capturing the writes and draining the queue as each write is made to the database.

Both options provide a reliable way to manage the write requests and ensure that they are not dropped. However, option D is the better solution because it offers more control over the processing of the write requests.

By using an SQS FIFO queue, the application can place the database write requests in a queue, and the database can then process them in the order they were received. The database can then return a response to the application when the write request has been successfully processed, and the application can then remove the write request from the queue. This ensures that the database is not overloaded with too many write requests at once and that no write requests are lost.

Question 12: As the question asks for improving the application’s infrastructure, I think using AWS Fargate (serverless) to manage application can reduce overhead of maintaining the infrastructure (asked in questions) could be the correct option here. So E is the option.. Not sure about the other option (Option ‘A’ definitely not.. as migrating db increases the overhead and Option ‘B’ talks about hosting on EC2 instances which again is not a good option)

Please confirm the correct Options..s



The recommended solution for improving the company's infrastructure is to migrate the PostgreSQL database to Amazon Aurora and migrate the web application to be hosted on AWS Fargate with Amazon Elastic Container Service (Amazon ECS). These actions will reduce the operational overhead and allow for better capacity planning.

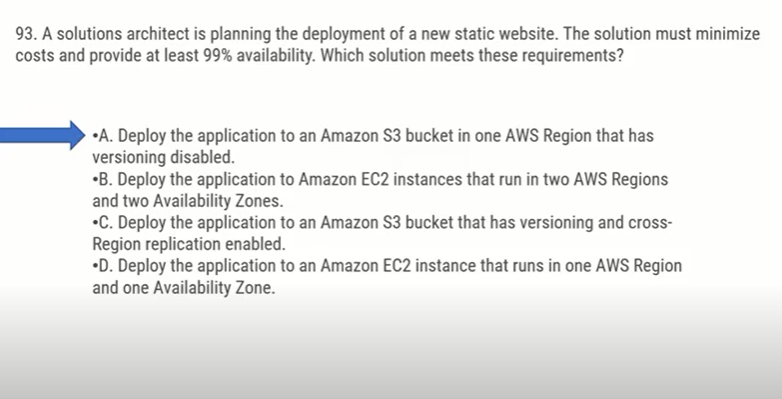
Option C, setting up an Amazon CloudFront distribution for the web application content, can improve the application's performance by caching content close to the end-users.

Option D, setting up Amazon ElastiCache between the web application and the PostgreSQL database, can improve the application's performance by caching frequently accessed data, but it doesn't address the operational overhead and capacity planning issues.

Therefore, the correct options are A and E.

Question 14:

Correct Answer is marked below. How can the versioning disable achieve 99% availability as mentioned in the question? Is Option A the correct one?



The most cost-effective and highly available solution for deploying a static website on AWS is to use Amazon S3. Option C is the correct answer since it satisfies both requirements of minimizing costs and providing at least 99% availability. By enabling versioning and cross-Region replication, the solution can provide durability, availability, and low latency for static website hosting. S3's multi-AZ architecture ensures high availability, and S3's built-in features such as Versioning, Object Lock, and Cross-Region Replication provide durability and resilience. Options A and D are incorrect because they do not provide the necessary level of availability. Option B is incorrect because it is more expensive than using Amazon S3 and does not provide the same level of durability and availability as Amazon S3.