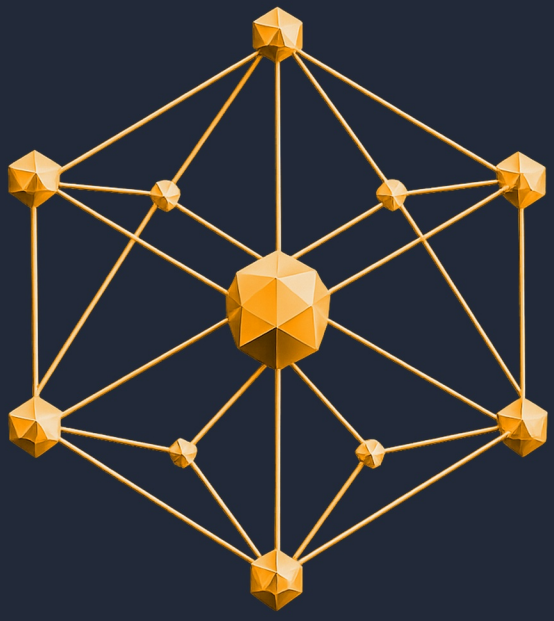
AWS Certified Cloud Practitioner

CLF-C02



Lite Edition • 3 Full-Length Exams

Sample from the 15-Exam Mastery Book

Fuad Efendi
Mastery Exam Prep

AWS Certified Cloud Practitioner

CLF-C02

Lite Edition • 3 Full-Length Exams

Sample from the 15-Exam Mastery Book

Fuad Efendi

Mastery Exam Prep
An imprint of Tokenizer Inc.

Through endless ages, I've journeyed afar,
Upon giants' shoulders, I reach for the stars.
What I know is not mine alone—
It was given to me, and I must pass it on.

— *Fuad Efendi*

Mastery Exam Prep
An imprint of Tokenizer Inc.

Edition & Version
2025 Lite Edition • Version 1.0 (last updated 2025-11-24)
This Lite Edition is a sample of the full 15-exam book and does not carry its own ISBN.

Updates & Contact
Content feedback or institutional/bulk licensing: editor@masteryexamprep.com.
Order or download issues: support@masteryexamprep.com.

How to Use This Book

Welcome! This Lite Edition gives you **3 full-length practice exams (Exams 13–15)** for the **AWS Certified Cloud Practitioner (CLF-C02)**. In the complete Mastery edition, you'll find all **15 full-length exams** built to the same standard; this sample lets you test the content, formatting, and device compatibility before committing to the full book. Each exam has **65 questions**, closely matching the style and pacing of the real CLF-C02, plus explanations designed to make your review **efficient and focused**.

Lite Edition vs Full Edition

This Lite Edition contains **Exams 13–15** exactly as they appear in the full 15-exam book.

- **What's included here:** 3 complete CLF-C02 practice exams (195 questions with explanations), using the same blueprint-driven design as the full edition.
- **What's in the full book:** Exams 1–12 plus these three, for a total of 15 exams and 975 questions, along with the complete curriculum appendix.
- **How to use both:** If you later purchase the full edition, treat Exams 13–15 as a spaced-repetition re-test near the end of your study plan, rather than "wasted" overlap.

How This Mirrors the Real Exam

- **Length and format:** 65 questions in 90 minutes, with a mix of multiple choice and multiple response items. AWS scores **50** of the 65 questions; the remaining items are unscored pilots.
- **Blueprint-weighted:** In the full 15-exam edition, questions reflect the official domain weightings, and the three Lite exams in this sample are built with the same blueprint-driven design:
  - **D1 Cloud Concepts — 24%**
  - **D2 Security and Compliance — 30%**
  - **D3 Cloud Technology and Services — 34%**
  - **D4 Billing, Pricing, and Support — 12%**
- **Question styles:** A realistic mix of scenario-based items (best solution, comparison, troubleshooting, optimization, quantitative) and knowledge items (definitions, limits/facts, principles). Multiple response items appear where they naturally fit the blueprint. The overall mix is tuned for **face validity** and realistic pacing.

Answer Choice Visuals

- **Multiple choice** (one correct answer) is shown with **radio-style symbols** (○).
- **Multiple response** items in this book are always "Select TWO." They are

shown with **checkbox-style symbols** (☐). For readability, we usually omit "(Select TWO)" from the stem because the visual treatment makes the requirement clear.

Explanations: Read Without Googling

Each question includes an explanation that:

- Summarizes the underlying concept in plain language.
- Explains **why the correct option is right**.
- Explains **why each distractor is wrong** (scope, prerequisite, limit, cost, or trade-off).
- Identifies the relevant **Domain/Task** in the explanation heading (for example, *Domain 2 • Task 2.1*) so you can review weak areas systematically and cross-reference the appendix and online curriculum.

> **Try the question first.** For the best learning effect, answer the question under exam-like conditions before jumping to the explanation.

Trust and Transparency

What you can expect from these questions and explanations:

- We focus on core ideas and realistic trade-offs, not obscure trivia.
- For single-response items, exactly one option is correct.
- For multiple-response items, exactly two options form the correct set.
- For negative questions (for example, "INCORRECT," "AVOID"), the keyed option(s) are the only ones that violate AWS guidance within the stated constraints.
- Explanations name the specific limit, cost, behavior, or design principle that makes each distractor wrong.
- We write to **time-stable AWS behaviors** and call out volatility where it matters.
- AI helps with drafting and checking, but **AWS-certified human editors** are responsible for accuracy and clarity.

Found a problem or have a suggestion? Email **editor@masteryexamprep.com** and we will review and fix issues promptly.

For the latest official edition, errata, and digital updates to this book, as well as bonus materials (including free online guides and quick quizzes), please visit **MasteryExamPrep.com**. You'll also find links to our companion web application and the upcoming *Mastery Cloud* apps for iOS, Android, and web. For institutional or bulk licensing inquiries—or to share feedback and testimonials that can help shape future editions—please contact **editor@masteryexamprep.com**.

Are You Ready? The Decision Point

Use your last few timed exams as a readiness check:

- If your **last three** timed scores average ≥ **45/65 (≈ 69%)**, you are typically in the ready range for CLF-C02. AWS only scores 50 of the 65 questions, so your true score on the real exam may be a bit higher.
- Avoid training yourself to **65/65** on these specific questions. That usually means memorizing phrasing rather than building transferable judgment.
- Instead, rotate across different exams and revisit your weak **Domains/Tasks** until your performance is consistently above the decision point.

Practice Strategy

- **Chunk when busy:** When you are short on time, do 20–30 question sets with immediate review.
- **Space your review:** Revisit missed or uncertain questions after 24–48 hours to solidify learning.
- **Track by domain:** Use the Domain/Task breadcrumbs in explanations to map your misses to specific blueprint areas.
- **Simulate exam conditions:** Once you are above the decision point, take full 90-minute exams to practice pacing and focus.

Good luck on your CLF-C02 journey—and enjoy the practice. Treat each question as a small design review, not a trivia check: pause, identify the constraints, and decide what you'd do in a real AWS environment. If you stay honest about your weak spots and keep iterating, this book will not only help you pass the exam, but also make you more confident in day-to-day cloud decisions.

Exam 13

Question 13.1

A company adopts cloud services and sets a single primary success metric focused on delivering IT capabilities faster. Which metric best matches that goal?

○ **A.** Lower capital expenditure on data center hardware

○ **B.** Increase in number of security controls implemented

○ **C.** Shorter time-to-market for new features and services

○ **D.** Higher average CPU utilization across instances

Explanation→

Question 13.2

Which of the following S3-related security problems is AWS Trusted Advisor designed to identify and flag for remediation?

○ **A.** Buckets that already use server-side encryption with AWS KMS for all stored objects.

○ **B.** Network firewall rules applied to EC2 instances inside a VPC.

○ **C.** IAM password policies that do not require complexity or rotation.

○ **D.** Objects or buckets that are publicly accessible and allow unintended read or write access.

Explanation→

Question 13.3

A team stores 10 TB of rarely accessed objects in an archival storage class to save on monthly capacity costs. One month, users need to download several terabytes for analysis. Which storage pricing component will increase most directly because of these downloads?

○ **A.** Monthly capacity billing measured in GB-months for the stored objects.

○ **B.** Per-API request charges for uploading new objects (PUT requests).

○ **C.** Per-GB retrieval charges applied when reading objects from archival or infrequent tiers.

○ **D.** Lifecycle transition fees for moving objects between storage classes.

Explanation→

Question 13.4

A team's continuous integration job uses a fully managed build service to compile code. Builds start failing when the build container cannot download private packages from the project's hosted repository due to access denied errors. What is the most direct fix so the build service can retrieve the dependencies?

○ **A.** Give each developer a user policy that can access the package repository so their commits carry credentials during the build.

○ **B.** Move the package repository into the same VPC as the build service to avoid permission issues.

○ **C.** Update the build service's IAM role to grant permissions to access the private package repository.

○ **D.** Modify the project buildspec to skip dependency download and use prebuilt artifacts instead.

Explanation→

Question 13.5

A company is migrating on-premises Windows file shares to AWS. Users need SMB access, integration with Active Directory, preservation of NTFS permissions and Shadow Copies, and a managed option that can run across multiple Availability Zones. Which service best meets these requirements with minimal operational overhead?

○ **A.** Amazon Elastic File System (Amazon EFS)

○ **B.** Amazon FSx for Lustre

○ **C.** Amazon FSx for Windows File Server (Multi-AZ enabled)

○ **D.** Amazon FSx for OpenZFS

Explanation→

Question 13.6

Which statement best describes economies of scale in AWS Cloud pricing?

○ **A.** Costs decrease only when customers commit to 1-year or 3-year plans; without a commitment, the unit price does not change with usage.

○ **B.** At higher usage levels, unit cost increases due to throttling and required capacity reservations for peak traffic.

○ **C.** Economies of scale mean paying a flat, fixed monthly fee regardless of usage because AWS manages capacity.

○ **D.** Average unit cost decreases as total usage grows because fixed costs are spread over more units; AWS passes this on via tiered pricing and improved price-performance.

Explanation→

Question 13.7

Which statement best describes the purpose of an EC2 launch template used with an Auto Scaling group?

○ **A.** It tracks running instance counts and automatically increases or decreases capacity based on CloudWatch alarms.

○ **B.** It records API activity for the Auto Scaling group so administrators can audit lifecycle events and security changes.

○ **C.** It stores a reusable configuration for instances (AMI, instance type, networking, and tags) that the group uses when creating or replacing EC2 instances.

○ **D.** It provides a long-term schedule that forces instances to terminate at fixed times to control cost.

Explanation→

Question 13.8

Which AWS load balancer type is intended to pass traffic through to third-party virtual appliances (such as firewalls or intrusion-detection systems) while keeping traffic flow transparent for deep packet inspection?

○ **A.** Application Load Balancer — for HTTP/HTTPS content-based routing and layer 7 features.

○ **B.** Network Load Balancer — for high-performance, TCP/UDP level pass-through and low-latency connections.

○ **C.** Classic Load Balancer — legacy option that balances HTTP and TCP but lacks modern transparent appliance forwarding features.

○ **D.** Gateway Load Balancer — for transparent forwarding of traffic to third-party virtual appliances for inspection and processing.

Explanation→

Question 13.9

A developer deploys an AWS Lambda function that processes customer files. Which responsibility remains with the customer when using this serverless compute service?

○ **A.** Managing the operating system patches for the underlying servers that run the function.

○ **B.** Writing and securing the function code, configuring its permissions, and handling related data access.

○ **C.** Provisioning physical hardware and networking cables in the data center used by the function.

○ **D.** Scaling the Lambda service capacity by adding or removing virtual machines manually.

Explanation→

Question 13.10

A company is starting its first migration wave across several AWS accounts. Engineers are already launching resources, but tags are inconsistent and Finance cannot map spend to products. Security wants to prevent unapproved services and provide centralized, least-privilege access for employees. The migration lead wants a quick, foundational move that aligns with AWS Cloud Adoption Framework guidance and sets guardrails before scaling the migration. What should the team do next?

○ **A.** Prioritize Platform work: design the VPC with IPv6 and deploy a pilot using Amazon EC2 on Graviton and Amazon Aurora Serverless v2 to validate performance.

○ **B.** Enable Amazon CloudWatch metrics and alarms, configure Auto Scaling, and set up AWS Backup to improve availability and efficiency for initial workloads.

○ **C.** Establish a governance baseline with AWS Organizations and IAM Identity Center: standardize tagging and cost allocation, apply guardrails using Service Control Policies, and centralize least-privilege access with permission sets.

○ **D.** Turn on AWS CloudTrail and Amazon GuardDuty across all accounts to improve logging and threat detection for the migration wave.

Explanation→

Question 13.11

Which statement is WRONG about an Amazon Aurora Global Database deployment intended to give low-latency local reads across Regions?

○ **A.** The primary cluster is the only writable cluster and handles transactional writes.

○ **B.** Secondary clusters in other Regions are read-only and serve local read requests.

○ **C.** Storage is copied asynchronously so replicas can be promoted quickly if the primary fails.

○ **D.** All clusters can accept concurrent writes and automatically merge transactions across Regions.

Explanation→

Question 13.12

A company must move a production Oracle database to Amazon RDS for PostgreSQL with minimal downtime and preserve ongoing transactions during migration. Which AWS approach best meets this requirement?

○ **A.** Take a full export from Oracle, transform files, and import them into RDS during a maintenance window.

○ **B.** Use AWS Schema Conversion Tool alone to perform the migration without any data replication service.

○ **C.** Build a custom ETL pipeline with AWS Lambda functions to periodically copy changed rows to the target database.

○ **D.** Use AWS Database Migration Service to replicate data continuously with validation, cut over when synchronized.

Explanation→

Question 13.13

A company wants to keep using its on-premises backup software that writes to tape over iSCSI, but reduce media costs and retain backups for years in the cloud using Amazon S3 Glacier Deep Archive. Which AWS solution best meets this requirement with minimal changes?

○ **A.** AWS Storage Gateway Volume Gateway — iSCSI block volumes with point-in-time EBS snapshots; not a VTL for tape-based backups.

○ **B.** AWS Storage Gateway File Gateway — NFS/SMB shares backed by S3 with lifecycle transitions; does not emulate tapes or use iSCSI.

○ **C.** AWS Storage Gateway Tape Gateway — Virtual tape library (VTL) over iSCSI; tapes stored in S3 and archived to S3 Glacier Deep Archive.

○ **D.** Amazon S3 Lifecycle policies — Transition objects between classes; no VTL or iSCSI support for legacy backup software.

Explanation→

Question 13.14

A company runs EC2 instances in private subnets and must access AWS services without sending traffic over the public internet. The instances must remain non–internet-routable. Which TWO actions meet this requirement?

☐ **A.** Deploy a NAT Gateway in a public subnet so private subnets can reach AWS services without inbound exposure.

☐ **B.** Create a Gateway VPC endpoint for Amazon S3 to keep S3 traffic within the VPC without NAT/IGW.

☐ **C.** Enable VPC Flow Logs to monitor and block internet egress from private subnets.

☐ **D.** Use AWS PrivateLink interface endpoints for supported AWS services to maintain private, VPC-contained connectivity.

☐ **E.** Attach an Internet Gateway and restrict inbound access with security groups to allow outbound-only access to AWS services.

☐ **F.** Establish VPC peering with a VPC that has internet access to reach AWS services privately.

Explanation→

Question 13.15

Which definition best describes how Auto Scaling helps meet cloud workload needs while controlling expenses?

○ **A.** Automatically adjusts compute capacity up or down based on demand to maintain performance and reduce idle cost.

○ **B.** Provides a fixed pool of reserved instances to guarantee low latency at a predictable price.

○ **C.** Schedules long-term archival of data to lower storage costs without affecting compute resources.

○ **D.** Automatically encrypts data in transit and at rest to reduce security incident response time.

Explanation→

Question 13.16

A company has two on-premises offices near different metro fiber hubs and needs a private, low-latency link to an AWS VPC in a single Region. Which Direct Connect location selection best meets the requirement to minimize packet travel time between on-premises and AWS?

○ **A.** Choose the Direct Connect facility geographically closest to the larger office and establish a private virtual interface to the target Region.

○ **B.** Select the Direct Connect location closest to the majority of user traffic and connect with a private virtual interface in the Region hosting the VPC.

○ **C.** Use any available Direct Connect site and rely on public virtual interfaces to reach AWS endpoints, since public VIFs automatically reduce latency.

○ **D.** Pick a Direct Connect location in a different Region to force traffic over a Direct Connect gateway for path diversity and thus lower latency.

Explanation→

Question 13.17

A security team is evaluating how Amazon GuardDuty finds suspicious activity. Which statements about GuardDuty's detection inputs and methods are INCORRECT?

☐ **A.** It depends only on customer-written rules and manual allow/deny lists to raise findings.

☐ **B.** It does not employ machine learning or statistical modeling to detect unusual patterns.

☐ **C.** It leverages built-in threat intelligence about known bad IP addresses and domains.

☐ **D.** It analyzes API activity recorded by AWS CloudTrail as a signal source.

☐ **E.** It profiles typical behavior per resource to highlight deviations that may indicate threats.

Explanation→

Question 13.18

A company is migrating dozens of applications to AWS and wants to accelerate onboarding without sacrificing governance. Requirements:

- Separate Prod, Non-Prod, and Sandbox accounts provisioned in minutes
- Single sign-on for engineers
- Preventive and detective guardrails
- Centralized CloudTrail and AWS Config logs in a dedicated log account
- Consistent VPC design and encryption by default
- Ability to tailor controls by environment

Which approach is the BEST fit with minimal operational effort?

○ **A.** Use AWS Organizations with consolidated billing; manually create accounts, apply tag policies, aggregate AWS Config, store CloudTrail in the management account, and manage access with IAM users.

○ **B.** Keep one AWS account with multiple VPCs per environment, enforce KMS-by-default via IAM policies, send CloudTrail to CloudWatch Logs, and rely on AWS Security Hub for oversight.

○ **C.** AWS Control Tower landing zone with OUs and Account Factory; enable SCP/AWS Config guardrails, centralized CloudTrail to Log Archive, and IAM Identity Center SSO.

○ **D.** Combine IAM Identity Center and AWS Service Catalog to standardize provisioning; use AWS Config rules to block noncompliance; share a central S3 log bucket with teams.

Explanation→

Question 13.19

A company needs near-instant text search across millions of documents with fast lookup of specific words and phrases. Which AWS service is best suited because it uses an inverted index for rapid term lookup?

○ **A.** Amazon Athena — interactive query service for running SQL on data in S3.

○ **B.** Amazon OpenSearch Service — managed search cluster that builds inverted indexes for fast term-based queries.

○ **C.** Amazon RDS — managed relational database for structured transactional data.

○ **D.** Amazon Kinesis Data Streams — real-time data streaming for ingesting and processing events.

Explanation→

Question 13.20

Which statement best describes the benefit of a managed service that automatically adjusts capacity based on demand?

○ **A.** It guarantees zero downtime by replicating every request across multiple regions without cost impact.

○ **B.** It eliminates the need for application performance testing because resources are always sufficient.

○ **C.** It fixes software bugs by applying vendor updates to application code automatically.

