Boost up Your Certification Score

Amazon
SAP-C03
AWS Certified Solutions Architect – Professional

For More Information – Visit link below:

https://www.examsboost.com/

Product Version

✓ Up to Date products, reliable and verified.
✓ Questions and Answers in PDF Format.

Visit us at: https://www.examsboost.com/test/sap-c03

Latest Version: 6.0

Question: 1

An online retailer is designing an event-driven order-processing flow. For each order, the producer publishes transitions in their authoritative sequence. Billing and Fulfillment must each receive every transition and apply that order's transitions in sequence. Both applications must process unrelated orders concurrently, and their worker fleets must scale independently. No ordering is required between different orders.
The team compares these transport designs:
Each application can atomically commit its local transition state and a processed-event record. This transaction cannot include the external payment API. The Billing application performs one initial charge per order when it processes ChargeRequested.
The payment provider offers the following contract:
• Charge accepts an optional operation key. For an order's entire processing lifetime, repeated calls with the same key and charge parameters refer to the same charge and cannot create an additional charge.
• Calls without an operation key are independent charge requests.
• An authoritative status lookup by operation key supports reconciliation. A pending or unresolved outcome does not establish that a charge failed.
During a failure, the provider successfully charges a customer, but Billing loses the response and crashes before recording the result locally or deleting the message. Billing subsequently receives the message again. The design must prevent a second initial charge and must eventually record the successful charge after recovery.
Which complete design meets these requirements? Choose ONE.

A. Use an SNS FIFO topic and a separate SQS FIFO queue for each application. Set MessageGroupId to order ID, process each group serially, and use event ID for transport deduplication. When Billing has no processed-event record, call Charge without an operation key. After a successful response, atomically commit the local transition and processed-event record, then delete the message. If the response is lost, repeat this procedure.
B. Use an SNS FIFO topic and a separate SQS FIFO queue for each application. Set MessageGroupId to the constant value all-orders and process each group serially. For Billing, use a stable initial-charge operation key per order and reconcile unresolved outcomes with that key. Once the charge succeeds, atomically commit the local transition and processed-event record, then delete the message.
C. Use an SNS FIFO topic and a separate SQS FIFO queue for each application. Set MessageGroupId to order ID and process each group serially while allowing different groups to run concurrently. For Billing, use a stable initial-charge operation key per order across retries and reconcile unresolved outcomes with that same key. Once the charge succeeds, atomically commit the local transition and processed-event record, then delete the message; an already committed event can be acknowledged without charging again.
D. Use one SQS FIFO queue shared by the Billing and Fulfillment worker fleets as competing consumers. Set MessageGroupId to order ID and process each group serially while allowing different groups to run concurrently. For Billing, use a stable initial-charge operation key per order and reconcile unresolved outcomes with that key. Once the charge succeeds, atomically commit the local transition and processed-event record, then delete the message.

Answer: C

Explanation:
C satisfies the distribution, ordering, concurrency, and payment-recovery requirements. Separate application queues give Billing and Fulfillment their own copies and independent processing capacity. Grouping by order ID preserves the required entity sequence without serializing unrelated orders, provided each application processes a group serially. The payment provider's stated operation-key contract protects the external charge across retries. After the lost response, Billing can reconcile or retry the same logical operation using the same key, discover the successful charge, and commit its local result. An atomic local transition and processed-event record also handle a later redelivery after local commit but before message deletion. Transport deduplication cannot resolve the gap between an external charge and the local transaction, so this design does not claim an atomic end-to-end exactlyonce guarantee. A leaves that gap unprotected, B unnecessarily serializes unrelated orders, and D fails to give every application every transition.
A - Incorrect: The fan-out and per-order grouping meet the distribution and ordering requirements, but the payment procedure fails the stated recovery case. Billing has no local processed-event record even though the provider already charged the customer. Repeating an unkeyed Charge call creates an independent request and can charge the customer again. Transport deduplication does not establish whether the external charge committed and does not protect an application retry of that operation.
B - Incorrect: The separate queues support application fan-out, and the stable payment key supports safe reconciliation. However, putting every order in the same message group and processing that group serially prevents each application from processing unrelated orders concurrently. Per-order ordering requires a group per order, rather than one sequence covering all orders.
C - Correct: Each application receives a separate copy through its own FIFO queue. An order ID identifies the required ordering scope, and serial processing within that group preserves its transition sequence while other groups can progress concurrently. Billing reuses the same initial-charge key after the lost response, so the provider cannot create another charge for that logical operation. Reconciliation establishes the outcome before Billing commits its local transition and processed-event record. Subsequent redelivery can use that committed record without repeating the charge.
D - Incorrect: The payment procedure and per-order grouping are appropriate, but a shared work queue distributes messages among competing consumers. It does not give Billing and Fulfillment an independent copy of every transition. A transition consumed and deleted by one application is therefore not reliably available to the other application, violating the explicit distribution requirement.