○ **D.** It increases or decreases compute resources in response to traffic, reducing manual intervention and preventing overloads.

Explanation→

Question 13.21

A small startup only requires help with account setup and billing questions and does not want to pay for technical guidance or incident support. Which AWS Support plan best matches this need?

○ **A.** Developer Support

○ **B.** Business Support

○ **C.** Basic Support

○ **D.** Enterprise Support

Explanation→

Question 13.22

Which term best matches this definition in cloud economics: Per-unit cost decreases as total output grows because fixed costs are spread across more usage and variable costs drop through scale efficiencies?

○ **A.** Pay-as-you-go pricing

○ **B.** Elasticity

○ **C.** Economies of scale

○ **D.** Savings Plans

Explanation→

Question 13.23

A public website is receiving bursts of automated requests from a few client IPs causing short-term load spikes. Which Amazon WAF feature should you use to limit requests from a single client IP over time without blocking other clients?

○ **A.** Configure a rate-based rule to throttle requests from that client IP.

○ **B.** Create a managed rule group that inspects request bodies for SQL injection.

○ **C.** Use an IP set to permanently block the client IP address.

○ **D.** Apply a regex pattern set to reject requests with suspicious query strings.

Explanation→

Question 13.24

A developer reports they cannot grant an IAM role permissions to manage S3 objects, even though an administrator attached a broad inline policy to the role. The role has an IAM permissions boundary attached and still cannot perform

PutObject. Which single action will most directly resolve this troubleshooting issue?

○ **A.** Attach the S3 managed policy (AmazonS3FullAccess) to the IAM user who created the role.

○ **B.** Create a new role with the same inline policy and have the developer assume that role without changing any other settings.

○ **C.** Enable cross-account access to the bucket by modifying the bucket policy to allow the role's AWS account.

○ **D.** Update the role's permission boundary to include the necessary S3 actions on the target bucket.

Explanation→

Question 13.25

A startup wants to reduce wasted, always-on compute capacity that sits idle between events. Which TWO AWS approaches best follow a serverless pattern to avoid provisioning servers?

☐ **A.** Use AWS Lambda functions for event-driven processing that run only when invoked.

☐ **B.** Store infrequently accessed objects in Amazon S3 Intelligent-Tiering to lower storage cost.

☐ **C.** Provision EC2 instances with Reserved Instances to get lower hourly costs for steady workloads.

☐ **D.** Implement Auto Scaling groups with EC2 instances that add or remove servers based on demand.

☐ **E.** Choose Amazon DynamoDB with On-Demand capacity so the database scales and billing matches request traffic.

Explanation→

Question 13.26

Which AWS Health capability enables a delegated administrator to centrally aggregate and monitor account-specific security events across all accounts in an AWS Organizations environment?

○ **A.** AWS Health Organizational View with a delegated administrator in AWS Organizations

○ **B.** AWS Health public Service health view (Region-wide service status)

○ **C.** AWS Health Your account view (formerly Personal Health Dashboard)

○ **D.** Amazon GuardDuty findings dashboard

Explanation→

Question 13.27

Which AWS service is the primary mechanism for recording account API calls and management events so administrators can audit user and service activity?

○ **A.** AWS CloudTrail

○ **B.** Amazon CloudWatch Logs

○ **C.** AWS Config

○ **D.** AWS Identity and Access Management (IAM)

Explanation→

Question 13.28

An e-commerce company is building a serverless shopping cart service on Amazon DynamoDB for a global flash sale. Requirements:

- Single-digit millisecond latency at scale
- Fast per-user cart retrieval
- Microsecond read latency for a hot read path (e.g., price/availability lookups)
- Automatic removal of carts idle for 48 hours
- Low-latency access and active-active resilience across two AWS Regions

Which statement is NOT an appropriate use of DynamoDB features for these requirements?

○ **A.** Amazon DynamoDB table with a composite primary key (partition key userId, sort key productId) to query a user's current cart items efficiently.

○ **B.** Use DynamoDB Accelerator (DAX) to achieve microsecond write latency for cart updates during the flash sale.

○ **C.** Enable DynamoDB Time to Live (TTL) on an expiresAt attribute to automatically purge abandoned carts after 48 hours.

○ **D.** Use Amazon DynamoDB Global Tables to provide active-active replication across two Regions for global low-latency access and resilience.

Explanation→

Question 13.29

A company runs a stateless web application in two subnets across different Availability Zones behind an Application Load Balancer. An Amazon EC2 Auto Scaling group spans both AZs with a minimum of 2 instances. The database is a single-AZ Amazon RDS instance. During an AZ disruption that affected

the database's AZ, users saw errors and the site was unavailable, even though instances in the other AZ were healthy. What change will most directly prevent this outage in the future?

○ **A.** Increase the Auto Scaling group minimum capacity to 4 instances across both AZs.

○ **B.** Adjust Application Load Balancer health checks to route traffic away from the impaired AZ.

○ **C.** Convert the database to an Amazon RDS Multi-AZ deployment with a synchronous standby in another AZ and automatic failover.

○ **D.** Store user session state in Amazon DynamoDB to avoid in-memory sessions on instances.

Explanation→

Question 13.30

A company wants to prevent unintended public access to Amazon S3 buckets and also maintain a complete audit trail of who did what in their AWS account. Which TWO AWS controls should they enable?

☐ **A.** AWS Config rule to detect publicly accessible S3 buckets

☐ **B.** Amazon S3 Block Public Access to prevent public bucket and object access

☐ **C.** AWS WAF to filter malicious web requests to applications

☐ **D.** Amazon GuardDuty to continuously monitor for threats

☐ **E.** AWS Key Management Service (KMS) default encryption for S3

☐ **F.** AWS CloudTrail to record and retain API activity for auditing

Explanation→

Question 13.31

Which statement best defines envelope encryption when using AWS Key Management Service (AWS KMS) to protect data at rest?

○ **A.** All data is sent to AWS KMS, which encrypts and decrypts it inside hardware security modules; applications never handle encryption keys locally.

○ **B.** The KMS key material is exported to applications for local encryption so workloads can run offline without calling KMS.

○ **C.** A single KMS key encrypts all objects directly in the storage service, eliminating per-object data keys and extra metadata.

○ **D.** AWS KMS issues a unique data key used locally to encrypt the data; only the ciphertext and the data key encrypted under a KMS key are stored.

Explanation→

Question 13.32

A startup is onboarding battery-powered sensors to AWS IoT Core and will use MQTT for telemetry and remote commands. To keep the design aligned with MQTT's lightweight publish/subscribe model, which TWO actions should the team AVOID?

☐ **A.** Have sensors publish data to topics while back-end services subscribe to those topics.

☐ **B.** Open a separate one-to-one connection from each sensor to every consumer device to deliver messages directly.

☐ **C.** Use small, periodic MQTT messages to reduce overhead for constrained devices.

☐ **D.** Duplicate-publish the same telemetry multiple times to different destinations instead of using a single topic that multiple subscribers can consume.

☐ **E.** Subscribe devices to topics to receive asynchronous command messages from the cloud.

☐ **F.** Design topics by function (for example, telemetry and commands) so producers and consumers remain loosely coupled.

Explanation→

Question 13.33

A company stores millions of objects with unpredictable access patterns and wants minimal operational effort while keeping immediate access when an object is requested. Which Amazon S3 storage class best meets this requirement?

○ **A.** Amazon S3 Standard (frequently accessed, low latency)

○ **B.** Amazon S3 Intelligent-Tiering (automatic tiering to reduce cost without operational changes)

○ **C.** Amazon S3 Standard-Infrequent Access (lower storage cost, retrieval fee)

○ **D.** Amazon S3 Glacier Instant Retrieval (archival with very low-cost storage and millisecond retrieval)

Explanation→

Question 13.34

A healthcare startup runs on Amazon EC2, Amazon RDS, and Amazon S3 and is preparing for an audit. They must: (1) provide auditor-ready evidence of AWS-owned controls, and (2) put account-level terms in place to handle protected health information (PHI). Which TWO actions should they take?

☐ **A.** Use AWS Artifact Reports to download SOC 2 Type II and PCI DSS Attestation of Compliance as evidence of AWS-owned controls.

☐ **B.** Enable AWS Artifact to continuously monitor EC2, RDS, and S3 for HIPAA configuration compliance.

☐ **C.** Review and accept the HIPAA Business Associate Addendum in AWS Artifact Agreements.

☐ **D.** Rely solely on AWS Artifact documents to confirm the application is compliant.

☐ **E.** Publish AWS Artifact compliance reports on the company website for transparency.

☐ **F.** Use AWS Config to obtain ISO 27001 certificates for auditors.

Explanation→

Question 13.35

Select TWO actions that are appropriate uses of scheduled scaling for an Auto Scaling group:

☐ **A.** Use historical usage patterns to automatically predict future capacity needs without fixed times.

☐ **B.** Continuously adjust capacity to keep a specific CPU utilization percentage.

☐ **C.** Scale only when a metric threshold is breached, stepping up or down in defined increments.

☐ **D.** Increase instance count before predictable daily traffic peaks (for example, just prior to business hours).

☐ **E.** Lower the number of instances during known off-peak hours to reduce running costs.

Explanation→

Question 13.36

A company has completed its migration to AWS. It wants a partner to operate the environment with 24x7 monitoring, patching, incident response, and ongoing cost optimization reviews, using least-privilege access in the

company's AWS accounts. The company will keep its AWS Support Business plan for guidance. Which option best meets these needs?

○ **A.** Managed Service Provider (MSP) delivering ongoing operations under SLAs using AWS Systems Manager, Amazon CloudWatch, AWS CloudTrail, and AWS Config.

○ **B.** Systems Integrator (SI) to design the architecture, build a landing zone, and execute a migration project.

○ **C.** Independent Software Vendor (ISV) subscription through AWS Marketplace for a monitoring tool.

○ **D.** AWS Support Business plan to perform runbooks, patching, and incident response on the company's behalf.

Explanation→

Question 13.37

What is the primary purpose of using the AWS Well-Architected Tool for a workload?

○ **A.** To automatically deploy a production architecture that meets compliance requirements without human review.

○ **B.** To monitor real-time application performance metrics and send alerts for operational incidents.

○ **C.** To manage AWS billing and apply reserved pricing recommendations across accounts.

○ **D.** To evaluate a workload against the Well-Architected pillars, highlight risks, and get guidance to prioritise fixes.

Explanation→

Question 13.38

Which AWS service is purpose-built to improve network path performance and availability for globally distributed TCP/UDP applications?

○ **A.** AWS Global Accelerator

○ **B.** Amazon CloudFront

○ **C.** Amazon Route 53

○ **D.** Amazon CloudWatch

Explanation→

Question 13.39

A startup is launching a new mobile API with unpredictable traffic spikes. The team wants to minimize operational overhead (no servers to patch), scale automatically within minutes, pay only for actual usage, and have high availability across multiple Availability Zones by default. Which approach best meets these goals while avoiding lift-and-shift overprovisioning?

○ **A.** Preprovision peak-capacity servers and run them 24/7 to handle spikes.

○ **B.** Use AWS Global Accelerator alone to absorb spikes without changing the backend.

○ **C.** Rely on AWS Budgets and AWS Cost Explorer to control costs while keeping the current design.

○ **D.** Amazon CloudFront + Amazon S3 for static content; Amazon API Gateway + AWS Lambda for API logic; Amazon DynamoDB with automatic scaling.

Explanation→

Question 13.40

Which statement best describes how metered pricing helps organizations pay only for the resources they use?

○ **A.** Customers pay a fixed monthly fee regardless of how much they use services, providing predictable billing but not usage alignment.

○ **B.** Organizations reserve capacity for a long term and receive a discount, which charges based on committed capacity rather than measured consumption.

○ **C.** Billing is calculated from measured resource consumption (for example, compute-hours or data transferred), so charges reflect actual use over time.

○ **D.** Costs are allocated per named user account so each individual pays only for their own activity, independent of resource metrics.

Explanation→

Question 13.41

A company runs customer-facing APIs in an AWS Region. They must: 1) deliver single-digit millisecond latency for a real-time processing component to an on-premises system in a nearby metro area, and 2) increase resilience against data center failures within the Region. Which combination of actions will best meet these requirements?

☐ **A.** Choose the Region with the most Availability Zones to guarantee the lowest latency to users and on-premises systems.

☐ **B.** Create multiple subnets within one Availability Zone to improve resilience against data center failures.

☐ **C.** Deploy duplicate environments in two separate AWS Regions to achieve high availability within the Region.

☐ **D.** Run all production resources in a single Local Zone to meet both latency and availability needs.

☐ **E.** Deploy the application across at least two Availability Zones in the parent Region using an Elastic Load Balancer and Amazon RDS Multi-AZ.

☐ **F.** Opt in to the nearest AWS Local Zone and run the real-time component on Amazon EC2 with Amazon EBS in that Local Zone.

Explanation→

Question 13.42

A media company needs to speed up delivery of images and videos to viewers worldwide. Which CloudFront feature best meets this requirement?

○ **A.** Large regional storage centers intended for long-term archival of original media files.

○ **B.** The origin servers that permanently host the original images and videos.

○ **C.** Private network endpoints used to create VPN connections between data centers.

○ **D.** Physical sites that cache frequently requested files near viewers to lower latency and reduce load on the origin.

Explanation→

Question 13.43

A company moving applications to AWS faces slow decision-making and stalled migration milestones. Which action most directly removes organizational barriers and ensures timely cloud adoption decisions?

○ **A.** Run a comprehensive role-based cloud training program for all engineers before migrating any workloads.

○ **B.** Assign a senior leader as visible sponsor with clear decision authority and regular status reviews.

○ **C.** Implement automated technical guardrails and IAM policies to enforce best practices across accounts.

○ **D.** Create financial incentives tied to cost savings and publish adoption KPIs monthly.

Explanation→

Question 13.44

An online learning platform runs a stateless web tier on Amazon EC2 Auto Scaling across two Availability Zones in one Region. Daily traffic shifts between AZs. The team uses Linux/Unix with default tenancy and sometimes changes instance sizes within the same family. They want to reduce costs, keep elasticity, and avoid locking capacity to a specific AZ. Which purchase choice best meets these goals?

○ **A.** Zonal Reserved Instances in one AZ for a specific instance size; includes a capacity reservation and discount only in that AZ.

○ **B.** Zonal Reserved Instances split evenly across both AZs for the exact size; provides capacity reservations in each AZ with discounts tied to each AZ.

○ **C.** Regional Reserved Instances for Linux/Unix (default tenancy) in the chosen instance family; discount applies across all AZs with instance size flexibility; no capacity reservation.

○ **D.** Use only On-Demand Instances and rely on Auto Scaling to shift load between AZs to minimize cost.

Explanation→

Question 13.45

A compliance team uses AWS Config to evaluate resource conformity and wants to enforce fixes when violations occur. Which statement about Config remediation is NOT correct?

○ **A.** You can configure automatic remediation to run a Lambda function when a resource is marked noncompliant.

○ **B.** All remediation requires manual approval before any corrective action can be executed by Config.

○ **C.** Remediation actions can be grouped and deployed as reusable policy packages for consistent enforcement across accounts.

○ **D.** Remediation can be used to restore desired configurations when drift is detected by an evaluation.

Explanation→

Question 13.46

A company wants to reduce the chance that a single datacenter failure makes its web application unavailable. Which design choice best addresses this single point of failure concern?

○ **A.** Deploy redundant application instances in multiple Availability Zones and keep data synchronized.

○ **B.** Run all application servers in one AZ but use larger instance types for capacity.

○ **C.** Store a backup of the database in the same AZ and perform daily manual restore tests.

○ **D.** Use a single, highly available EC2 instance with local storage and scheduled snapshots to S3.

Explanation→

Question 13.47

Which statement best describes the purpose of an IAM role's trust policy in AWS?

○ **A.** It defines the actions, resources, and conditions the role is allowed to perform in AWS.

○ **B.** It encrypts temporary credentials and manages automatic rotation for the role.

○ **C.** It allows target services like Amazon S3 or AWS Lambda to grant the role access using a resource-based policy.

○ **D.** It specifies the trusted principals (for example, lambda.amazonaws.com, ec2.amazonaws.com, or another AWS account) that can assume the role via AWS STS.

Explanation→

Question 13.48

Which post-migration activity focuses on changing the sizes or types of compute and storage resources so they match actual workload demand and reduce cost?

○ **A.** Purchasing long-term capacity commitments such as Reserved Instances or Savings Plans to get discounts.

○ **B.** Adjusting instance sizes and storage configurations to better reflect observed usage patterns.

○ **C.** Configuring Auto Scaling to add or remove instances automatically based on load.

○ **D.** Moving infrequently accessed objects to a cheaper storage tier to lower storage charges.

Explanation→

Question 13.49

A company exposes a partner-facing API over the internet. Security requires that both the client and the server present X.509 certificates during the HTTPS handshake. Which AWS-supported mechanism directly satisfies this two-way certificate authentication requirement?

○ **A.** Standard TLS with only a server certificate

○ **B.** MACsec on an AWS Direct Connect link

○ **C.** Mutual TLS (mTLS) for the API connection

○ **D.** IPSec site-to-site VPN between the networks

Explanation→

Question 13.50

A startup runs a WordPress site on Amazon Lightsail. To increase availability, they will add a Lightsail load balancer and place instances in different Availability Zones. They also schedule one automated snapshot per day for each instance. During a 28-day month, what is the minimum number of Lightsail instances they should run, and how many instance snapshots will be created for the month?

○ **A.** Two Amazon Lightsail instances across two Availability Zones; 28 snapshots in 28 days

○ **B.** Three Amazon Lightsail instances across three Availability Zones; 84 snapshots in 28 days

○ **C.** Two Amazon Lightsail instances in different Availability Zones behind a Lightsail load balancer; 56 snapshots in 28 days

○ **D.** One Amazon Lightsail instance with a static public IP; 28 snapshots in 28 days

Explanation→

Question 13.51

Which Amazon S3 storage class is designed for infrequently accessed objects that are stored in a single Availability Zone to reduce cost?

○ **A.** S3 Standard (frequent-access, multi-AZ high availability)

○ **B.** S3 Standard-IA (infrequent access with rapid retrieval across multiple AZs)

○ **C.** S3 One Zone-IA (infrequent access in a single AZ)

○ **D.** S3 Glacier Instant Retrieval (archival with millisecond retrieval)

Explanation→

Question 13.52

A regulated business must keep an immutable, searchable record of every API call and account activity across its AWS account for 365 days to support audits. Which AWS service is designed to provide these immutable API activity and event logs for that retention requirement?

○ **A.** Amazon CloudWatch Logs — collects metrics and application logs for monitoring and short-term troubleshooting.

○ **B.** AWS Config — records resource configuration changes to evaluate compliance over time.

○ **C.** AWS Security Hub — aggregates security findings and compliance posture from multiple services.

○ **D.** AWS CloudTrail — records and preserves account API activity and events as immutable logs.

Explanation→

Question 13.53

Which statement best defines an AWS cost allocation tag?

○ **A.** Key–value metadata on AWS resources that, when activated in Billing and Cost Management, become reporting dimensions in Cost Explorer, AWS Budgets, and the Cost and Usage Report (CUR).

○ **B.** An AWS Organizations policy type that enforces required tag keys and allowed values across accounts to standardize tagging.