Question: 2

An AWS solutions architect is validating the access of a temporary role session. The role, session, and S3 bucket belong to AWS account 123456789012. Role assumption has already succeeded.
Role ARN: arn:aws:iam::123456789012:role/ReportPublisher
Active session ARN: arn:aws:sts::123456789012:assumed-role/ReportPublisher/nightly-42
Bucket: audit-reports-123456789012
The role's identity-based policy contains one Allow statement for s3:GetObject, s3:PutObject, and s3:DeleteObject on arn:aws:s3:::audit-reports-123456789012/*.
The role's permissions boundary contains only an Allow for s3:GetObject on arn:aws:s3:::audit-reports123456789012/*. The session policy also contains only an Allow for s3:GetObject on that resource.

Neither policy contains an explicit Deny.
The bucket policy contains exactly these statements, with no conditions:
There are no other grants or explicit denies. No organization guardrail or service-specific authorization requirement changes the result, and all nonauthorization prerequisites are satisfied.
The active session makes four independent requests:
1. s3:GetObject on arn:aws:s3:::audit-reports-123456789012/reports/summary.csv
2. s3:PutObject on arn:aws:s3:::audit-reports-123456789012/role-drop/new.csv
3. s3:PutObject on arn:aws:s3:::audit-reports-123456789012/session-drop/new.csv
4. s3:DeleteObject on arn:aws:s3:::audit-reports-123456789012/session-drop/obsolete.csv
Which option correctly evaluates all four requests? Choose ONE.

A. Request 1: allowed. Request 2: denied. Request 3: denied. Request 4: denied.
B. Request 1: allowed. Request 2: allowed. Request 3: allowed. Request 4: denied.
C. Request 1: allowed. Request 2: denied. Request 3: allowed. Request 4: denied.
D. Request 1: allowed. Request 2: denied. Request 3: allowed. Request 4: allowed.

Answer: C

Explanation:
C is correct. Request 1 is allowed because the role's identity-based policy, permissions boundary, and session policy all allow GetObject on the specified object, and the bucket's explicit deny applies only to DeleteObject. Request 2 is denied: although the bucket policy allows PutObject through a grant naming the IAM role ARN, that grant is limited by implicit denies in the boundary and session policy, which allow only GetObject. Request 3 is allowed because the bucket policy grants PutObject directly to the exact active assumed-role session ARN in the same account; implicit denies in the identity policy, boundary, or session policy do not limit this direct grant. Request 4 is denied because the bucket's explicit DeleteObject deny overrides its direct session allow. The decisive distinction is the principal receiving each resource grant, followed by the overriding effect of any applicable explicit deny. These outcomes depend on the stated same-account scope and absence of additional restricting controls.
A - Incorrect: Request 1 is allowed, and requests 2 and 4 are denied, but request 3 is allowed. Its bucketpolicy statement grants s3:PutObject directly to the exact active session ARN. Within the same account,
implicit denies in the identity policy, permissions boundary, or session policy do not limit that direct session grant. Treating every grant as an intersection incorrectly denies request 3.
B - Incorrect: Requests 1 and 3 are allowed, and request 4 is denied, but request 2 is denied. The bucket-policy grant for role-drop/* names the IAM role ARN. That grant remains limited by implicit denies
in the permissions boundary and session policy, both of which omit s3:PutObject. The exception for a grant directly to a session ARN cannot be applied to this role ARN grant.
C - Correct: All four outcomes are correct. Request 1 has matching identity-policy, boundary, and session-policy allows. Request 2 is blocked because the resource grant names the role ARN and both limiting policies omit s3:PutObject. Request 3 receives a direct same-account session ARN grant that is unaffected by those implicit denies. Request 4 matches the bucket policy's explicit s3:DeleteObject deny,
which overrides its direct session allow.
D - Incorrect: Requests 1 and 3 are allowed, and request 2 is denied, but request 4 is denied. Naming