○ **C.** A Cost Management feature that groups multiple tag values and AWS accounts into logical business views for reporting and chargeback.

○ **D.** A security control that automatically blocks creation of untagged EC2, RDS, and S3 resources at launch time.

Explanation→

Question 13.54

A team sees sudden monthly network charges after moving a web API to another AWS Region. Which cause most likely explains the increased per-GB billable traffic?

○ **A.** Clients in one Region are calling an API deployed in a different Region, causing cross-Region data transfer charges.

○ **B.** Traffic between instances in the same Availability Zone is being billed per GB because private IP addresses were used.

○ **C.** All data leaving AWS to the public internet is free when sourced from an EC2 instance in any Region.

○ **D.** VPC peering automatically routes traffic over a dedicated Direct Connect link, avoiding transfer fees.

Explanation→

Question 13.55

Select TWO AWS-native interactive sign-in activities from the list where adding multi-factor authentication (MFA) is most appropriate to reduce the risk of credential compromise and protect sensitive AWS access.

☐ **A.** Signing in as the account root user to perform billing or account-level changes

☐ **B.** Using long-lived programmatic access keys for automated scripts and services

☐ **C.** Interactive sign-in by an IAM user who manages account-wide administrative policies

☐ **D.** Accepting a federated login session from an external identity provider for a short-lived user

☐ **E.** Assuming a cross-account role for a brief, limited-privilege task

☐ **F.** Temporarily rotating a user password or updating non-sensitive profile settings

Explanation→

Question 13.56

Select TWO options that are NOT appropriate uses of an in-memory caching service like Amazon ElastiCache.

☐ **A.** Cache frequent database read results to reduce load on the primary database and lower read latency.

☐ **B.** Store ephemeral user session state for a web application to speed access.

☐ **C.** Keep long-term archival records (compliance retention) as the primary copy of data.

☐ **D.** Use the cache as the authoritative, durable store for transactional data requiring strong consistency.

☐ **E.** Offload many concurrent, low-latency read requests from a busy database to improve throughput.

Explanation→

Question 13.57

A security analyst needs a quick visual map that shows how AWS resources and identities interacted during a suspicious incident to help find the root cause. Which Detective capability best provides an interactive relationship map of entities and their activities?

○ **A.** Behavior graphs — interactive relationship maps that link resources, principals, and activities for investigation.

○ **B.** CloudWatch metrics dashboards showing aggregated performance and utilization over time.

○ **C.** CloudTrail event logs that list API calls for auditing and compliance purposes.

○ **D.** GuardDuty alerts that prioritize suspicious findings detected across the account.

Explanation→

Question 13.58

Which purpose best describes an External ID when granting another AWS account temporary access to your resources via an IAM role?

○ **A.** A long-lived secret key stored in the role that authenticates the requester for ongoing access.

○ **B.** A value supplied by the role requester that helps prevent the confused-deputy problem during cross-account role assumption.

○ **C.** A tag added to resources that automatically grants permissions to a specific external account.

○ **D.** A policy condition that encrypts temporary credentials returned by STS for the requester.

Explanation→

Question 13.59

An enterprise rushed from assess to rehosting and skipped mobilize. After a few migration waves, they now have many AWS accounts with inconsistent IAM, ad hoc VPC designs, and logs scattered across accounts. Leadership fears rework and security gaps will erode migration ROI. What is the most effective next step to optimize outcomes before scaling migration?

○ **A.** Consolidate all workloads into a single AWS account and apply a centralized IAM policy to reduce complexity.

○ **B.** Proceed with rehosting using AWS Application Migration Service now and defer IAM, VPC, and logging standardization until after cutover.

○ **C.** Purchase Compute Savings Plans immediately to improve costs while keeping current account structures unchanged.

○ **D.** Establish an AWS Control Tower landing zone with AWS Organizations, IAM Identity Center, SCP guardrails, centralized CloudTrail/CloudWatch logging, and KMS encryption to standardize security and access across accounts.

Explanation→

Question 13.60

Which durability level does Amazon S3 promise for objects stored across multiple Availability Zones?

○ **A.** 99.99% durability

○ **B.** Six nines (99.9999%) durability

○ **C.** No durability guarantee—customers must replicate data themselves

○ **D.** 11 nines (99.999999999%) durability

Explanation→

Question 13.61

Which statement best describes the operational responsibility difference between running a database on Amazon EC2 versus using Amazon RDS?

○ **A.** Both EC2 and RDS require the customer to install and patch the underlying host operating system manually.

○ **B.** On EC2 you manage the operating system and database software; with RDS AWS handles the database engine and OS patching for you.

○ **C.** RDS gives full root access to the database host so customers can patch the OS, while EC2 does not allow OS changes.

○ **D.** EC2 automatically updates database engines, whereas RDS requires customers to update engine versions and operating system packages.

Explanation→

Question 13.62

A company has 30 AWS accounts in AWS Organizations. All workloads run only in us-east-1. Security leadership wants: 1) a single, centralized view of security findings across all accounts, and 2) automatic isolation of EC2 instances when high-severity findings are generated by integrated services like Amazon

GuardDuty or Amazon Inspector. The team prefers native AWS integrations. Which TWO actions should the team take?

☐ **A.** Designate a delegated administrator for AWS Security Hub in AWS Organizations to centrally manage and view findings across member accounts.

☐ **B.** Enable Amazon Macie automatic quarantine in Security Hub to remove sensitive S3 data exposures without additional configuration.

☐ **C.** Set a global aggregation Region in AWS Security Hub so you do not need to enable Security Hub in other Regions.

☐ **D.** Configure AWS Config conformance packs so failed Security Hub controls are resolved automatically by the service.

☐ **E.** Create an Amazon EventBridge rule that filters high-severity Security Hub findings and invokes an AWS Systems Manager Automation runbook to isolate affected EC2 instances.

☐ **F.** Use AWS CloudTrail event rules to route Security Hub findings directly to remediation workflows.

Explanation→

Question 13.63

A retailer serves its static website directly from a single Amazon S3 bucket in one Region. Global users report high latency. The company also wants to lower S3 origin load to reduce cost and prefers minimal changes. Which is the BEST solution?

○ **A.** Enable S3 Intelligent-Tiering on the bucket to auto-move objects between access tiers for better latency and lower transfer costs.

○ **B.** Put Amazon CloudFront in front of the S3 bucket to cache objects at edge locations, cutting latency and reducing direct S3 requests.

○ **C.** Serve files through Amazon API Gateway backed by AWS Lambda for automatic scaling without servers.

○ **D.** Use AWS WAF to filter requests so fewer reach S3, improving latency for global users.

Explanation→

Question 13.64

Select TWO actions that directly use budget-based, threshold-driven forecasting to help control spend across multiple AWS accounts:

☐ **A.** Create budgets that send alerts when forecasted month-end spend exceeds a chosen percentage of budgeted amount.

☐ **B.** Configure automated responses tied to budget thresholds to trigger operational controls when forecasts predict overspend.

☐ **C.** Use Trusted Advisor checks exclusively to forecast next month's spend for all accounts.

☐ **D.** Apply chargeback tags to resources and then wait for end-of-month cost allocation reports to allocate spend.

☐ **E.** Rely only on the monthly consolidated invoice to detect unexpected account-level overruns after the billing cycle closes.

☐ **F.** Depend on manual review of CloudWatch metrics as the only mechanism to forecast account billing.

Explanation→

Question 13.65

Under the shared responsibility model for AWS Lambda, which task is the customer's responsibility?

○ **A.** Scan and remediate vulnerabilities in the AWS Lambda deployment package or container image and keep dependencies updated.

○ **B.** Patch the underlying operating system and managed runtimes for AWS Lambda functions.

○ **C.** Provide multi-AZ availability and automatic scaling for the Lambda compute fleet.

○ **D.** Create and manage the elastic network interfaces (ENIs) when a Lambda function is attached to a VPC.

Explanation→

Exam 13 — Answer Key

Question	Correct answer(s)
13.1	C
13.2	D
13.3	C
13.4	C
13.5	C
13.6	D
13.7	C
13.8	D
13.9	B
13.10	C
13.11	D
13.12	D
13.13	C
13.14	B, D
13.15	A
13.16	B
13.17	A, B
13.18	C
13.19	B
13.20	D
13.21	C
13.22	C
13.23	A
13.24	D
13.25	A, E
13.26	A
13.27	A

Question	Correct answer(s)
13.28	B
13.29	C
13.30	B, F
13.31	D
13.32	B, D
13.33	B
13.34	A, C
13.35	D, E
13.36	A
13.37	D
13.38	A
13.39	D
13.40	C
13.41	E, F
13.42	D
13.43	B
13.44	C
13.45	B
13.46	A
13.47	D
13.48	B
13.49	C
13.50	C
13.51	C
13.52	D
13.53	A
13.54	A
13.55	A, C
13.56	C, D
13.57	A

Question	Correct answer(s)
13.58	B
13.59	D
13.60	D
13.61	B
13.62	A, E
13.63	B
13.64	A, B
13.65	A

Exam 13 Explanations

13.1 — Business drivers: agility, innovation, cost (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 13.1

A company adopts cloud services and sets a single primary success metric focused on delivering IT capabilities faster. Which metric best matches that goal?

Options & Rationales

A ✗ — Lower capital expenditure on data center hardware

Rationale: Lower capital costs can be a benefit of cloud but do not directly indicate how quickly the organization delivers new services.

B ✗ — Increase in number of security controls implemented

Rationale: More controls may improve security posture but do not measure the speed of delivering IT services.

C ✓ — Shorter time-to-market for new features and services

Rationale: Directly measures how quickly the business delivers new capabilities, aligning with faster provisioning and iterative deployments enabled by cloud.

D ✗ — Higher average CPU utilization across instances

Rationale: Resource utilization shows efficiency but does not directly reflect how fast new features reach users.

Explanation

The core objective in this scenario is accelerating how fast the organization delivers IT capabilities to customers. The most relevant success metric is shorter time-to-market because it directly measures delivery speed—how quickly new features, fixes, or services move from idea to production. Cloud characteristics such as rapid provisioning, iterative deployment cycles, and managed services reduce friction in delivery, making time-to-market a good single indicator of success. Operational controls like Auto Scaling and capacity optimization help sustain faster delivery while controlling cost, but they are supporting mechanisms rather than the primary metric.

Why the correct choice fits and others do not. Shorter time-to-market matches the stated goal because it quantifies delivery speed. Lower capital expenditure is a financial benefit of cloud migration but does not prove faster delivery. Increasing the number of security controls improves protection but can slow delivery if not balanced; it is not a speed metric. Higher average CPU utilization reflects resource efficiency but may indicate overcommitment and does not show how fast the business releases capabilities.

Key Concepts

- **Time-to-market:** Measures the elapsed time from a product idea or change request to its availability to users; a direct indicator of delivery speed.
- **Agility in cloud:** Refers to faster provisioning and the ability to iterate quickly, enabling shorter release cycles and experimentation without heavy upfront investment.
- **Supporting controls:** Automation and capacity optimization (for example, scaling and right-sizing) help maintain rapid delivery while managing costs and performance.

Common Pitfalls

- Confusing cost reduction or resource efficiency with delivery speed; they are related but distinct outcomes.
- Treating supporting operational metrics (like utilization or number of controls) as primary success measures instead of focusing on outcomes that reflect user-facing delivery cadence.

Glossary

- **Auto Scaling:** A mechanism to automatically adjust compute capacity in response to demand, helping maintain performance and support rapid delivery.

13.2 — Trusted Advisor security checks (D2.T2.4)

Domain 2 • Task 2.4

↑ Back to Question 13.2

Which of the following S3-related security problems is AWS Trusted Advisor designed to identify and flag for remediation?

Options & Rationales

A ✗ — Buckets that already use server-side encryption with AWS KMS for all stored objects.

Rationale: Trusted Advisor encourages secure configurations and would not flag a bucket that already has server-side encryption correctly enabled as a problem needing remediation.

B ✗ — Network firewall rules applied to EC2 instances inside a VPC.

Rationale: Firewall rules for EC2 (security groups or NACLs) are not an S3 permission issue; Trusted Advisor reports S3 access misconfigurations specifically for buckets/objects.

C ✗ — IAM password policies that do not require complexity or rotation.

Rationale: IAM password policy settings are an identity configuration topic; Trusted Advisor's S3 checks concentrate on bucket and object access controls rather than password policy strength.

D ✓ — Objects or buckets that are publicly accessible and allow unintended read or write access.

Rationale: Trusted Advisor inspects S3 bucket policies and ACLs and reports buckets that are exposed to public access so owners can restrict permissions.

Explanation

AWS Trusted Advisor performs automated inspections of common account-level security controls and surfaces prioritized recommendations. For Amazon S3, its security checks focus on misconfigurations that could expose data or weaken protection, including public access and bucket-level encryption settings. Detecting publicly accessible buckets or objects is a primary purpose because public read/write settings directly allow outside users to retrieve or modify stored content.

Trusted Advisor does not scan for every possible configuration gap, but it concentrates on high-impact security issues. Other concerns, such as fine-grained IAM password complexity or non-security-related configuration details, are handled by different tools, checks, or separate AWS services and best-practice reviews.

Key Concepts

- **S3 access controls:** S3 access is governed by bucket policies, ACLs, and block-public-access settings that determine who can read or write objects. Trusted Advisor evaluates these to find overly permissive exposure.
- **Public exposure risk:** A publicly accessible bucket or object allows anyone on the internet to access data, creating a high-priority security issue that Trusted Advisor flags for remediation.
- **Scope of checks:** Trusted Advisor's S3 security checks emphasize access and encryption misconfigurations rather than all possible hardening measures or identity policy details.

Common Pitfalls

- Assuming Trusted Advisor will report every S3 best-practice item; it prioritizes high-impact exposure and protection issues rather than every tuning knob.
- Confusing S3 access findings with unrelated security areas such as EC2 network firewall rules or IAM password policies, which are separate concerns.

Glossary

- **Bucket policy:** A JSON document attached to an S3 bucket that defines permissions for principals and actions.
- **Block Public Access:** An account- or bucket-level setting that helps prevent public access to S3 data.

13.3 — Storage pricing components: requests/retrieval (D4.T4.1)

Domain 4 • Task 4.1

↑ Back to Question 13.3

A team stores 10 TB of rarely accessed objects in an archival storage class to save on monthly capacity costs. One month, users need to download several terabytes for analysis. Which storage pricing component will increase most directly because of these downloads?

Options & Rationales

A ✗ — Monthly capacity billing measured in GB-months for the stored objects.

Rationale: Capacity charges are based on how much data is stored and the time retained. Download activity does not change the stored GB-months unless objects are deleted or moved.

B ✗ — Per-API request charges for uploading new objects (PUT requests).

Rationale: PUT request fees relate to writing objects. Downloading existing objects does not generate additional PUT charges.

C ✓ — Per-GB retrieval charges applied when reading objects from archival or infrequent tiers.

Rationale: Retrieving data from lower-cost archival or infrequent tiers typically incurs a per-GB read fee. Large downloads will therefore increase retrieval costs more than other components.

D ✗ — Lifecycle transition fees for moving objects between storage classes.

Rationale: Lifecycle transition fees occur when objects are moved between classes by policies. User downloads do not directly trigger lifecycle transition charges.

Explanation

Downloading large amounts of data from a low-cost, rarely accessed storage tier mainly affects the charges associated with reading that data. Storage cost is composed of several separate components: the baseline capacity cost for how much data is stored over time, per-operation request fees for API calls (such as PUT and GET), fees charged when data is retrieved from lower-cost tiers, and potential charges for lifecycle transitions or early deletion. In this scenario the stored volume remains the same and no new objects are uploaded or transitioned, so the dominant extra cost comes from the per-GB retrieval pricing applied when objects are read from infrequent or archival tiers.

Why other choices are incorrect. Monthly capacity billing stays constant because the stored 10 TB and its retention did not change. PUT request charges apply only to uploads, not downloads. Lifecycle transition fees occur when objects are programmatically moved between classes, which is not described here.

Key Concepts

- **Capacity billing:** Charges based on data volume and retention time (GB-month). Capacity fees change if stored size or retention changes.

- **Request fees:** Small per-operation charges for API calls; writes and reads are billed separately and affect cost when many operations occur.
- **Retrieval charges:** Additional per-GB fees applied when reading data from lower-cost classes designed for infrequent access.

Common Pitfalls

- A common mistake is to assume downloads increase storage capacity fees; they only affect capacity when data is added or retained longer. Confusing upload (PUT) fees with download costs is another frequent error.

13.4 — Developer tools: CodeBuild automation (D3.T3.8)

Domain 3 • Task 3.8 ↑ Back to Question 13.4

A team's continuous integration job uses a fully managed build service to compile code. Builds start failing when the build container cannot download private packages from the project's hosted repository due to access denied errors. What is the most direct fix so the build service can retrieve the dependencies?

Options & Rationales

A ✗ — Give each developer a user policy that can access the package repository so their commits carry credentials during the build.

Rationale: Developer user permissions do not propagate to the build container. The build service uses its own role at runtime, so developer policies won't resolve container access errors.

B ✗ — Move the package repository into the same VPC as the build service to avoid permission issues.

Rationale: Network co-location may change connectivity but does not address authorization; access denied indicates an IAM/permission problem, not a VPC placement issue.

C ✓ — Update the build service's IAM role to grant permissions to access the private package repository.

Rationale: The build environment runs with a service role; giving that role the necessary permissions to read from the hosted package store allows the build container to fetch dependencies securely.

D ✗ — Modify the project buildspec to skip dependency download and use prebuilt artifacts instead.

Rationale: Skipping dependency installation avoids the immediate error but removes required libraries and breaks the build output; it's not a proper fix for permission failures.

Explanation

When a managed build job cannot fetch private packages, the root cause is usually authorization: the build container operates using a service role that

must be granted explicit permission to access other AWS resources. Fixing the role's permissions is the most direct and secure remedy because it gives the runtime identity the ability to call repository APIs and retrieve artifacts.

Key Concepts

- **Build service runtime role:** The build environment assumes an IAM role while running; that role's policies determine which resources the build can access.
- **Resource permissions vs. network placement:** Authorization (IAM) controls access even when network connectivity exists; network changes rarely fix access denied errors.
- **Least privilege:** Grant only the permissions the build needs (read-only access to the package repository) rather than broad user-level credentials.

Why the correct action works. Updating the build service's IAM role to include the required read permissions (for example, token retrieval and package read actions) gives the container the authority to download private dependencies during the build. This preserves automation, follows security best practices by using a dedicated runtime identity, and avoids embedding developer credentials.

Why the other options are incorrect. Giving developers repository access does not help the build process because the build does not run under a developer's account. Changing VPC placement addresses connectivity but not authorization errors. Skipping dependency downloads prevents the immediate failure but breaks the build correctness and masks the underlying permission issue.

Common Pitfalls

- Assuming a developer's permissions apply to automated builds; they do not. Ensure the build role has needed rights.
- Confusing network connectivity problems with IAM authorization failures; check error messages and logs for "access denied" vs. networking timeouts.

13.5 — FSx offerings incl. Windows File Server (D3.T3.6)

Domain 3 • Task 3.6 ↑ Back to Question 13.5

A company is migrating on-premises Windows file shares to AWS. Users need SMB access, integration with Active Directory, preservation of NTFS permissions and Shadow Copies, and a managed option that can run across multiple Availability Zones. Which service best meets these requirements with minimal operational overhead?