the exact session ARN prevents implicit denies in the listed permission sets from limiting the direct grant;
it does not override explicit denies. The bucket policy explicitly denies s3:DeleteObject on every object in
the bucket, including session-drop/obsolete.csv.

Question: 3

A solutions architect must reduce network costs for an application running in AZ-A and AZ-B of one AWS Region. Each AZ currently uses its own NAT gateway for calls to a same-Region partner API and for unrelated internet traffic.
For this scenario, the API supports both its existing public path through NAT and an AWS PrivateLink interface endpoint. It does not support a gateway endpoint. Both supported paths meet the application's
security and performance requirements. Traffic through the interface endpoint bypasses NAT.
Availability requirement: After either application AZ becomes completely unavailable, the surviving AZ must continue accessing both the API and unrelated internet destinations without depending on resources in the failed AZ. The API provider remains available. Both NAT gateways must remain for unrelated internet traffic. Endpoint clients use their local endpoint when one is provisioned; a design with
an endpoint only in AZ-A sends AZ-B's API traffic across AZs. No endpoint-to-NAT fallback is configured.
Use this complete illustrative cost ledger for a 720-hour month. These rates are scenario assumptions, not live AWS prices.
Count request and response data in processing charges. Count each GB once at each processing resource it traverses. The cross-AZ rate is the combined charge for each GB that crosses between the AZs; do not add another per-side charge. Local paths have no cross-AZ charge. There are no other charges or discounts in this comparison, and traffic volumes remain unchanged.
Which architecture and cost conclusion provides the lowest monthly cost while satisfying the availability requirement? Choose ONE.

A. Retain both NAT gateways and provision the interface endpoint only in AZ-A. Route both AZs' API traffic through that endpoint without fallback. Budget $127.20 per month, saving $70.80 compared with the current design.
B. Retain the current design, routing each AZ's API and unrelated internet traffic through its local NAT gateway. Budget $108.00 per month and reject the two-AZ interface endpoint design because it would cost $128.40.
C. Retain both NAT gateways for unrelated internet traffic and provision the interface endpoint in both AZs. Route each AZ's API traffic through its local endpoint. Budget $128.40 per month, saving $69.60 compared with the current design.
D. Retain both NAT gateways for unrelated internet traffic and provision the interface endpoint in both AZs. Route each AZ's API traffic through its local endpoint. Budget $56.40 per month, saving $141.60 compared with the current design.

Answer: C

Explanation:
C is correct. Each AZ generates 1,200 GB of API traffic and 200 GB of unrelated internet traffic, including