Options & Rationales

A ✗ — Amazon Elastic File System (Amazon EFS)

Rationale: Provides NFS for Linux, not SMB; does not support Windows-specific features like NTFS ACLs or Shadow Copies needed for Windows home directories.

B ✗ — Amazon FSx for Lustre

Rationale: Optimized for HPC/ML throughput with sub-millisecond latency and S3 integration; not intended for everyday Windows SMB collaboration shares.

C ✓ — Amazon FSx for Windows File Server (Multi-AZ enabled)

Rationale: Native Windows file system with SMB, joins Active Directory, honors NTFS ACLs and Shadow Copies; Multi-AZ supports high availability for Windows shares.

D ✗ — Amazon FSx for OpenZFS

Rationale: Delivers NFS for Linux with snapshots and cloning; lacks SMB and Windows Shadow Copies for user shares.

Explanation

Selecting the right managed file service starts with matching the protocol and operating system expectations of the workload. The Amazon FSx family offers fully managed, enterprise-grade file systems that integrate with Amazon VPC, encrypt data at rest with AWS KMS, support automated backups through AWS Backup, and can be deployed for high availability across multiple Availability Zones. Each FSx type targets a distinct use case.

Windows home directories and departmental shares require SMB, tight integration with Active Directory, and preservation of Windows-native features such as NTFS access control lists and Shadow Copies. Amazon FSx for Windows File Server is a native Microsoft Windows file system that exposes SMB shares, can join Active Directory, and honors NTFS ACLs, Shadow Copies, and user quotas. It also offers Multi-AZ deployments for high availability—minimizing operational overhead versus self-managing file servers.

By contrast, Amazon EFS provides NFS for Linux clients and is not suitable for Windows SMB workloads or Windows-specific features like NTFS ACLs and Shadow Copies. Amazon FSx for Lustre is optimized for high-performance computing and machine learning, delivering very high throughput and sub-millisecond latencies with Amazon S3 integration; it is not meant for general collaboration file shares. Amazon FSx for OpenZFS delivers NFS for Linux with snapshotting and cloning for development and analytics, not SMB or Windows Shadow Copies.

Key Concepts

- **Protocol-to-workload fit:** SMB is for Windows file sharing; NFS commonly serves Linux clients. Choosing the wrong protocol breaks compatibility and management expectations.
- **FSx portfolio specialization:** Windows File Server (Windows SMB), Lustre (HPC/ML throughput, S3 integration), OpenZFS (NFS for Linux). Each targets a specific use case.
- **High availability:** Multi-AZ deployments improve reliability for shared file storage without the overhead of self-managed clustering.

Common Pitfalls

- Selecting Amazon EFS for Windows SMB workloads, which require Windows-native features and SMB.
- Using FSx for Lustre for general-purpose collaboration instead of HPC-focused use cases.

13.6 — Economies of scale on AWS pricing (D1.T1.4)

Domain 1 • Task 1.4 ↑ Back to Question 13.6

Which statement best describes economies of scale in AWS Cloud pricing?

Options & Rationales

A ✗ — Costs decrease only when customers commit to 1-year or 3-year plans; without a commitment, the unit price does not change with usage.

Rationale: Commitments like Savings Plans lower rates for predictable usage, but economies of scale also appear in automatic volume tiers and periodic on-demand price reductions.

B ✗ — At higher usage levels, unit cost increases due to throttling and required capacity reservations for peak traffic.

Rationale: Economies of scale imply the opposite trend: higher aggregate usage generally reduces per-unit cost. Throttling and reservations are not pricing principles.

C ✗ — Economies of scale mean paying a flat, fixed monthly fee regardless of usage because AWS manages capacity.

Rationale: AWS primarily uses pay-as-you-go pricing. Economies of scale reduce per-unit cost with volume; they do not convert services into flat-fee subscriptions.

D ✓ — Average unit cost decreases as total usage grows because fixed costs are spread over more units; AWS passes this on via tiered pricing and improved price-performance.

Rationale: This is the core definition: as demand aggregates, per-unit costs fall and are reflected in customer pricing through volume tiers and newer, more efficient offerings.

Explanation

Economies of scale describe how average cost per unit falls as total volume increases. In cloud terms, AWS aggregates demand across millions of customers and spreads large, largely fixed costs (data centers, networks, control plane software) over more units. As utilization, automation, and procurement efficiency improve, the per-unit cost declines and is reflected in customer pricing. This shows up through usage-based models and volume advantages: tiered storage pricing in Amazon S3, data transfer volume tiers, and periodic price/performance improvements (for example, Graviton-based instances).

Customers can also capture predictable-use discounts via Savings Plans, while still benefiting from on-demand models like per-second billing for EC2 and AWS Fargate.

The statement that average unit cost decreases as usage grows and that AWS passes this through via tiered pricing and improved price-performance is the correct definition. The idea that only 1- or 3-year commitments lower cost is incomplete: commitments help for steady usage, but economies of scale also appear in automatic S3 volume tiers and on-demand price reductions. The claim that unit cost increases with usage contradicts the core principle; while workloads may need architecture changes at scale, pricing per unit typically trends downward. Finally, a flat, fixed monthly fee regardless of usage does not represent AWS's pay-as-you-go approach and is unrelated to economies-of-scale pricing.

Key Concepts

- **Economies of scale:** Average unit cost falls as total volume increases due to spreading fixed costs and gaining operational efficiencies.
- **Usage-based and tiered pricing:** Per-unit rates can drop at higher volumes (for example, S3 storage tiers, data transfer tiers), reflecting scale benefits.
- **Commitments vs. flexibility:** Savings Plans lower unit rates for steady usage; on-demand and serverless still capture scale via periodic price cuts and per-second billing.

Common Pitfalls

- Assuming only prepaid commitments yield lower unit prices, ignoring automatic volume tiers and on-demand price/performance improvements.

13.7 — EC2 Auto Scaling for elasticity (D3.T3.3)

Domain 3 • Task 3.3 ↑ Back to Question 13.7

Which statement best describes the purpose of an EC2 launch template used with an Auto Scaling group?

Options & Rationales

A ✗ — It tracks running instance counts and automatically increases or decreases capacity based on CloudWatch alarms.

Rationale: Tracking counts and reacting to metrics is the role of a scaling policy, not the purpose of a launch template.

B ✗ — It records API activity for the Auto Scaling group so administrators can audit lifecycle events and security changes.

Rationale: API auditing is performed by services like CloudTrail; launch templates do not serve as audit logs.

C ✓ — It stores a reusable configuration for instances (AMI, instance type, networking, and tags) that the group uses when creating or replacing EC2 instances.

Rationale: A launch template defines instance settings such as AMI, instance type, networking, and tags so Auto Scaling can consistently launch or replace instances.

D ✗ — It provides a long-term schedule that forces instances to terminate at fixed times to control cost.

Rationale: Scheduled actions can change group size, but templates do not enforce termination schedules.

Explanation

A launch template is a saved set of instance configuration values that an Auto Scaling group references when it launches or replaces EC2 instances. It centralizes choices such as the Amazon Machine Image (AMI), instance type, key pair, security group settings, network settings, and resource tags so instances are created consistently without manual repetition. The template does not itself perform monitoring, scaling decisions, or auditing.

Why the correct choice fits. The correct description focuses on the template's role as a blueprint for instance attributes that Auto Scaling uses when creating or replacing instances. The other choices describe separate Auto Scaling concerns: scaling actions are driven by policies and CloudWatch metrics, auditing is handled by CloudTrail, and scheduled capacity changes are handled by scheduled actions rather than the template itself.

Key Concepts

- **Launch template:** A reusable definition of instance parameters used to create EC2 instances consistently for scaling operations.
- **Scaling policy vs. template:** A scaling policy defines when to add or remove instances based on metrics or schedules; the launch template defines how those instances are configured.
- **Audit and logging separation:** Recording API calls and lifecycle events is the responsibility of auditing services, not instance templates.

Common Pitfalls

- Confusing configuration with control: assuming the template makes scaling decisions rather than supplying instance settings leads to incorrect designs.
- Expecting templates to act as logs: treating them as an audit trail is incorrect because templates are blueprints, not records of operations.

Glossary

- **AMI:** Amazon Machine Image, a preconfigured operating system and application image used to launch an EC2 instance.
- **Scaling policy:** Rules that determine how an Auto Scaling group changes capacity in response to metrics or schedules.

13.8 — ALB vs NLB vs GWLB options (D3.T3.3)

Domain 3 • Task 3.3

↑ Back to Question 13.8

Which AWS load balancer type is intended to pass traffic through to third-party virtual appliances (such as firewalls or intrusion-detection systems) while keeping traffic flow transparent for deep packet inspection?

Options & Rationales

A ✗ — Application Load Balancer — for HTTP/HTTPS content-based routing and layer 7 features.

Rationale: This load balancer operates at the application layer and focuses on HTTP/HTTPS content routing, not transparent forwarding to third-party appliances.

B ✗ — Network Load Balancer — for high-performance, TCP/UDP level pass-through and low-latency connections.

Rationale: This option provides connection-level pass-through for performance and low latency but does not specialize in transparent forwarding to virtual appliance chains.

C ✗ — Classic Load Balancer — legacy option that balances HTTP and TCP but lacks modern transparent appliance forwarding features.

Rationale: The legacy Classic option supports basic layer 4 and layer 7 balancing but does not provide the appliance chaining and transparent forwarding capabilities required for third-party inspection.

D ✓ — Gateway Load Balancer — for transparent forwarding of traffic to third-party virtual appliances for inspection and processing.

Rationale: This load balancer is built specifically to steer traffic to virtual appliances while keeping flows transparent so appliances can inspect and modify packets as needed.

Explanation

AWS offers multiple load balancing alternatives that target different layers and use cases. One type is specifically intended to deliver traffic through third-party virtual appliances (for example, firewalls, intrusion detection systems, or traffic inspection devices) while preserving the original flow so those appliances can perform deep packet inspection and any necessary modification.

The Gateway Load Balancer is the service designed for this purpose: it forwards network traffic to virtual appliance instances in a transparent manner so appliances see and can act on the original traffic. This makes it suitable when an organization needs to insert specialized network functions into the path without breaking source/destination visibility.

Why the other options are not correct:

- Application Load Balancer focuses on application-layer (HTTP/HTTPS) content inspection and routing decisions, not chaining traffic through external virtual appliances.
- Network Load Balancer is optimized for very high throughput and low-latency connection-level pass-through (TCP/UDP) but does not provide

the appliance-chaining transparency and inspection workflow that gateway forwarding offers.
- Classic Load Balancer is an older, legacy option that provides basic layer 4 and limited layer 7 balancing; it lacks the modern appliance-forwarding capabilities.

Key Concepts

- **Transparent forwarding:** Sending traffic through intermediate devices while preserving source/destination details so those devices can inspect or modify packets.
- **Virtual appliance chaining:** The pattern of routing traffic to third-party software-based network functions (firewalls, IDS/IPS) deployed in the cloud.
- **Layer purpose alignment:** Choose load balancers based on whether you need application-layer routing, raw connection pass-through, or transparent appliance insertion.

Common Pitfalls

- Confusing high-throughput pass-through with transparent appliance chaining leads to choosing a Network Load Balancer when appliance visibility is required.
- Selecting an application-focused balancer when packet-level inspection by third-party appliances is needed.

13.9 — Responsibility shifts for serverless (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 13.9

A developer deploys an AWS Lambda function that processes customer files. Which responsibility remains with the customer when using this serverless compute service?

Options & Rationales

A ✗ — Managing the operating system patches for the underlying servers that run the function.

Rationale: The cloud provider handles OS patching and the managed runtime for Lambda; the customer does not manage underlying server OS patching.

B ✓ — Writing and securing the function code, configuring its permissions, and handling related data access.

Rationale: Customers are responsible for their application code, its configuration (including IAM permissions), and controlling access to any data the function uses.

C ✗ — Provisioning physical hardware and networking cables in the data center used by the function.

Rationale: Physical hardware and cabling are the cloud provider's responsibility for serverless services; customers do not provision on-premises equipment for Lambda.

D ✗ — Scaling the Lambda service capacity by adding or removing virtual machines manually.

Rationale: Lambda automatically scales based on demand; customers do not manually add or remove VMs to change capacity.

Explanation

Serverless compute shifts operational duties: the cloud provider operates the platform, but the customer continues to own the code and its runtime settings. In this scenario, the important point is that creating, maintaining, and protecting the function's application logic and its access controls remains the customer's job. The provider handles patching, virtualization, physical hardware, and automatic scaling for the execution environment.

Why the correct choice fits. The correct response identifies writing and securing the function code, setting its configuration and IAM permissions, and managing data access as the customer's responsibilities. These tasks include ensuring least-privilege permissions for the function, implementing any needed encryption for data the function reads or writes, and keeping dependencies or libraries the function uses up to date.

Why the other choices are wrong. Stating that the customer must manage operating system patches or physical hardware is incorrect because those are platform responsibilities in a serverless model. Claiming that customers must manually scale by adding virtual machines is also incorrect because the service provides automatic scaling for function invocations.

Key Concepts

- **Shared responsibility in serverless:** The provider manages the execution environment and infrastructure; the customer manages their code and related configurations.
- **Function configuration and permissions:** The customer configures runtime settings and uses IAM to grant only necessary permissions to resources the function accesses.
- **Data access and protection:** The customer must control who and what can access function data and apply encryption or access policies as needed.

Common Pitfalls

- Assuming the provider secures application code or IAM policies leads to under-protected functions.
- Confusing automatic runtime management with responsibility for the function's business logic and data handling.

Glossary

- **Lambda (AWS Lambda):** A serverless compute service that runs customer-provided code in response to events without requiring management of servers.
- **IAM (Identity and Access Management):** AWS service used to define who can access which resources and what actions they can perform.

13.10 — Six AWS CAF perspectives (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 13.10

A company is starting its first migration wave across several AWS accounts. Engineers are already launching resources, but tags are inconsistent and Finance cannot map spend to products. Security wants to prevent unapproved services and provide centralized, least-privilege access for employees. The migration lead wants a quick, foundational move that aligns with AWS Cloud Adoption Framework guidance and sets guardrails before scaling the migration. What should the team do next?

Options & Rationales

A ✗ — Prioritize Platform work: design the VPC with IPv6 and deploy a pilot using Amazon EC2 on Graviton and Amazon Aurora Serverless v2 to validate performance.

Rationale: Improves the landing zone and workload performance but does not solve cost allocation or centralized least-privilege access and guardrails across accounts.

B ✗ — Enable Amazon CloudWatch metrics and alarms, configure Auto Scaling, and set up AWS Backup to improve availability and efficiency for initial workloads.

Rationale: These Operations tasks aid reliability and cost efficiency at the workload level but do not establish account-wide cost governance or access controls.

C ✓ — Establish a governance baseline with AWS Organizations and IAM Identity Center: standardize tagging and cost allocation, apply guardrails using Service Control Policies, and centralize least-privilege access with permission sets.

Rationale: Directly addresses cost visibility and control (tagging and cost allocation), enforces guardrails (SCPs), and centralizes workforce access with least privilege (Identity Center).

D ✗ — Turn on AWS CloudTrail and Amazon GuardDuty across all accounts to improve logging and threat detection for the migration wave.

Rationale: Enhances Security observability and detection but does not provide cost allocation, tagging standards, or centralized least-privilege access and service guardrails.

Explanation

The AWS Cloud Adoption Framework (AWS CAF) groups guidance into six perspectives to align people, process, and technology. The business-focused perspectives are Business, People, and Governance. The technology-focused perspectives are Platform, Security, and Operations. Early in migration, establishing clear ownership and guardrails in Governance and People prevents

cost surprises, shadow IT, and access sprawl—common risks when teams begin building in multiple accounts.

In this scenario, Finance needs cost visibility, and Security wants to prevent use of unapproved services and centralize least-privilege access. Those needs align primarily to the Governance and People perspectives. The most effective next step is to set a governance baseline that standardizes tagging and cost allocation and applies account guardrails, coupled with centralized workforce access.

Creating an AWS Organizations foundation with Service Control Policies (SCPs) lets you set preventive controls at the account boundary (for example, blocking disallowed services or regions, or requiring tags at creation using condition keys). Standardizing tags supports cost allocation and accountability across products and teams. Using IAM Identity Center to centralize workforce access provides single sign-on and least-privilege authorization via permission sets, reducing the risk of shadow IT and over-privileged identities. This sequencing reflects AWS CAF: address People and Governance early so Platform, Security, and Operations work proceeds within safe boundaries.

The option focused on designing a VPC and deploying a pilot with compute and database services improves the Platform perspective but leaves cost allocation and centralized access unresolved. Enabling CloudWatch, Auto Scaling, and AWS Backup strengthens the Operations perspective (observability, efficiency, resilience) but does not create governance controls or tagging standards. Turning on CloudTrail and GuardDuty improves the Security perspective (logging and threat detection) yet still lacks cost governance and workforce access centralization. Only the approach that combines Organizations with SCP-based guardrails, tagging and cost allocation standards, and IAM Identity Center addresses all stated requirements as a foundational, next-step action.

Key Concepts

- **AWS CAF Governance perspective:** Establishes tagging, cost allocation, change control, and guardrails to manage risk and finances during migration.
- **AWS Organizations with SCPs:** Central policy engine to restrict actions and services across accounts; can enforce preventive controls at scale.
- **IAM Identity Center:** Centralizes workforce authentication and least-privilege authorization using permission sets and groups.
- **Tagging for cost allocation:** Consistent tags enable accurate chargeback/showback and financial accountability.

Common Pitfalls

- Treating migration as only a Platform task; this often causes skill gaps, shadow IT, and cost surprises when governance and access controls are missing.

13.11 — Aurora Global Database for low latency (D3.T3.4)

Domain 3 • Task 3.4

↑ Back to Question 13.11

Which statement is WRONG about an Amazon Aurora Global Database deployment intended to give low-latency local reads across Regions?

Options & Rationales

A ✗ — The primary cluster is the only writable cluster and handles transactional writes.

Rationale: This is true: the primary cluster accepts writes while other Regions serve read traffic.

B ✗ — Secondary clusters in other Regions are read-only and serve local read requests.

Rationale: This is true: secondary clusters are used for local read scalability and low-latency access.

C ✗ — Storage is copied asynchronously so replicas can be promoted quickly if the primary fails.

Rationale: This is true: asynchronous replication of storage enables rapid recovery by promoting a secondary.

D ✓ — All clusters can accept concurrent writes and automatically merge transactions across Regions.

Rationale: This is wrong because Aurora Global Database uses a single writable cluster; concurrent multi-Region writes with automatic cross-Region merge are not supported.

Explanation

Amazon Aurora Global Database is designed for cross-Region read scaling and disaster recovery. A single writable cluster in the primary Region handles transactional updates. Other Regions host read-only clusters that serve local read traffic and reduce latency for remote users. The underlying replication copies storage asynchronously from the primary to secondary Regions so a secondary can be promoted to writable in failover events.