requests and responses. The current two-NAT design costs 2 × 720 × $0.050 + 2,800 × $0.045 = $198.00 per month. Moving only API traffic to local interface endpoints preserves $72.00 of NAT availability charges and $18.00 of unrelated NAT processing, while adding $14.40 of endpoint availability charges and $24.00 of endpoint processing. The resulting $128.40 total saves $69.60. Equivalently, replacing API processing saves 2,400 × ($0.045 − $0.010) = $84.00 before subtracting $14.40 of added endpoint hours. Provisioning the endpoint in only AZ-A reduces the total to $127.20, but leaves the surviving AZ-B without API access if AZ-A fails. Keeping local NAT paths is compliant but costs $198.00. The $56.40 endpoint estimate incorrectly treats retained NAT hours as avoidable. These calculations apply to the stated interface endpoint scenario; they do not assign interface endpoint charges to a gateway endpoint.
The illustrative ledger does not establish a production estimate.
A - Incorrect: The arithmetic is correct, but the architecture fails the availability requirement. Retained NAT hours cost 2 × 720 × $0.050 = $72.00. Unrelated NAT processing costs 400 × $0.045 = $18.00. One provisioned endpoint AZ costs 720 × $0.010 = $7.20, and processing all 2,400 GB of API requests and responses costs $24.00. AZ-B's 1,200 GB of API traffic adds $6.00 in cross-AZ transfer. The total is $127.20, saving $70.80 against $198.00. However, losing AZ-A also removes AZ-B's only configured API endpoint path. The additional $1.20 monthly saving compared with C does not justify violating the required resilience.
B - Incorrect: The local NAT architecture satisfies the availability requirement, but its stated cost omits all
response data. The $108.00 figure equals $72.00 in gateway hours plus $36.00 for only 800 GB of requests: 600 GB for the API and 200 GB for unrelated destinations. Responses add another 2,000 GB, costing $90.00. The complete NAT-only cost is therefore $198.00, which exceeds the compliant two-AZ endpoint design's $128.40.
C - Correct: This design satisfies the availability requirement and has the lowest cost among the compliant designs. Both local NAT gateways remain available for unrelated internet traffic, and each AZ has its own API endpoint path. Retained NAT hours cost $72.00, and 400 GB of unrelated requests and responses cost $18.00 in NAT processing. Endpoint hours cost 2 × 720 × $0.010 = $14.40, and 2,400 GB of API requests and responses cost $24.00 in endpoint processing. No traffic crosses AZs. The total is $128.40, saving $69.60 compared with the $198.00 NAT-only baseline.
D - Incorrect: The architecture satisfies the availability requirement, but the cost incorrectly removes the retained NAT gateways' fixed charges. Its $56.40 total includes $18.00 of unrelated NAT processing, $14.40 of endpoint hours and $24.00 of endpoint processing, but omits $72.00 of NAT gateway hours. Moving API traffic does not eliminate those hours because both gateways must remain for unrelated internet destinations. Adding the unavoidable $72.00 produces $128.40 and a $69.60 saving.

Question: 4

An AWS workload must survive the loss of its primary Region. The business requires an RTO of 15 minutes and an RPO of 30 seconds. Recovery is complete only when the service handles the required production load and its business-transaction and data-consistency checks succeed. Normal monthly disaster-recovery spending must not exceed 35% of the primary stack's monthly baseline cost; temporary incident-time scaling is funded separately.
The team exercised the following fixed proposals against an infrastructure outage beginning at 10:00:00. Primary-Region writes stopped at that instant. The figures are illustrative observations for these configurations, not AWS recovery guarantees. The exercise covered required dependencies and capacity. No redesign of the proposals is included in this decision.

The warm-standby exercise recorded this sequence:
• 10:01:00: Detection and recovery decision complete.
• 10:02:00: Original database writer fenced against accepting writes, including if its Region returns unexpectedly.
• 10:04:00: Recovery database promotion complete; exclusive write authority verified.
• 10:08:00: Required application capacity ready.
• 10:09:00: Client traffic reaches the recovery application.
• 10:12:00: Required full-load business and data checks complete.
The standby databases remain read-only until explicitly promoted. The routing system selects application endpoints and does not change database roles. The tested fencing and promotion procedures are available and included in the standby exercise results.
After failover, the recovery Region will accept new acknowledged writes. Failback must preserve every such write. A brief controlled write pause is permitted during the return to the original Region.
Which plan satisfies all requirements using the supplied readiness and recovery evidence? Choose ONE.