Key Concepts

- **Primary writable cluster:** The central, write-capable cluster that processes transactions; it is the authoritative source of data.
- **Read-only secondary clusters:** Replica clusters in other Regions that provide low-latency reads and do not accept application writes.
- **Asynchronous physical replication:** Storage changes are propagated to secondaries without blocking primary writes, enabling quick promotion but not synchronous multi-master writes.

Why the wrong statement is incorrect. The claim that every cluster accepts writes and automatically merges transactions across Regions describes a multi-master, conflict-resolving system. Aurora Global Database does not provide that behavior; it enforces a single writer to maintain transactional consistency. Attempting to write concurrently to multiple Regions would

violate the design and lead to data conflicts that the service is not built to resolve automatically.

Common Pitfalls

- Assuming secondary clusters allow writes leads to design errors and potential data inconsistency expectations.
- Confusing fast cross-Region reads with cross-Region multi-master capabilities; low-latency reads are supported, but multi-Region writes are not.

Glossary

- **Promote:** Convert a read-only secondary into a writable primary during failover.
- **Asynchronous replication:** Replication that does not wait for replicas to acknowledge every write before returning success to the client.

13.12 — DMS for heterogeneous DB migrations (D3.T3.4)

Domain 3 • Task 3.4 ↑ Back to Question 13.12

A company must move a production Oracle database to Amazon RDS for PostgreSQL with minimal downtime and preserve ongoing transactions during migration. Which AWS approach best meets this requirement?

Options & Rationales

A ✗ — Take a full export from Oracle, transform files, and import them into RDS during a maintenance window.

Rationale: A one-time export/import causes prolonged downtime and misses transactions that occur after the export, so it does not meet the minimal-downtime requirement.

B ✗ — Use AWS Schema Conversion Tool alone to perform the migration without any data replication service.

Rationale: The Schema Conversion Tool helps translate schema and queries but does not handle continuous data replication needed to keep data synchronized during cutover.

C ✗ — Build a custom ETL pipeline with AWS Lambda functions to periodically copy changed rows to the target database.

Rationale: A custom periodic ETL risks data loss or inconsistent state during high-change periods and typically requires more effort than a managed replication service.

D ✓ — Use AWS Database Migration Service to replicate data continuously with validation, cut over when synchronized.

Rationale: AWS Database Migration Service supports continuous change capture between different database engines, keeps source and target synchronized, and includes validation to minimize cutover downtime.

Explanation

When migrating between different database engines and aiming for minimal downtime, a managed replication service that captures ongoing changes is the appropriate solution. Such a service continuously copies new and changed records from the source to the target while the source remains operational, allowing a short cutover once the target is caught up and validated. Simply exporting and importing data requires the source to be offline or risks missing transactions that occur after the export. A schema conversion utility is valuable for translating database structures and SQL dialects but does not provide continuous data synchronization by itself. A custom, periodic ETL approach can copy changes but introduces complexity, higher maintenance, and a greater chance of inconsistent data during busy periods compared to a managed change-data-capture service.

Key Concepts

- **Continuous change capture:** Captures and applies ongoing data modifications so source and target remain synchronized during migration.
- **Heterogeneous migration:** Moving between different database engines requires both schema conversion and data replication to preserve functionality and data.
- **Data validation and minimal cutover:** Verifying consistency between source and target reduces risk and shortens the switch-over window.

Common Pitfalls

- Assuming schema conversion alone completes a migration; schema conversion does not replicate live data.
- Relying on periodic bulk transfers for active production databases, which can miss transactions and increase downtime.

Glossary

- **Change-data-capture:** A technique that records and transmits only data changes rather than full dataset copies.
- **Cutover:** The final switch from the source system to the target system during migration.

13.13 — Storage Gateway for hybrid storage (D3.T3.6)

Domain 3 • Task 3.6

↑ Back to Question 13.13

A company wants to keep using its on-premises backup software that writes to tape over iSCSI, but reduce media costs and retain backups for years in the cloud using Amazon S3 Glacier Deep Archive. Which AWS solution best meets this requirement with minimal changes?

Options & Rationales

A ✗ — AWS Storage Gateway Volume Gateway — iSCSI block volumes with point-in-time EBS snapshots; not a VTL for tape-based backups.

Rationale: Volume Gateway provides block storage and snapshots for DR but does not emulate tapes, so legacy tape workflows are not supported.

B ✗ — AWS Storage Gateway File Gateway — NFS/SMB shares backed by S3 with lifecycle transitions; does not emulate tapes or use iSCSI.

Rationale: File Gateway is for file shares (NFS/SMB) mapped to S3. It cannot present a VTL and is not designed for tape-centric backup software.

C ✓ — AWS Storage Gateway Tape Gateway — Virtual tape library (VTL) over iSCSI; tapes stored in S3 and archived to S3 Glacier Deep Archive.

Rationale: Tape Gateway emulates a VTL compatible with existing iSCSI-based backup software. Virtual tapes land in S3 and can be archived to S3 Glacier Deep Archive.

D ✗ — Amazon S3 Lifecycle policies — Transition objects between classes; no VTL or iSCSI support for legacy backup software.

Rationale: Lifecycle policies optimize S3 storage classes but do not provide a tape interface or integrate with iSCSI-based backup applications.

Explanation

AWS Storage Gateway bridges on-premises environments with Amazon S3 using familiar protocols and a local cache for low-latency access. It offers three gateway types, each targeting a different workload pattern. For existing backup software that expects tapes over iSCSI, Tape Gateway is the fit-for-purpose option because it emulates a virtual tape library (VTL). Virtual tapes are durably stored in Amazon S3 and can be archived to Amazon S3 Glacier Flexible Retrieval or S3 Glacier Deep Archive for low-cost, long-term retention—meeting the requirement without changing the backup software. File Gateway maps NFS/SMB shares to S3 for file workloads and supports S3 features like lifecycle transitions, but it does not emulate tapes. Volume Gateway exposes iSCSI block volumes and supports point-in-time EBS snapshots for backup/DR of block workloads, not tape workflows. S3 lifecycle policies optimize storage tiering for objects but do not provide an iSCSI or VTL interface.

Key Concepts

- **AWS Storage Gateway:** Hybrid storage service with local caching and secure transfer (TLS) to S3, using familiar protocols (NFS/SMB, iSCSI) to avoid app changes.
- **Tape Gateway (VTL):** Emulates a virtual tape library over iSCSI; virtual tapes are stored in S3 and can be archived to S3 Glacier Deep Archive for long retention.
- **File vs. Volume Gateway:** File Gateway maps NFS/SMB shares to S3; Volume Gateway provides iSCSI block volumes with EBS snapshots—neither presents a virtual tape interface.
- **S3 Lifecycle Transitions:** Move objects between S3 storage classes (e.g., Intelligent-Tiering) but do not integrate with tape-based or iSCSI workflows.

Common Pitfalls

- Confusing Volume Gateway's iSCSI block volumes with a VTL; block volumes and snapshots do not replace tape library emulation.
- Assuming S3 Lifecycle or File Gateway can substitute for a tape interface; they cannot satisfy legacy iSCSI-based backup software requirements.

Glossary

- **Virtual Tape Library (VTL):** A storage virtualization that presents tape devices to backup software while storing data on disk/object storage.
- **iSCSI:** IP-based protocol for transporting SCSI commands used by block and tape devices.

13.14 — VPC purpose: resource isolation (D3.T3.5)

Domain 3 • Task 3.5 ↑ Back to Question 13.14

A company runs EC2 instances in private subnets and must access AWS services without sending traffic over the public internet. The instances must remain non–internet-routable. Which TWO actions meet this requirement?

Options & Rationales

A ✗ — Deploy a NAT Gateway in a public subnet so private subnets can reach AWS services without inbound exposure.

Rationale: NAT Gateway provides outbound internet access for private subnets; traffic still traverses the public internet and does not stay private to the VPC.

B ✓ — Create a Gateway VPC endpoint for Amazon S3 to keep S3 traffic within the VPC without NAT/IGW.

Rationale: Gateway VPC endpoints for S3 keep traffic on the AWS network inside your VPC and do not require NAT or an Internet Gateway.

C ✗ — Enable VPC Flow Logs to monitor and block internet egress from private subnets.

Rationale: VPC Flow Logs are for visibility and analysis only; they do not filter or block traffic.

D ✓ — Use AWS PrivateLink interface endpoints for supported AWS services to maintain private, VPC-contained connectivity.

Rationale: AWS PrivateLink provides interface endpoints that keep connections to many AWS services and partner SaaS private within your VPC.

E ✗ — Attach an Internet Gateway and restrict inbound access with security groups to allow outbound-only access to AWS services.

Rationale: An Internet Gateway enables internet routing; security groups limit exposure but traffic still uses the public internet.

F ✗ — Establish VPC peering with a VPC that has internet access to reach AWS services privately.

Rationale: VPC peering connects VPCs, not AWS service endpoints; it does not provide private access to AWS services.

Explanation

In Amazon VPC, connectivity is explicit, not implicit. By default, a VPC is isolated from the internet and other VPCs. To keep traffic to AWS services off the public internet from private subnets, you should use VPC endpoints. There are two relevant endpoint types:

- Gateway VPC endpoints provide private connectivity for Amazon S3 and Amazon DynamoDB. Traffic to these services remains within the AWS network and VPC path and does not require an Internet Gateway (IGW) or NAT.
- AWS PrivateLink interface endpoints provide private, VPC-contained connectivity to many other AWS services and partner SaaS. These endpoints keep service traffic inside your VPC, avoiding traversal over the public internet.

NAT Gateways allow instances in private subnets to initiate outbound connections, but that egress uses the public internet path; therefore, NAT does not meet the requirement to keep traffic off the public internet. An Internet Gateway explicitly enables internet routing; while security groups can restrict inbound access, they do not change the fact that the path is over the public internet. VPC peering is for private VPC-to-VPC connectivity and does not provide private access to AWS service endpoints. VPC Flow Logs give visibility into traffic flows but are not a control plane feature; they cannot block or filter traffic.

Key Concepts

- **Gateway VPC endpoint (S3/DynamoDB):** Keeps service traffic private within the VPC/AWS network without IGW or NAT.
- **AWS PrivateLink (interface endpoints):** Offers private connectivity from a VPC to many AWS services and partner SaaS, remaining inside the VPC.
- **IGW and NAT Gateway:** Enable internet egress; they do not keep traffic off the public internet.
- **Security groups and NACLs:** Control allow/deny at instance/subnet layers but do not determine whether traffic uses the internet.

Common Pitfalls

- Assuming a NAT Gateway keeps traffic private because instances are not directly reachable; NAT still uses the public internet for egress.
- Believing VPC Flow Logs can block traffic; they only provide visibility and governance, not enforcement.

Glossary

- **Internet Gateway:** A VPC attachment that enables internet routing.
- **NAT Gateway:** Managed egress for private subnets to the internet.

- **Gateway VPC endpoint:** Private access to S3/DynamoDB within a VPC.
- **AWS PrivateLink:** Interface endpoints for private access to many AWS services.

13.15 — Trade-offs among Well-Architected pillars (D1.T1.2)

Domain 1 • Task 1.2 ↑ Back to Question 13.15

Which definition best describes how Auto Scaling helps meet cloud workload needs while controlling expenses?

Options & Rationales

A ✓ — Automatically adjusts compute capacity up or down based on demand to maintain performance and reduce idle cost.

Rationale: Auto Scaling changes the number of instances in response to load, keeping performance during spikes and lowering costs when demand falls.

B ✗ — Provides a fixed pool of reserved instances to guarantee low latency at a predictable price.

Rationale: Reserved capacity is a pricing commitment for steady use; it does not dynamically change capacity with load.

C ✗ — Schedules long-term archival of data to lower storage costs without affecting compute resources.

Rationale: Data lifecycle or archival reduces storage expense but does not adjust compute resources in response to traffic.

D ✗ — Automatically encrypts data in transit and at rest to reduce security incident response time.

Rationale: Encryption protects data confidentiality and supports security goals, but it does not scale compute capacity for demand changes.

Explanation

Auto Scaling is a capability that changes the number of compute resources automatically in response to workload signals (for example, CPU usage or request rate). Its primary purpose is to keep application performance within targets during load increases by adding capacity, and to decrease the number of running instances when demand drops so you avoid paying for unused resources.

Key Concepts

- **Auto Scaling behavior:** Automatically increases or decreases capacity based on defined metrics or schedules so applications can handle variable demand.
- **Performance vs. cost trade-off:** Adding instances improves performance and availability during peaks; removing instances reduces ongoing cost during low demand.
- **Use case fit:** Best for workloads with variable or unpredictable traffic where right-sizing in real time improves cost efficiency and experience.

Why the correct answer fits. The correct choice describes dynamic adjustment of compute capacity tied to demand, explicitly linking maintained performance and reduced idle cost. That matches Auto Scaling's role as an automated mechanism to balance resource supply with application demand.

Why the other choices are incorrect. The option about reserved instances describes a cost model that offers discounts for committed usage but does not change capacity as load changes. The archival/storage scheduling option addresses storage cost management, not compute scaling. The encryption option relates to data protection and incident response, which are security responsibilities and unrelated to automatically changing compute capacity.

Common Pitfalls

- Confusing Auto Scaling with pricing commitments (Reserved Instances or Savings Plans) leads to thinking it provides predictable pricing rather than dynamic capacity.
- Assuming scaling changes storage tiering or security controls; Auto Scaling specifically targets compute capacity.

Glossary

- **Auto Scaling:** A feature that automatically adjusts the number of running compute instances to match application demand.
- **Reserved Instance:** A billing construct that reduces cost for a committed, steady level of usage but does not dynamically scale capacity.

13.16 — Direct Connect location affects design (D3.T3.2)

Domain 3 • Task 3.2 ↑ Back to Question 13.16

A company has two on-premises offices near different metro fiber hubs and needs a private, low-latency link to an AWS VPC in a single Region. Which Direct Connect location selection best meets the requirement to minimize packet travel time between on-premises and AWS?

Options & Rationales

A ✗ — Choose the Direct Connect facility geographically closest to the larger office and establish a private virtual interface to the target Region.

Rationale: Placing the connection nearest only the larger office helps that site but neglects the other office; it may not minimize overall latency for both locations.

B ✓ — Select the Direct Connect location closest to the majority of user traffic and connect with a private virtual interface in the Region hosting the VPC.

Rationale: Locating the connection near the bulk of traffic reduces round-trip distance and gives the VPC a shorter private path, lowering latency compared with more distant facilities.

C ✗ — Use any available Direct Connect site and rely on public virtual interfaces to reach AWS endpoints, since public VIFs automatically reduce latency.

Rationale: Public virtual interfaces are for access to public AWS endpoints and do not guarantee the lowest private path latency to a VPC compared with a private virtual interface placed nearby.

D ✗ — Pick a Direct Connect location in a different Region to force traffic over a Direct Connect gateway for path diversity and thus lower latency.

Rationale: Cross-Region attachments add extra network hops; they improve path options but typically increase latency versus a local Direct Connect site into the same Region.

Explanation

Selecting a Direct Connect presence close to where most traffic originates reduces the physical distance packets travel to reach the VPC, which directly lowers network latency. A private virtual interface provides dedicated connectivity into a VPC within the same Region, offering a shorter, more predictable path than routing through public endpoints or across Regions.

Key Concepts

- **Direct Connect location proximity:** Locating the connection near the primary on-premises traffic source shortens the network path and decreases round-trip time.
- **Private virtual interface:** Used to connect an on-premises network directly to a VPC in a Region, providing private routing that typically yields lower latency than public endpoints.
- **Cross-Region attachment trade-off:** While attaching across Regions via a gateway can offer redundancy or broader reach, it usually adds hops and can increase latency compared with a same-Region attachment.

Why the correct choice is right. Choosing the Direct Connect site closest to the majority of user traffic and creating a private virtual interface into the Region hosting the VPC reduces packet travel distance and avoids extra routing through public internet paths or additional Region hops. This yields the lowest and most consistent latency for the primary workload.

Why the other choices are wrong. Selecting only the larger office overlooks latency for the other site and may not minimize overall travel time. Relying on public virtual interfaces targets public AWS endpoints and does not provide the private, shortest path into a VPC. Intentionally using a different Region increases routing distance and typically raises latency despite providing different path options.

Common Pitfalls

- Assuming any Direct Connect site gives equal latency; proximity matters.
- Confusing public virtual interfaces with private connectivity to a VPC.

13.17 — GuardDuty detects anomalies (D2.T2.2)

Domain 2 • Task 2.2

↑ Back to Question 13.17

A security team is evaluating how Amazon GuardDuty finds suspicious activity. Which statements about GuardDuty's detection inputs and methods are INCORRECT?

Options & Rationales

A ✓ — It depends only on customer-written rules and manual allow/deny lists to raise findings.

Rationale: This is incorrect. GuardDuty uses managed analytics, not solely customer-authored rules; it provides built-in detections.

B ✓ — It does not employ machine learning or statistical modeling to detect unusual patterns.

Rationale: This is incorrect. GuardDuty applies ML and statistical approaches to surface anomalies.

C ✗ — It leverages built-in threat intelligence about known bad IP addresses and domains.

Rationale: GuardDuty incorporates threat intel to flag connections to known malicious endpoints.

D ✗ — It analyzes API activity recorded by AWS CloudTrail as a signal source.

Rationale: GuardDuty ingests CloudTrail management events to help spot risky API patterns.

E ✗ — It profiles typical behavior per resource to highlight deviations that may indicate threats.

Rationale: GuardDuty establishes baselines of normal activity and alerts on deviations.

Explanation

Amazon GuardDuty is a managed threat detection service that continuously analyzes account and network-related telemetry to identify malicious or anomalous behavior. It combines multiple input signals and techniques: activity logs from the account, built-in intelligence on known bad actors, and analytics that learn what "normal" looks like so deviations can be flagged.

- GuardDuty's use of CloudTrail management events enables it to observe API usage patterns and detect risky or unusual calls.
- The claim that it relies only on customer-written rules is wrong because GuardDuty provides managed detections; customers do not need to author or maintain rule sets for core findings.
- Threat intelligence about malicious IPs and domains helps GuardDuty quickly identify contact with known bad infrastructure.
- The statement that it does not use machine learning or statistics is incorrect; GuardDuty applies these methods to surface anomalies.
- Establishing typical per-resource behavior (baselining) allows GuardDuty to detect deviations that may indicate compromise.

Key Concepts

- **CloudTrail ingestion:** API and management event logs provide account activity signals for detection.
- **Threat intelligence:** Built-in lists of risky IPs/domains enhance detection of known bad actors.
- **Machine learning/statistics:** Analytical models identify anomalies beyond simple signatures.
- **Behavioral baselining:** Learning normal patterns per resource makes deviation-based alerts possible.

Common Pitfalls

- Assuming GuardDuty is a rule-only or customer-authored IDS leads to underestimating its managed analytics and anomaly detection capabilities.

13.18 — Landing zone & Control Tower accelerate (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 13.18

A company is migrating dozens of applications to AWS and wants to accelerate onboarding without sacrificing governance. Requirements:

- Separate Prod, Non-Prod, and Sandbox accounts provisioned in minutes
- Single sign-on for engineers
- Preventive and detective guardrails
- Centralized CloudTrail and AWS Config logs in a dedicated log account
- Consistent VPC design and encryption by default
- Ability to tailor controls by environment

Which approach is the BEST fit with minimal operational effort?

Options & Rationales

A ✗ — Use AWS Organizations with consolidated billing; manually create accounts, apply tag policies, aggregate AWS Config, store CloudTrail in the management account, and manage access with IAM users.

Rationale: Meets some governance needs but is manual, lacks SSO, and centralizes logs in the management account rather than a dedicated Log Archive account. Not the lowest effort or best practice.

B ✗ — Keep one AWS account with multiple VPCs per environment, enforce KMS-by-default via IAM policies, send CloudTrail to CloudWatch Logs, and rely on AWS Security Hub for oversight.

Rationale: A single-account model increases blast radius, weakens cost isolation, and cannot tailor guardrails by environment. It also omits a dedicated Log Archive account for centralized logs.

C ✓ — AWS Control Tower landing zone with OUs and Account Factory; enable SCP/AWS Config guardrails, centralized CloudTrail to Log Archive, and IAM Identity Center SSO.

Rationale: Control Tower automates a secure multi-account foundation with OUs, Account Factory, preventive (SCP) and detective (AWS Config) guardrails, centralized logging to a Log Archive account, and integrated SSO.

D ✗ — Combine IAM Identity Center and AWS Service Catalog to standardize provisioning; use AWS Config rules to block noncompliance; share a central S3 log bucket with teams.

Rationale: Identity Center and Service Catalog do not create a governed multi-account baseline. AWS Config is detective, not preventive, and this lacks a dedicated Log Archive account.

Explanation

A landing zone is a secure, scalable multi-account foundation that standardizes identity, logging, networking, and baseline security from day one. AWS Control Tower automates building and governing this landing zone by using AWS Organizations for account structure, IAM Identity Center for access, and Account Factory to rapidly create accounts from blueprints. Control Tower applies preventive guardrails with Service Control Policies (SCPs) and detective guardrails with AWS Config rules, routes CloudTrail and AWS Config logs to a dedicated Log Archive account, and supports centralized security monitoring (often via a separate Security account that can use services like Amazon GuardDuty). Organizational units (OUs) let teams tailor guardrails per environment, while standard account baselines enforce practices such as encryption by default and consistent VPC designs.

Given the requirements—fast provisioning of Prod/Non-Prod/Sandbox accounts, single sign-on, both preventive and detective controls, centralized logging in a dedicated account, consistency, and environment-specific controls—the most appropriate solution is to deploy an AWS Control Tower landing zone with OUs and Account Factory, enable SCP and AWS Config guardrails, centralize CloudTrail/Config logs to a Log Archive account, and integrate IAM Identity Center. This minimizes operational effort and accelerates onboarding while preserving least privilege and reducing blast radius.

Alternatives fall short: Building only with AWS Organizations and manual setup lacks automation and SSO and commonly centralizes logs in the management account instead of a dedicated Log Archive account. A single-account strategy undermines blast radius reduction and cost isolation and cannot tailor controls by environment. Combining Identity Center with Service Catalog and relying exclusively on AWS Config still misses a governed multi-account foundation and mistakes Config (detective) for a preventive control mechanism.

Key Concepts

- **Landing Zone & AWS Control Tower:** Automates multi-account setup with standardized OUs, baselines, and Account Factory for rapid, consistent account provisioning.

- **Guardrails:** Preventive controls use SCPs to restrict actions; detective controls use AWS Config rules to detect and flag noncompliance.
- **Centralized Logging Pattern:** Route AWS CloudTrail and AWS Config logs to a dedicated Log Archive account for tamper-resistant, organization-wide visibility.
- **OUs & Identity:** OUs tailor guardrails by environment; IAM Identity Center provides single sign-on aligned with least privilege.

Common Pitfalls

- Assuming AWS Config can block actions: AWS Config is detective only; it cannot prevent actions like SCPs can.
- Centralizing logs in the management account: Best practice is a separate Log Archive account to reduce risk and improve isolation.

13.19 — OpenSearch for logs & search (D3.T3.7)

Domain 3 • Task 3.7 ↑ Back to Question 13.19

A company needs near-instant text search across millions of documents with fast lookup of specific words and phrases. Which AWS service is best suited because it uses an inverted index for rapid term lookup?

Options & Rationales

A ✗ — Amazon Athena — interactive query service for running SQL on data in S3.

Rationale: Athena runs SQL queries over objects in S3 but does not provide inverted-indexed full-text search optimized for rapid term lookup across document collections.

B ✓ — Amazon OpenSearch Service — managed search cluster that builds inverted indexes for fast term-based queries.

Rationale: OpenSearch Service is designed for full-text search and builds inverted indexes, enabling rapid lookup of documents that contain specific terms; it is the appropriate managed search solution.

C ✗ — Amazon RDS — managed relational database for structured transactional data.

Rationale: RDS is optimized for relational workloads and transactions; it is not tailored for tokenized, relevance-ranked full-text search using an inverted index.

D ✗ — Amazon Kinesis Data Streams — real-time data streaming for ingesting and processing events.

Rationale: Kinesis is for ingesting and processing streaming data in real time, not for indexing text with an inverted index for fast term-based document retrieval.

Explanation

An inverted index is a data structure that maps each unique term to the list of documents that contain that term, enabling rapid retrieval of documents by word or phrase. Services designed for full-text search build and maintain inverted indexes so searches for terms return results quickly, even across millions of documents.

Amazon OpenSearch Service is a managed search and analytics platform that creates and uses inverted indexes for tokenized, term-based queries and relevance-ranked results. This makes it the best fit when the primary need is fast text lookup across large document sets.

Other services are not ideal for this use case. Amazon Athena executes SQL queries over objects in Amazon S3 and is suited to ad-hoc query and analytics, not inverted-indexed full-text search. Amazon RDS provides managed relational databases for structured transactional data and does not natively implement inverted indexes for relevance-ranked text search. Amazon Kinesis Data Streams handles real-time ingestion and streaming processing; it is not a search index or a retrieval engine.

Key Concepts

- **Inverted index:** A mapping from terms to documents that contain them, enabling fast term-based lookup across large text corpora.
- **Full-text search service:** A platform that tokenizes text, builds indexes like inverted indexes, and ranks results by relevance for user queries.

Common Pitfalls

- Confusing data-processing or query engines with search engines; a streaming or SQL query service does not replace an inverted-index search system.
- Assuming a relational database provides efficient, relevance-ranked full-text search without additional search tooling or extensions.

Glossary

- **Tokenize:** Breaking text into words or terms used as keys in an inverted index.
- **Relevance-ranked:** Ordering search results by how well they match the query.

13.20 — Managed services remove heavy lifting (D1.T1.1)

Domain 1 • Task 1.1 ↑ Back to Question 13.20

Which statement best describes the benefit of a managed service that automatically adjusts capacity based on demand?

Options & Rationales

A ✗ — It guarantees zero downtime by replicating every request across multiple regions without cost impact.

Rationale: Replicating requests across regions does not guarantee zero downtime and adds cost; automatic scaling focuses on adjusting capacity, not free global replication.

B ✗ — It eliminates the need for application performance testing because resources are always sufficient.

Rationale: Autoscaling helps match capacity to load but does not replace performance testing; right-sizing and testing remain important to ensure correct scaling behavior.

C ✗ — It fixes software bugs by applying vendor updates to application code automatically.

Rationale: Managed scaling handles resource capacity, not code fixes; updating application logic remains the customer's responsibility.

D ✓ — It increases or decreases compute resources in response to traffic, reducing manual intervention and preventing overloads.

Rationale: Automatic capacity changes match resource supply to usage, lowering operational work and improving availability during traffic spikes or drops.

Explanation

Managed services that handle capacity adjustments use rules or policies to add or remove resources as demand changes. This reduces the need for administrators to manually provision extra instances during traffic increases or to remove idle capacity during low usage periods. The primary effect is improved availability during spikes and reduced operational effort and cost during quiet periods.

Why the correct choice fits. The correct statement describes automatic scaling: increasing or decreasing compute resources in response to traffic, with the benefits of less manual work and fewer overloads. That captures the core idea of automated capacity management provided by managed services.

Why the other statements are wrong. The option claiming guaranteed zero downtime via cross-region replication conflates geographic redundancy with autoscaling; replication can improve resilience but does not eliminate downtime nor is it cost-free. The option saying performance testing is unnecessary misstates responsibilities; autoscaling helps but does not replace testing and validation of application behavior under load. The option about vendor fixes for application code confuses operational tasks; managed capacity does not correct bugs in customer code.

Key Concepts

- **Autoscaling:** Automatically adjusts resource count or size based on predefined metrics or schedules to match workload needs.
- **Operational overhead reduction:** Automating capacity reduces manual provisioning and intervention during demand changes.

- **Resilience vs. scaling:** Replication and multi-Region architectures improve fault tolerance; scaling manages resource quantity to meet load.

Common Pitfalls

- Assuming autoscaling removes the need for capacity planning or performance testing can lead to misconfigured scaling policies.
- Confusing resource scaling with code maintenance or guaranteed zero downtime results in incorrect architecture decisions.

13.21 — Support plans: Basic → Enterprise (D4.T4.3)

Domain 4 • Task 4.3 ↑ Back to Question 13.21

A small startup only requires help with account setup and billing questions and does not want to pay for technical guidance or incident support. Which AWS Support plan best matches this need?

Options & Rationales

A ✗ — Developer Support

Rationale: Includes technical guidance for development and non-production issues with response SLAs, which the startup does not require.

B ✗ — Business Support

Rationale: Offers 24/7 production-level technical support and infrastructure help, exceeding the startup's stated requirement and cost preference.

C ✓ — Basic Support

Rationale: Provides only account and billing assistance without technical support or operational guidance, matching the startup's minimal needs and lowest cost.

D ✗ — Enterprise Support

Rationale: Provides comprehensive technical account management and fastest SLAs for critical operations, far more than the startup needs.

Explanation

AWS offers multiple support tiers to match different organizational needs for technical assistance and operational continuity. For an organization that only needs help with account administration and billing, the entry-level support option is the appropriate choice because it focuses on those transactional services and avoids charges for technical troubleshooting, architecture reviews, or 24/7 incident response.

Why the correct choice fits. The correct description focuses on the template's role as a blueprint for instance attributes that Auto Scaling uses when creating or replacing instances. The other choices describe separate Auto Scaling concerns: scaling actions are driven by policies and CloudWatch metrics, auditing is handled by CloudTrail, and scheduled capacity changes are handled by scheduled actions rather than the template itself.

Key Concepts

- **Support tier purpose:** Each plan bundles a different level of technical engagement and response-time commitments, from basic account help to full technical account management.
- **Cost versus coverage:** Higher support tiers include more proactive technical services and faster SLAs, which increase cost; choose the lowest tier that meets required services.

Common Pitfalls

- Assuming the cheapest plan includes technical assistance; the entry-level plan does not provide technical troubleshooting.
- Confusing development guidance with billing support; development-oriented plans add technical help and SLAs.

Glossary

- **SLA:** Service Level Agreement, a documented response-time commitment for support cases.
- **Technical account management:** Higher-touch, proactive guidance and architectural reviews included in premium support tiers.

13.22 — Economies of scale drive lower prices (D1.T1.1)

Domain 1 • Task 1.1

↑ Back to Question 13.22

Which term best matches this definition in cloud economics: Per-unit cost decreases as total output grows because fixed costs are spread across more usage and variable costs drop through scale efficiencies?

Options & Rationales

A ✗ — Pay-as-you-go pricing

Rationale: This is a consumption billing model—pay only for what you use—but it does not inherently reduce the per-unit price as usage increases.

B ✗ — Elasticity

Rationale: Elasticity is the ability to match capacity to demand to avoid overprovisioning; it does not describe a per-unit price reduction from scale.

C ✓ — Economies of scale

Rationale: This is the definition: unit costs fall as output rises. At AWS, aggregated demand, higher utilization, and engineering efficiencies lower marginal cost.

D ✗ — Savings Plans

Rationale: These provide lower prices in exchange for a usage commitment for steady workloads; they are discounts by commitment, not cost reductions from scale.

Explanation

Economies of scale describe a structural cost advantage: as total output grows, the cost per unit falls. In the cloud, a provider aggregates demand across millions of workloads, purchases hardware and energy in bulk, and runs highly utilized, automated, multi-tenant infrastructure. Fixed costs (sites, power, cooling, R&D) are amortized over vast usage, while variable costs drop through supplier negotiations and engineering advances such as custom silicon. The result is a lower marginal cost for compute, storage, and networking.

AWS passes these efficiencies to customers through mechanisms like pay-as-you-go pricing, periodic price reductions, and volume pricing (for example, lower per-GB rates at higher usage bands in Amazon S3). Elastic services let customers right-size capacity, eliminating on-premises overprovisioning, while commitment-based options (such as Savings Plans or Reserved Instances) offer additional price reductions for steady usage. However, those models are distinct from the underlying economics of scale.

The correct choice is the concept that explicitly means falling per-unit cost as output increases. Pay-as-you-go is about when you are charged (consumption-based), not why the unit price changes. Elasticity is about matching capacity to demand to avoid idle resources, not reducing the unit price through scale. Savings Plans are commitment-based discounts, not a scale-driven cost effect.

Key Concepts

- **Economies of scale:** Per-unit cost decreases as output grows by spreading fixed costs and reducing variable costs via scale efficiencies.
- **Volume pricing:** Lower unit rates at higher usage bands (e.g., S3 per-GB tiers) are one way AWS passes scale benefits to customers.
- **Pay-as-you-go:** Consumption-based billing; pay only for what you use, independent of scale-driven unit price changes.
- **Elasticity:** Dynamically match capacity to demand to avoid overprovisioning and idle cost.

Common Pitfalls

- Confusing elasticity or pay-as-you-go with economies of scale; the former reduce waste or align cost to usage, while economies of scale reduce the unit price itself.

Glossary

- **Fixed costs:** Costs that do not vary with short-term output (e.g., facilities, R&D).
- **Variable costs:** Costs that change with output (e.g., consumables, energy).
- **Marginal cost:** The cost of producing one additional unit of output.

13.23 — AWS WAF mitigates web attacks (D2.T2.4)

Domain 2 • Task 2.4 ↑ Back to Question 13.23

A public website is receiving bursts of automated requests from a few client IPs causing short-term load spikes. Which Amazon WAF feature should you use to limit requests from a single client IP over time without blocking other clients?

Options & Rationales

A ✓ — Configure a rate-based rule to throttle requests from that client IP.

Rationale: Rate-based rules track the number of requests from an IP and automatically limit traffic when a threshold is exceeded, reducing load from high-request sources while allowing other clients to continue.

B ✗ — Create a managed rule group that inspects request bodies for SQL injection.

Rationale: Managed rule groups focus on detecting patterns like SQL injection; they do not specifically throttle or limit request rates from a single IP.

C ✗ — Use an IP set to permanently block the client IP address.

Rationale: An IP set can block IPs but is typically used for allow/deny lists; permanently blocking may be unnecessary for short bursts and lacks automated rate tracking.

D ✗ — Apply a regex pattern set to reject requests with suspicious query strings.

Rationale: Regex pattern sets match text patterns in requests and are useful for content validation, not for limiting the request rate from a client IP.

Explanation

When an individual client generates a high number of requests in a short period, the goal is to reduce that client's traffic without affecting legitimate users. A rate-based rule provides automated, time-windowed throttling by counting requests per client IP and taking action when a configured threshold is exceeded. This approach is suitable for short-term bursts and can be tuned to avoid false positives.

Other options are less appropriate because they address different problems. A managed rule group is designed to detect known attack patterns (for example, injection or cross-site scripting) rather than to limit request frequency. An IP set can block or allow specific addresses, which is useful for known malicious actors but is a blunt instrument for temporary spikes and requires manual updates. A regex pattern set helps match text patterns in request components and is used for content-based filtering; it does not provide request-rate controls.

Key Concepts

- **Rate-based rule:** Counts requests per client IP over a sliding time window and triggers configured actions when a threshold is reached, enabling automated throttling.

- **IP set**: A reusable list of IP addresses used for allow or deny decisions; suitable for static blocking or whitelisting.
- **Managed rule group**: A vendor-maintained collection of detection rules focused on common attack signatures, not on per-IP rate limiting.

Common Pitfalls

- Confusing content inspection with rate control: matching malicious payloads does not limit how many requests a client can send.
- Using permanent IP blocks for transient spikes can inadvertently block legitimate users and requires manual maintenance.

Glossary

- **Client IP**: The IP address from which an HTTP(S) request originates.
- **Throttling**: Temporarily restricting request handling to reduce load or abuse.

13.24 — Least privilege with IAM policies (D2.T2.3)

Domain 2 • Task 2.3 ↑ Back to Question 13.24

A developer reports they cannot grant an IAM role permissions to manage S3 objects, even though an administrator attached a broad inline policy to the role. The role has an IAM permissions boundary attached and still cannot perform PutObject. Which single action will most directly resolve this troubleshooting issue?

Options & Rationales

A ✗ — Attach the S3 managed policy (AmazonS3FullAccess) to the IAM user who created the role.

Rationale: Granting permissions to the user who created the role does not change the role's effective permissions. Roles use their own policies and boundaries, so this won't allow the role to perform PutObject.

B ✗ — Create a new role with the same inline policy and have the developer assume that role without changing any other settings.

Rationale: Making a new role with the same inline policy will still be subject to any existing permission boundary or organization limits. If a boundary blocks the action, the new role will also be blocked unless the boundary is changed.

C ✗ — Enable cross-account access to the bucket by modifying the bucket policy to allow the role's AWS account.

Rationale: A bucket policy controls resource permissions, but here the role is blocked from calling PutObject by its own permission limits. Adjusting the role's permission boundary is the direct fix; bucket policy changes are unnecessary if the role is in the same account and the boundary is the blocker.

D ✓ — Update the role's permission boundary to include the necessary S3 actions on the target bucket.

Rationale: A permission boundary limits the maximum permissions an identity

can have. Even if an inline policy grants PutObject, the boundary can block it; expanding the boundary to allow the S3 actions enables the role to act.

Explanation

A permission boundary is a managed constraint applied to an identity that sets the maximum permissions that identity can exercise. When an identity (user or role) has policies granting actions, the effective permissions are the intersection of those policies and any permission boundary. If the boundary does not include a required action, granting that action in an inline or managed policy will not enable the identity to perform it.

Why updating the permission boundary is correct. The role already has an inline policy that attempts to grant S3 PutObject, but the role's permission boundary restricts its maximum capabilities. Expanding the boundary to permit the S3 actions on the specific bucket raises the upper limit so that the inline policy can take effect. This directly resolves the mismatch between granted permissions and enforced limits.

Why the other options are incorrect. Granting a managed policy to the user who created the role changes that user's permissions, not the role's; roles use their own policies and boundaries. Creating a new role with the same policy will still be subject to the same type of boundary or organizational restriction, so it may not fix the problem. Modifying the bucket policy affects resource-based access controls; if the role is blocked by its own permission boundary, changing the bucket policy does not remove that boundary constraint.

Key Concepts

- **Permission boundary:** A constraint that defines the maximum permissions an identity can exercise; effective permissions are limited by both granted policies and the boundary.
- **IAM role policy vs. boundary:** A role's policies grant permissions, but those permissions are capped by any attached permission boundary.