A. Adopt pilot light. Deploy the missing serving tier, fence the original writer, promote the recovery database, scale, switch traffic, and complete the exercised business checks. For failback, rebuild the original Region, synchronize it from the recovery writer, pause writes for final synchronization, and transfer exclusive write authority before validating and returning traffic.
B. Adopt warm standby. Follow the exercised fencing, promotion, scaling, routing, and validation sequence. For failback, rebuild the original Region, restore its last pre-outage database snapshot as the writer, and resume replication from that database to the recovery Region. Validate the rebuilt stack and return traffic once the original Region is healthy.
C. Adopt multi-site active/active at its tested capacity and cost. Use its exercised write-ownership controls and traffic-evacuation procedure to operate from the surviving Region. For failback, synchronize the returning Region from the current authoritative data and validate write ownership and application behavior before readmitting that Region to service.
D. Adopt warm standby. Follow the exercised fencing, promotion, scaling, routing, and validation sequence. For failback, rebuild the original Region and synchronize it from the recovery writer. Pause writes for final synchronization, fence the recovery writer, promote the synchronized original database, validate the service, and then return traffic.

Answer: D

Explanation:
D is the only plan that satisfies every stated requirement. Warm standby costs 32%, has a 20-second recoverable-data gap, and reaches validated full-load service in 12 minutes. The traffic transition at minute 9 is not the RTO endpoint because required validation finishes at minute 12. Writer fencing and promotion must occur independently of endpoint routing. Once recovery-site writes begin, that site's current data must be synchronized back before a controlled return of write authority. A takes 18 minutes and breaches the RTO; B would discard newer recovery-site writes during failback; C exceeds the ongoing-cost ceiling. The comparison uses configuration-specific illustrative evidence rather than treating published strategy ranges as guarantees. The illustrative timings do not guarantee future recovery performance.
A - Incorrect: Pilot light meets the cost limit at 18% and the RPO with a recoverable commit 20 seconds before the outage. Its complete recovery takes 18 minutes, exceeding the 15-minute RTO. Correct writer handover and safe failback do not compensate for missing the recovery-time requirement. A general

description of pilot-light recovery cannot override this configuration's exercise results.
B - Incorrect: Warm standby meets the numerical requirements: 32% ongoing cost, 20 seconds of potential data loss, and 12 minutes to validated full-load service. Its initial failover also establishes exclusive write authority correctly. However, restoring the original Region's pre-outage snapshot as authoritative and restarting replication from it would omit the newer acknowledged writes accepted in the recovery Region. Infrastructure health does not establish that the restored database contains the authoritative data.
C - Incorrect: The active/active proposal meets the recovery objectives in the exercise: validated service returns in 3 minutes and its recoverable commit is 5 seconds before the outage. Its ownership and reintegration procedures also address data consistency. However, its fixed ongoing DR cost is 75%, exceeding the mandatory 35% limit. The fastest proposal is not an acceptable choice when it violates an explicit business constraint.
D - Correct: Warm standby meets the 35% cost ceiling at 32%. Its latest recoverable acknowledged commit is 20 seconds before the outage, within the 30-second RPO. Recovery completes after business and data validation at 10:12:00, within the 15-minute RTO. Fencing the original writer and promoting the
recovery database establish write authority separately from routing. During failback, synchronization from the recovery writer preserves the newest accepted data; the final write pause and controlled writer
transfer prevent losing recovery-site writes or creating competing writers.

Question: 5

A solutions architect must recover an Amazon ECS payment-processing service that uses rolling deployments with early success criteria. Both the deployment circuit breaker and CloudWatch alarm failure detection are configured with automatic rollback enabled.
Deployment timeline:
• 09:00: Release A, task definition payments:41, reaches COMPLETED. Its immutable application artifact and configuration remain available for redeployment.
• 10:00: Release B, task definition payments:42, starts deploying.
• 10:04: ECS marks B COMPLETED after its early success criteria are satisfied.
• 10:12: The configured payment-error alarm enters ALARM. Business checks confirm that B has issued unintended payment requests. No recovery action has occurred.
Both releases use the same database. B has written payment records with status CAPTURED_V2, which A cannot process. A tested compatibility conversion maps that status to A-supported CAPTURED without changing business meaning, amounts, provider transaction IDs, or idempotency keys. After request intake and payment workers are paused, this conversion makes all retained records usable by A. All records created since 10:00 must be preserved.
Some unintended payments have already been captured by the external payment provider, and related messages have already caused downstream actions. An existing reconciliation runbook can identify those effects and apply authorized compensation.
Which procedure correctly distinguishes the available automatic rollback before 10:04 from recovery for the actual 10:12 failure, while addressing persistent state and external effects? Choose ONE.