Common Pitfalls

- Assuming that attaching a wide policy to an identity always grants full access; a permission boundary can still prevent the identity from using those permissions.
- Confusing user permissions with role permissions; changing the creator's permissions does not alter a role's effective rights.

Glossary

- **Permission boundary:** An IAM feature that sets an upper limit on what an identity can do, regardless of its policies.
- **Inline policy:** A policy embedded directly on a single identity that grants specific actions on resources.

13.25 — Sustainability design: minimize idle resources (D1.T1.2)

Domain 1 • Task 1.2

↑ Back to Question 13.25

A startup wants to reduce wasted, always-on compute capacity that sits idle between events. Which TWO AWS approaches best follow a serverless pattern to avoid provisioning servers?

Options & Rationales

A ✓ — Use AWS Lambda functions for event-driven processing that run only when invoked.

Rationale: Lambda is a serverless compute service that executes code in response to events and charges for actual execution time, avoiding always-running instances.

B ✗ — Store infrequently accessed objects in Amazon S3 Intelligent-Tiering to lower storage cost.

Rationale: S3 Intelligent-Tiering optimizes storage cost but addresses storage efficiency, not eliminating provisioned compute servers.

C ✗ — Provision EC2 instances with Reserved Instances to get lower hourly costs for steady workloads.

Rationale: Reserved Instances reduce cost for steady, long-running servers but still require managing provisioned instances that can remain idle.

D ✗ — Implement Auto Scaling groups with EC2 instances that add or remove servers based on demand.

Rationale: Auto Scaling reduces idle capacity compared to fixed fleets, but it still relies on customer-managed EC2 instances rather than a managed serverless model.

E ✓ — Choose Amazon DynamoDB with On-Demand capacity so the database scales and billing matches request traffic.

Rationale: DynamoDB On-Demand is a serverless database mode that automatically handles capacity and charges per request, removing the need to provision capacity ahead of demand.

Explanation

Serverless architectures shift responsibility for running and scaling infrastructure to the cloud provider so compute and database capacity do not remain provisioned when idle. AWS Lambda is a function-as-a-service: code runs only when triggered and billing reflects execution time and resources used, which directly prevents paying for idle servers. Amazon DynamoDB On-Demand similarly removes the need to provision read/write capacity ahead of time by automatically scaling and billing per request, avoiding preallocated database capacity.

Other options are plausible but do not match the serverless pattern. Purchasing Reserved Instances lowers hourly costs for predictable, always-on EC2 workloads but still requires maintaining provisioned servers. Auto Scaling reduces idle capacity by adjusting the number of EC2 instances, yet it continues to rely on customer-managed instances rather than fully managed, on-demand execution. S3 Intelligent-Tiering helps reduce storage costs by moving objects between tiers but does not eliminate provisioned compute resources.

Key Concepts

- **Serverless compute:** Services like AWS Lambda run code on demand without requiring users to provision or manage servers; billing aligns with actual execution.
- **Serverless data services:** On-demand database modes (for example, DynamoDB On-Demand) automatically scale capacity and charge per request, avoiding reserved capacity.
- **Provisioned vs. managed:** Provisioned resources (EC2, reserved instances) require capacity planning and can remain idle; managed serverless services handle scaling and reduce idle resources.

Common Pitfalls

- Confusing cost savings with serverless: Lowering hourly costs through reservations does not remove the operational burden or idle capacity of provisioned servers.
- Mixing resource types: Auto Scaling reduces waste but is not the same as serverless; it still involves managing instances.

13.26 — Health Dashboard security notifications (D2.T2.4)

Domain 2 • Task 2.4 ↑ Back to Question 13.26

Which AWS Health capability enables a delegated administrator to centrally aggregate and monitor account-specific security events across all accounts in an AWS Organizations environment?

Options & Rationales

A ✓ — AWS Health Organizational View with a delegated administrator in AWS Organizations

Rationale: Correct. Organizational View aggregates account-level AWS Health events from all member accounts so central teams can coordinate response.

B ✗ — AWS Health public Service health view (Region-wide service status)

Rationale: Incorrect. The public Service health view shows AWS service issues by Region and is not account-specific or cross-account.

C ✗ — AWS Health Your account view (formerly Personal Health Dashboard)

Rationale: Incorrect. Your account view lists alerts for a single account and does not aggregate across multiple accounts.

D ✗ — Amazon GuardDuty findings dashboard

Rationale: Incorrect. GuardDuty provides threat-detection findings, not aggregation of AWS Health advisories across accounts.

Explanation

AWS Health informs customers about AWS-initiated advisories and required actions that may affect their resources. It provides details such as scope (affected Regions/resources), impact timelines, and recommended steps so teams can prioritize remediation under the shared responsibility model.

There are two core views. The public Service health view shows current and historical issues with AWS services by Region. It helps understand broader service status but does not include account-specific notifications. The Your account view (formerly Personal Health Dashboard) is account-specific: it lists alerts such as security bulletins, required certificate rotations, protocol deprecations, and service configuration changes for that one account.

Enterprises operating multiple accounts need centralized visibility. AWS Health Organizational View, enabled with AWS Organizations and a delegated administrator, aggregates account-level Health events from all member accounts. This gives central operations or security teams a single place to monitor and coordinate response across the organization.

AWS Health can integrate with automation via the AWS Health API and Amazon EventBridge to route notifications to email, chat, ticketing, or trigger runbooks (for example, a Systems Manager Automation document). This integration distributes and automates responses but is not the mechanism for cross-account aggregation.

By contrast, Amazon GuardDuty is a separate threat-detection service that surfaces findings about suspicious or malicious activity in your environment; it does not replace AWS Health advisories.

Key Concepts

- **AWS Health Dashboard views:** Service health (public, Region-level) vs. Your account (account-specific alerts).
- **Organizational View:** Aggregates account-specific Health events across all member accounts via a delegated administrator.
- **EventBridge integration:** Routes and automates responses to Health events; not an aggregation feature.

Common Pitfalls

- Relying only on the public Service health page and missing account-specific advisories.
- Confusing Health advisories with GuardDuty threat-detection findings.

13.27 — Access audit logs: CloudTrail/Logs (D2.T2.2)

Domain 2 • Task 2.2

↑ Back to Question 13.27

Which AWS service is the primary mechanism for recording account API calls and management events so administrators can audit user and service activity?

Options & Rationales

A ✓ — AWS CloudTrail

Rationale: CloudTrail captures and records account API calls and management events across AWS services, providing event history for auditing and investigation.

B ✗ — Amazon CloudWatch Logs

Rationale: CloudWatch Logs stores and analyzes log data from resources and applications but is not the primary service for recording account API activity and management events.

C ✗ — AWS Config

Rationale: AWS Config records resource configuration changes and compliance over time, not the broad stream of account API calls and management events.

D ✗ — AWS Identity and Access Management (IAM)

Rationale: IAM manages identities, permissions, and authentication but does not itself record the history of API calls and management events for auditing.

Explanation

Recording account actions is essential for security governance and auditability. A service that captures API requests and management events provides a chronological record of who did what and when across an AWS account. This historical event data supports investigations, compliance reviews, and troubleshooting.

Why CloudTrail is correct. CloudTrail is designed to log and deliver account-level API activity and management events from AWS services. It generates event records that include the identity making the request, the time, the source IP, and the actions taken, which auditors and security teams use to reconstruct activity.

Why the others are incorrect:

- Amazon CloudWatch Logs is a centralized log store for application and system logs and can ingest custom events, but it is not the dedicated service that automatically records all account API and management events across services. CloudWatch Logs complements auditing by storing and analyzing log streams.
- AWS Config tracks resource configuration states and changes over time, helping with compliance of resource settings; it does not serve as the primary recorder of API calls and general management events.
- IAM provides identity, authentication, and authorization controls (users, roles, policies) but does not itself produce the comprehensive API call history required for account-level auditing.

Key Concepts

- **Audit event recording:** Capturing who performed actions, when, and from where to support investigations.
- **Account API and management events:** Requests that change or query AWS services and account settings; these form the audit trail used by security and compliance teams.

Common Pitfalls

- Confusing log storage with audit capture: storing logs (CloudWatch Logs) is different from the automatic capture of account API events (CloudTrail).
- Assuming configuration history (AWS Config) is the same as API activity; Config shows resource state changes, not the full API call stream.

13.28 — DynamoDB NoSQL use cases (D3.T3.4)

Domain 3 • Task 3.4 ↑ Back to Question 13.28

An e-commerce company is building a serverless shopping cart service on Amazon DynamoDB for a global flash sale. Requirements:

- Single-digit millisecond latency at scale
- Fast per-user cart retrieval
- Microsecond read latency for a hot read path (e.g., price/availability lookups)
- Automatic removal of carts idle for 48 hours
- Low-latency access and active-active resilience across two AWS Regions

Which statement is NOT an appropriate use of DynamoDB features for these requirements?

Options & Rationales

A ✗ — Amazon DynamoDB table with a composite primary key (partition key userId, sort key productId) to query a user's current cart items efficiently.

Rationale: Modeling by access pattern with a composite primary key groups all of a user's cart items under the same partition, enabling fast per-user queries.

B ✓ — Use DynamoDB Accelerator (DAX) to achieve microsecond write latency for cart updates during the flash sale.

Rationale: DAX provides microsecond read performance via an in-memory cache; it does not make writes to DynamoDB microsecond-latency operations.

C ✗ — Enable DynamoDB Time to Live (TTL) on an expiresAt attribute to automatically purge abandoned carts after 48 hours.

Rationale: TTL automatically removes items after the specified timestamp, which fits ephemeral data like abandoned carts.

D ✗ — Use Amazon DynamoDB Global Tables to provide active-active replication across two Regions for global low-latency access and resilience.

Rationale: Global Tables enable active-active, multi-Region replication to serve users with low latency and improve multi-Region resilience.

Explanation

DynamoDB is a fully managed, serverless NoSQL database designed for single-digit millisecond latency at scale. Effective data modeling focuses on access patterns and the primary key. For a shopping cart, grouping each customer's cart items by user is ideal. Using a composite primary key with userId as the partition key and productId as the sort key stores related items together and enables a single, efficient Query to return the current cart. Ephemeral data like abandoned carts should be automatically removed; TTL is purpose-built for this and deletes items after a specified timestamp without application code. For global users who need low-latency access and multi-Region resilience, Global Tables provide active-active replication so the application can read and write in multiple Regions.

DAX is a managed, in-memory cache specifically for DynamoDB that provides microsecond read performance. It helps read-heavy, latency-sensitive workloads by caching responses and offloading reads from the table. However, it does not convert writes to microsecond latency; writes still go to DynamoDB. For spiky traffic, DynamoDB capacity can be operated in on-demand or auto scaled modes, but DAX remains a read accelerator.

Therefore, the statement that claims DAX achieves microsecond write latency is incorrect. The other statements align precisely with DynamoDB best-fit features: composite keys for per-user cart queries, TTL for automated cleanup, and Global Tables for active-active multi-Region access.

If capacity estimation is needed, DynamoDB uses throughput units. Strongly consistent reads approximate to RCUs $\approx \lceil \text{itemKB}/4 \rceil \times \text{req/s}$, and writes to WCUs $\approx \lceil \text{itemKB}/1 \rceil \times \text{req/s}$. Eventually consistent reads use about half the RCUs of strongly consistent reads.

Variables used:

- **itemKB:** Item size [KB]
- **req/s:** Requests per second [1/s]
- **RCUs:** Read Capacity Units [units]
- **WCUs:** Write Capacity Units [units]

Key Concepts

- **Primary key design (partition + sort key):** Organizes related items together to serve known access patterns with low latency.
- **TTL for ephemeral data:** Automatically deletes items after a timestamp, reducing storage and operational overhead.
- **Global Tables:** Active-active, multi-Region replication for global low-latency access and resilience.
- **DAX:** Managed, in-memory cache that delivers microsecond read performance; it is not a write accelerator.

Common Pitfalls

- Assuming DAX speeds up writes; it accelerates reads only. Misusing scans or relational-style modeling instead of designing keys and indexes for access patterns can cause hot partitions and higher costs.

13.29 — Multi-AZ achieves high availability (D3.T3.2)

Domain 3 • Task 3.2 ↑ Back to Question 13.29

A company runs a stateless web application in two subnets across different Availability Zones behind an Application Load Balancer. An Amazon EC2 Auto Scaling group spans both AZs with a minimum of 2 instances. The database is a single-AZ Amazon RDS instance. During an AZ disruption that affected the database's AZ, users saw errors and the site was unavailable, even though instances in the other AZ were healthy. What change will most directly prevent this outage in the future?

Options & Rationales

A ✗ — Increase the Auto Scaling group minimum capacity to 4 instances across both AZs.

Rationale: More EC2 instances do not help if the database is unavailable. The outage stemmed from a single-AZ database, not insufficient web tier capacity.

B ✗ — Adjust Application Load Balancer health checks to route traffic away from the impaired AZ.

Rationale: Elastic Load Balancing already routes to healthy targets. Routing alone cannot fix an unavailable single-AZ database dependency.

C ✓ — Convert the database to an Amazon RDS Multi-AZ deployment with a synchronous standby in another AZ and automatic failover.

Rationale: This removes the database as a single point of failure. RDS Multi-AZ keeps a synchronous standby in a second AZ and performs automatic failover during an AZ outage.

D ✗ — Store user session state in Amazon DynamoDB to avoid in-memory sessions on instances.

Rationale: Externalizing sessions improves statelessness, but it does not address the database being single-AZ. The data tier would still be a single point of failure.

Explanation

Availability Zones are independent fault domains within a Region. High availability requires distributing redundant resources across at least two AZs so a localized failure does not take down the workload. In the scenario, the web tier is already designed for resilience: an Application Load Balancer fronts an Auto Scaling group that spans multiple AZs, allowing traffic to be routed only to healthy instances if one AZ is impaired. However, the database is a

single-AZ Amazon RDS instance, which creates a single point of failure. When the AZ hosting the database failed, the application could not serve requests, even though web servers in the other AZ were healthy.

Making the database highly available resolves the root cause. An Amazon RDS Multi-AZ deployment maintains a synchronous standby in another AZ and performs automatic failover during an AZ disruption. This aligns the data tier with the same Multi-AZ pattern used by the compute tier, removing the single-AZ dependency and minimizing user disruption during an AZ failure.

Increasing the number of EC2 instances does not help because additional compute capacity cannot compensate for an unavailable database. Tuning Application Load Balancer health checks only influences routing to targets; it cannot recover a failed downstream data tier. Externalizing session state to Amazon DynamoDB is a good practice for stateless apps, but it does not address database availability; the application would still fail if its single-AZ database is unreachable.

Key Concepts

- **Availability Zones and fault domains:** Independent facilities within a Region; spreading resources across AZs reduces the blast radius of localized failures.
- **Multi-AZ data tier:** Amazon RDS Multi-AZ keeps a synchronous standby in another AZ and triggers automatic failover to maintain availability during an AZ outage.
- **Tier alignment:** All critical tiers (compute and database) must avoid single-AZ dependencies to achieve end-to-end high availability.

Common Pitfalls

- Assuming a Multi-AZ web tier alone ensures overall availability while leaving the database single-AZ.
- Expecting load balancer health checks or more EC2 instances to compensate for an unavailable database dependency.

13.30 — Detective vs preventive vs corrective (D2.T2.2)

Domain 2 • Task 2.2 ↑ Back to Question 13.30

A company wants to prevent unintended public access to Amazon S3 buckets and also maintain a complete audit trail of who did what in their AWS account. Which TWO AWS controls should they enable?

Options & Rationales

A ✗ — AWS Config rule to detect publicly accessible S3 buckets

Rationale: Detective control that flags violations but does not prevent public access or provide full API audit logs by itself.

B ✓ — Amazon S3 Block Public Access to prevent public bucket and object access

Rationale: Preventive control that blocks public access settings, reducing exposure from public ACLs or bucket policies.

C ✗ — AWS WAF to filter malicious web requests to applications

Rationale: Preventive control for HTTP/S traffic, not for governing S3 bucket public access or producing API audit logs.

D ✗ — Amazon GuardDuty to continuously monitor for threats

Rationale: Detective control that surfaces threat findings; it is not a comprehensive API audit log and does not prevent public S3 access.

E ✗ — AWS Key Management Service (KMS) default encryption for S3

Rationale: Preventive data-at-rest encryption improves confidentiality but does not stop public access or generate API audit logs.

F ✓ — AWS CloudTrail to record and retain API activity for auditing

Rationale: Detective control that captures API calls as audit logs, enabling investigations after events occur.

Explanation

Security controls are commonly grouped by when they act: preventive controls stop unwanted actions before they occur; detective controls find issues after they happen so teams can investigate and respond; corrective controls restore systems or mitigate impact after detection. Mapping requirements to these categories helps select the right AWS services.

In this scenario there are two distinct goals. First, preventing unintended public access to Amazon S3 requires a preventive control that narrows exposure before it occurs. Amazon S3 Block Public Access is purpose-built for this: it enforces settings that block public access, reducing the attack surface created by public ACLs or bucket policies. Second, maintaining a complete audit trail of who did what is a detective requirement for API visibility. AWS CloudTrail provides account-level API audit logs, recording management activity so security and compliance teams can review changes and investigate incidents. Together, these two controls align with the Well-Architected Security pillar's emphasis on identity, protection, and traceability.

Other listed services do not meet both requirements. An AWS Config rule that detects public buckets is a valuable detective signal but neither prevents the exposure nor substitutes for CloudTrail's API audit logs. AWS WAF is a preventive web application firewall for filtering malicious HTTP/S requests, not a control over S3 bucket public access or an audit logging service. AWS KMS default encryption enhances data confidentiality at rest but does not govern whether a bucket is public or produce API audit logs. Amazon GuardDuty is a detective threat detection service that generates findings from analyzed telemetry; it is not a full audit log and does not enforce preventive guardrails on S3.

Key Concepts

- **Preventive controls:** Stop risky actions before they occur (e.g., S3 Block Public Access reduces exposure from public ACLs/policies).
- **Detective controls:** Discover events/changes after they happen (e.g., CloudTrail provides API audit logs for investigations).
- **Control-to-requirement mapping:** Use preventive for "don't allow X" and detective for "show me what happened."

Common Pitfalls

- Choosing only detective alerts (e.g., Config or GuardDuty) without a preventive control leaves exposure possible and slows response compared to blocking it upfront.

13.31 — KMS protects data at rest (D2.T2.2)

Domain 2 • Task 2.2 ↑ Back to Question 13.31

Which statement best defines envelope encryption when using AWS Key Management Service (AWS KMS) to protect data at rest?

Options & Rationales

A ✗ — All data is sent to AWS KMS, which encrypts and decrypts it inside hardware security modules; applications never handle encryption keys locally.

Rationale: KMS does not perform bulk data encryption. Services use locally obtained data keys for encryption and call KMS only for key operations (generate/wrap/unwrap).

B ✗ — The KMS key material is exported to applications for local encryption so workloads can run offline without calling KMS.

Rationale: KMS keys never leave AWS KMS's FIPS 140–validated HSMs and are not exportable. Applications receive data keys, not KMS key material.

C ✗ — A single KMS key encrypts all objects directly in the storage service, eliminating per-object data keys and extra metadata.