A. A failure detected before 10:04 could trigger automatic rollback to A because A previously reached COMPLETED. For the 10:12 failure, pause intake and workers, perform the compatibility conversion, and

let the configured alarm roll B back automatically to A. Verify application and data behavior, then reconcile external effects using the existing runbook.
B. A failure detected before 10:04 could trigger automatic rollback to A because A previously reached COMPLETED. For the 10:12 failure, pause intake and workers, separately deploy A and resume processing, and then perform the compatibility conversion. Verify application and data behavior, then reconcile external effects using the existing runbook.
C. A failure detected before 10:04 could trigger automatic rollback to A because A previously reached COMPLETED. For the 10:12 failure, pause intake and workers, perform the compatibility conversion, and separately deploy A with its retained configuration. Verify application and data behavior before resuming processing, and reconcile external effects using the existing runbook.
D. A failure detected before 10:04 could trigger automatic rollback to A because A previously reached COMPLETED. For the 10:12 failure, pause intake and workers, perform the compatibility conversion, and separately deploy A with its retained configuration. Verify application and data behavior before resuming processing, treating the reversal as compensation for captured payments and downstream message effects.

Answer: C

Explanation:
C is correct. For this rolling deployment using early success criteria, deployment circuit breaker and CloudWatch alarm rollback apply until ECS completes the deployment. A failure detected before 10:04 could therefore trigger configured automatic rollback, with A satisfying the prerequisite of a previous deployment in COMPLETED state. The actual alarm occurs after B has completed, so these mechanisms no longer roll back that deployment. Recovery requires a separate controlled deployment or
another authorized remedy. A is available, but its inability to process CAPTURED_V2 makes immediate restoration unsafe. Pausing processing and applying the tested compatibility conversion first preserves the required records and makes A usable. Application and data checks must precede resumption. Restoring A and converting local records do not reverse captured payments or downstream actions caused by delivered messages; those require reconciliation and authorized compensation.
A - Incorrect: A correctly identifies the earlier rollback opportunity, the previous COMPLETED prerequisite, the need for compatible data, and the need to reconcile external effects. It fails at the deployment-state boundary: B reached COMPLETED at 10:04. For this deployment using early success criteria, the 10:12 alarm cannot trigger automatic rollback of that completed deployment. Preparing the database does not reactivate that rollback mechanism; a separate recovery action is required.
B - Incorrect: B correctly uses a separate deployment for the late failure and retains reconciliation of external effects. Its ordering is unsafe. A cannot process existing CAPTURED_V2 records, yet the procedure resumes processing before converting them. Pausing workers during redeployment does not solve that incompatibility once processing resumes. The tested conversion must make the retained records compatible before A processes them.
C - Correct: C correctly handles all three boundaries. Before B completed, the configured failure mechanisms could trigger rollback to A, which had previously reached COMPLETED. After B completed, restoring A requires a separate controlled deployment. The procedure first pauses processing and converts incompatible records without losing required data, then restores the retained A artifact and configuration and validates behavior. It also treats captured payments and downstream message effects as separate reconciliation and compensation work.
D - Incorrect: D correctly distinguishes the earlier automatic rollback window from the later controlled

recovery and correctly establishes data compatibility before resuming A. However, neither the compatibility conversion nor application reversal compensates external effects. Captured payments and downstream actions remain real even when local records are readable by A and the service is healthy. The existing reconciliation runbook must identify those effects and apply authorized compensation

Visit us at: https://www.examsboost.com/test/sap-c03

Thank You for Trying Our Product

For More Information – Visit link below:

https://www.examsboost.com/

15 USD Discount Coupon Code:

G74JA8UF

FEATURES

- ✓ 90 Days Free Updates
- ✓ Money Back Pass Guarantee
- ✓ Instant Download or Email Attachment
- ✓ 24/7 Live Chat Support
- ✓ PDF file could be used at any Platform
- ✓ 50,000 Happy Customer