Rationale: Envelope encryption uses unique data keys (often per object) and KMS keys only wrap those data keys; KMS keys do not directly encrypt the bulk data.

D ✓ — AWS KMS issues a unique data key used locally to encrypt the data; only the ciphertext and the data key encrypted under a KMS key are stored.

Rationale: This is envelope encryption: KMS provides a plaintext data key for local encryption, the plaintext key is discarded, and only ciphertext plus the wrapped data key are persisted.

Explanation

Envelope encryption is a pattern used by AWS services that integrate with AWS Key Management Service (AWS KMS) to protect data at rest. An application or service requests a unique data key from KMS. The plaintext data key is used

locally (for example, with a symmetric cipher such as AES-256) to encrypt the data. The plaintext data key is then immediately discarded. Only two artifacts are stored with the data: the ciphertext (encrypted data) and the same data key encrypted (wrapped) under a KMS key. The KMS key that protects data keys never leaves KMS's FIPS 140–validated hardware security modules. When decryption is needed, an authorized principal calls KMS to unwrap the encrypted data key; the bulk data is still decrypted locally using the recovered plaintext data key. CloudTrail records every KMS API call, supporting audit and governance.

The description that states KMS issues a unique data key used locally, while storing only ciphertext and the wrapped data key, matches envelope encryption exactly. The statement claiming all data is sent to KMS for encryption is incorrect because KMS performs key operations, not bulk data encryption. The claim that KMS key material is exported is incorrect because KMS keys are non-exportable and remain within KMS HSMs. The idea that a single KMS key directly encrypts all objects is wrong because envelope encryption relies on distinct data keys (often per object) with the KMS key only wrapping those data keys.

Key Concepts

- **Envelope encryption:** Use a locally applied, unique data key to encrypt data; store ciphertext plus the data key encrypted under a KMS key.
- **KMS key vs. data key:** KMS keys stay inside KMS HSMs (non-exportable); data keys are returned to clients for local cryptographic operations.
- **Authorization and audit:** Decrypt operations require authorized KMS API calls, and CloudTrail logs KMS usage for traceability.

Common Pitfalls

A frequent mistake is assuming KMS encrypts the bulk data inside HSMs or that KMS keys can be exported. In envelope encryption, only data keys encrypt data locally, and plaintext data keys are not stored.

13.32 — IoT Core for device connectivity (D3.T3.8)

Domain 3 • Task 3.8 ↑ Back to Question 13.32

A startup is onboarding battery-powered sensors to AWS IoT Core and will use MQTT for telemetry and remote commands. To keep the design aligned with MQTT's lightweight publish/subscribe model, which TWO actions should the team AVOID?

Options & Rationales

A ✗ — Have sensors publish data to topics while back-end services subscribe to those topics.

Rationale: This follows MQTT's publish/subscribe pattern where producers publish and consumers subscribe.

B ✓ — Open a separate one-to-one connection from each sensor to every consumer device to deliver messages directly.

Rationale: MQTT uses a publish/subscribe model; producers do not create direct per-subscriber delivery paths. Forcing point-to-point connections violates the pub/sub approach.

C ✗ — Use small, periodic MQTT messages to reduce overhead for constrained devices.

Rationale: Keeping messages small aligns with MQTT's lightweight design and helps constrained devices.

D ✓ — Duplicate-publish the same telemetry multiple times to different destinations instead of using a single topic that multiple subscribers can consume.

Rationale: In publish/subscribe, one publish fan-outs to many subscribers. Re-sending duplicates wastes bandwidth and undermines MQTT's efficiency.

E ✗ — Subscribe devices to topics to receive asynchronous command messages from the cloud.

Rationale: Subscribing to topics for commands matches the MQTT pub/sub pattern.

F ✗ — Design topics by function (for example, telemetry and commands) so producers and consumers remain loosely coupled.

Rationale: Topic-based decoupling is consistent with MQTT's pub/sub model and aids scalability.

Explanation

MQTT is a lightweight, publish/subscribe messaging protocol well-suited for constrained IoT devices. In a pub/sub model, producers publish messages to topics, and any interested consumers subscribe to those topics to receive the messages. This decouples senders from receivers, supports fan-out (one message to many subscribers), and keeps device overhead low.

The action proposing separate one-to-one connections from each sensor to every consumer conflicts with the publish/subscribe model because producers should not manage per-subscriber delivery paths. The action suggesting duplicate-publishing the same telemetry to multiple destinations undermines MQTT's efficiency: pub/sub is designed so a single publish can be consumed by multiple subscribers without resending identical payloads.

By contrast, publishing telemetry to topics, subscribing devices to topics for commands, keeping messages small, and organizing topics by function all align with MQTT's lightweight, decoupled design.

Key Concepts

- **MQTT publish/subscribe:** Producers publish to topics; consumers subscribe. Decouples endpoints and enables fan-out.

- **Lightweight messages:** Small payloads and minimal overhead help constrained devices conserve bandwidth and power.
- **Topic design for decoupling:** Function-oriented topics keep producers and consumers independent and scalable.

Common Pitfalls

- Treating MQTT like point-to-point messaging leads to unnecessary connections and complexity.
- Re-sending identical telemetry to multiple destinations inflates bandwidth and cost instead of leveraging pub/sub fan-out.

13.33 — S3 pricing tiers & Intelligent-Tiering (D4.T4.1)

Domain 4 • Task 4.1 ↑ Back to Question 13.33

A company stores millions of objects with unpredictable access patterns and wants minimal operational effort while keeping immediate access when an object is requested. Which Amazon S3 storage class best meets this requirement?

Options & Rationales

A ✗ — Amazon S3 Standard (frequently accessed, low latency)

Rationale: S3 Standard is optimized for frequent access and low latency but doesn't automatically move objects to lower-cost tiers for unpredictable access, so it can be more expensive when many objects are rarely accessed.

B ✓ — Amazon S3 Intelligent-Tiering (automatic tiering to reduce cost without operational changes)

Rationale: Intelligent-Tiering automatically moves objects between access tiers based on observed usage, minimizing manual management while preserving immediate access when an object becomes active.

C ✗ — Amazon S3 Standard-Infrequent Access (lower storage cost, retrieval fee)

Rationale: Standard-IA offers lower storage pricing for infrequently accessed objects but requires choosing that class up front and incurs retrieval charges, so it doesn't provide automatic tiering for unpredictable patterns.

D ✗ — Amazon S3 Glacier Instant Retrieval (archival with very low-cost storage and millisecond retrieval)

Rationale: Glacier Instant Retrieval is a low-cost archival option with rapid retrieval but is intended for long-term archive workloads and does not automatically adapt tiers for unpredictable, general-purpose object access patterns.

Explanation

When object access is uneven and unpredictable, the goal is to minimize storage cost without adding operational overhead or delaying retrieval. A storage option that automatically adapts to changing access patterns reduces manual

classification and the risk of overpaying for rarely read objects while preserving fast reads for active ones.

Why the correct choice fits. Amazon S3 Intelligent-Tiering monitors access patterns and moves objects between access tiers on the customer's behalf. This removes the need for manual transitions or lifecycle rules and keeps frequently accessed objects available immediately, while moving seldom-accessed objects to lower-cost tiers to save money.

Why the other choices do not match. S3 Standard provides consistent low-latency access but does not reduce costs for objects that become cold, so it can be expensive for unpredictable workloads. S3 Standard-Infrequent Access lowers monthly storage costs but requires you to select that class before use and charges for retrievals; it does not dynamically react to changing access. S3 Glacier Instant Retrieval targets archival use with very low storage cost and fast retrieval for archives, but it is intended for long-term retention rather than dynamic automatic tiering across general access tiers.

Key Concepts

- **Automatic tiering:** A feature where storage moves objects between cost/performance tiers based on observed usage without manual intervention.
- **Access pattern variability:** When object reads are unpredictable, dynamic cost-optimization reduces the need for guesswork about which class to choose.
- **Retrieval behavior trade-off:** Some classes reduce storage cost but add retrieval fees or delays; intelligent tiering aims to balance cost and instant access.

Common Pitfalls

- Assuming a single static class (Standard or Standard-IA) will be cost-optimal for unpredictable workloads leads to either overpaying or frequent manual reclassification.
- Treating archival classes as automatic tiering solutions misunderstands their purpose; archival classes are for long-term retention rather than dynamic lifecycle management.

13.34 — AWS Artifact for compliance reports (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 13.34

A healthcare startup runs on Amazon EC2, Amazon RDS, and Amazon S3 and is preparing for an audit. They must: (1) provide auditor-ready evidence of AWS-owned controls, and (2) put account-level terms in place to handle protected health information (PHI). Which TWO actions should they take?

Options & Rationales

A ✓ — Use AWS Artifact Reports to download SOC 2 Type II and PCI DSS Attestation of Compliance as evidence of AWS-owned controls.

Rationale: AWS Artifact Reports provide independent, auditor-ready evidence of AWS controls across data centers, network, and managed services.

B ✗ — Enable AWS Artifact to continuously monitor EC2, RDS, and S3 for HIPAA configuration compliance.

Rationale: Incorrect. AWS Artifact does not configure, monitor, or validate customer workloads; it only supplies AWS-owned compliance evidence and agreements.

C ✓ — Review and accept the HIPAA Business Associate Addendum in AWS Artifact Agreements.

Rationale: AWS Artifact Agreements is where you review and accept account-level agreements like the HIPAA BAA for PHI workloads.

D ✗ — Rely solely on AWS Artifact documents to confirm the application is compliant.

Rationale: Incorrect. Artifact covers AWS-owned controls only; customers must provide their own evidence (e.g., IAM policies, CloudTrail logs).

E ✗ — Publish AWS Artifact compliance reports on the company website for transparency.

Rationale: Incorrect. Redistribution is restricted; share only with parties who need the documents, per report terms.

F ✗ — Use AWS Config to obtain ISO 27001 certificates for auditors.

Rationale: Incorrect. ISO and other third-party reports are downloaded from AWS Artifact Reports, not from AWS Config.

Explanation

Under the shared responsibility model, AWS secures the infrastructure that runs AWS Cloud services (security of the cloud), while customers configure and operate their workloads securely (security in the cloud). Auditors typically require evidence for both sides.

AWS Artifact is the on-demand portal for independent compliance evidence covering AWS-owned controls. Through AWS Artifact Reports, customers can download SOC 1/2/3, ISO 27001/27017/27018 certificates, PCI DSS Attestation of Compliance, and program summaries that document the design and operating effectiveness of controls across AWS data centers, network, and managed services. Separately, AWS Artifact Agreements is where account owners review and accept account-level agreements such as the HIPAA Business Associate Addendum (BAA) and the GDPR Data Processing Addendum (DPA).

AWS Artifact does not configure, monitor, or validate customer workloads. Customer-owned evidence must come from their environment (for example, CloudTrail logs, AWS Config tracking of resource configurations, and least-privilege IAM policies). Conceptually, the total control set for an audit is the sum of AWS and customer controls: $C_{total} = C_{AWS} + C_{customer}$.

Variables used:

- C_{total}: total number of controls relied upon
- C_{AWS}: AWS-owned controls evidenced via AWS Artifact
- $C_{customer}$: customer-owned controls (configuration, processes)

Applied to the scenario: To provide auditor-ready evidence of AWS-owned controls, the correct action is to use AWS Artifact Reports to download items like SOC 2 Type II and PCI DSS AoC. To put account-level terms in place for PHI, the correct action is to review and accept the HIPAA BAA in AWS Artifact Agreements.

Other actions are incorrect: attempting to "enable monitoring" in AWS Artifact misunderstands the service; using AWS Config to obtain ISO certificates confuses its purpose; publicly posting reports violates redistribution restrictions (share only with parties who need them); and relying solely on AWS Artifact documents ignores the customer's responsibility to present their own configuration evidence.

Key Concepts

- **Shared Responsibility Model:** AWS secures the cloud; customers secure their workloads and configurations in the cloud.
- **AWS Artifact Reports:** Auditor-ready SOC/ISO/PCI documents evidencing AWS-owned controls.
- **AWS Artifact Agreements:** Portal to review and accept account-level terms like HIPAA BAA and GDPR DPA.
- **Combined Evidence:** Compliance artifacts must include both AWS and customer controls ($C_{total} = C_{AWS} + C_{customer}$).

Common Pitfalls

- Assuming AWS Artifact alone equals application compliance, or that it monitors AWS resources. It only supplies AWS-side evidence and agreements. Also avoid redistributing restricted reports broadly.

Glossary

- **AoC:** Attestation of Compliance (PCI DSS).
- **BAA:** Business Associate Addendum (HIPAA).

13.35 — Auto Scaling adjusts capacity to demand (D3.T3.1)

Domain 3 • Task 3.1 ↑ Back to Question 13.35

Select TWO actions that are appropriate uses of scheduled scaling for an Auto Scaling group:

Options & Rationales

A ✗ — Use historical usage patterns to automatically predict future capacity needs without fixed times.

Rationale: Automatically forecasting future demand is the purpose of predictive scaling, not fixed-time scheduled adjustments.

B ✗ — Continuously adjust capacity to keep a specific CPU utilization percentage.

Rationale: This describes maintaining a metric target, which is handled by target-tracking scaling, not scheduled changes.

C ✗ — Scale only when a metric threshold is breached, stepping up or down in defined increments.

Rationale: Responding to threshold breaches with incremental changes matches step-scaling rules, not scheduled scaling.

D ✓ — Increase instance count before predictable daily traffic peaks (for example, just prior to business hours).

Rationale: Scheduling an increase ahead of a known busy period ensures capacity is available when traffic rises without waiting for metric triggers.

E ✓ — Lower the number of instances during known off-peak hours to reduce running costs.

Rationale: Setting a timed decrease aligns capacity to expected lower demand, saving cost by removing unneeded instances at scheduled times.

Explanation

Scheduled scaling configures capacity changes to occur at specific calendar times. It is useful when traffic patterns are predictable — for example, daily business hours, weekly promotions, or known maintenance windows. Using scheduled actions, you can increase capacity before an expected load spike so resources are ready when users arrive, and reduce capacity during predictable quiet periods to save cost.

Why the correct answers fit. The option that increases instance count before predictable daily peaks matches scheduled scaling because it creates capacity ahead of the known demand window rather than waiting for a metric signal. The option that lowers instances during off-peak hours aligns with scheduled scaling because it removes capacity at set times to cut unnecessary expenditure.

Why the incorrect answers are wrong. The option describing continuous adjustment to keep CPU at a target refers to targeting a metric level; that behavior is implemented by target-tracking scaling, which reacts to observed metrics rather than a clock. The option about using historical patterns to autonomously predict needs describes predictive scaling, which forecasts demand from past data. The option about responding to metric thresholds with fixed increments matches step-scaling, which triggers based on threshold breaches rather than on a schedule.

Key Concepts

- **Scheduled capacity change:** A deliberate capacity modification configured to run at a specific time or recurrence so resources align with expected demand.

- **Predictable traffic patterns:** Workload behaviors that repeat on a known schedule (daily, weekly) and can be planned for in advance.
- **Cost vs. responsiveness trade-off:** Scheduled scaling saves cost by removing capacity when not needed, but it requires accurate knowledge of future demand times.

Common Pitfalls

- Confusing scheduled actions with metric-driven autoscaling leads to choosing a reactive scaling strategy when a time-based action is required. Another mistake is relying on schedules when traffic is unpredictable; scheduled changes may not match real-time demand and can cause over- or under-provisioning.

Glossary

- **Auto Scaling group:** A logical collection of instances managed together for scaling purposes.
- **Target-tracking scaling:** A scaling approach that adjusts capacity to maintain a specified metric value.
- **Predictive scaling:** A method that forecasts future demand from historical data to schedule capacity proactively.

13.36 — APN partner types: ISV/SI/MSP (D4.T4.3)

Domain 4 • Task 4.3 ↑ Back to Question 13.36

A company has completed its migration to AWS. It wants a partner to operate the environment with 24x7 monitoring, patching, incident response, and ongoing cost optimization reviews, using least-privilege access in the company's AWS accounts. The company will keep its AWS Support Business plan for guidance. Which option best meets these needs?

Options & Rationales

A ✓ — Managed Service Provider (MSP) delivering ongoing operations under SLAs using AWS Systems Manager, Amazon CloudWatch, AWS CloudTrail, and AWS Config.

Rationale: MSPs run and optimize your environment post-go-live, providing 24x7 monitoring, patching, incident response, and cost reviews—often in your account with least-privilege access.

B ✗ — Systems Integrator (SI) to design the architecture, build a landing zone, and execute a migration project.

Rationale: SIs focus on project-based design and migration work. They are not chartered for continuous 24x7 operations.

C ✗ — Independent Software Vendor (ISV) subscription through AWS Marketplace for a monitoring tool.

Rationale: ISVs provide software (SaaS/AMIs/containers) often billed via AWS Marketplace; purchasing software does not include managed operations.

D ✗ — AWS Support Business plan to perform runbooks, patching, and incident response on the company's behalf.

Rationale: AWS Support provides guidance, best practices, and troubleshooting—not implementation or operating workloads.

Explanation

The AWS Partner Network includes distinct partner types that complement AWS Support. A Managed Service Provider (MSP) takes responsibility for operating and optimizing a customer's AWS environment after go-live. Typical MSP scope includes 24x7 monitoring, patching, incident response, and ongoing cost optimization aligned to the AWS Well-Architected Framework. MSPs commonly work in the customer's accounts with least-privilege access and use services such as AWS Systems Manager, Amazon CloudWatch, AWS CloudTrail, and AWS Config to deliver outcomes under SLAs. This exactly matches the scenario's need for around-the-clock operations and cost reviews while the company retains an AWS Support plan for guidance.

By contrast, a Systems Integrator (SI) provides consulting to design, migrate, and modernize workloads—typically on a project basis. They build landing zones and execute migrations but are not engaged for continuous operations. An Independent Software Vendor (ISV) sells software that runs on or integrates with AWS—often procured through AWS Marketplace and visible on the consolidated invoice. Buying software does not equate to having a team perform 24x7 operations. AWS Support plans (Developer, Business, Enterprise On-Ramp, Enterprise) offer guidance, troubleshooting, and best practices rather than implementation or running workloads. Therefore, only engaging a Managed Service Provider fulfills all operational requirements stated.

Key Concepts

- **Managed Service Provider (MSP):** Operates and optimizes environments post-go-live with 24x7 monitoring, patching, incident response, and cost reviews using least-privilege access.
- **Systems Integrator (SI):** Project-based consulting for design, migration, and modernization; not ongoing operations.
- **Independent Software Vendor (ISV):** Provides software via AWS Marketplace; appears on the AWS invoice but does not deliver managed operations.
- **AWS Support plans**: Provide guidance and best practices; they do not build or run workloads.

Common Pitfalls

- Expecting AWS Support to perform patching or incident response; those are partner-delivered operational tasks.
- Assuming purchasing a monitoring tool replaces 24x7 managed operations by a service provider.

13.37 — Use Well-Architected Tool for reviews (D1.T1.2)

Domain 1 • Task 1.2 ↑ Back to Question 13.37

What is the primary purpose of using the AWS Well-Architected Tool for a workload?

Options & Rationales

A ✗ — To automatically deploy a production architecture that meets compliance requirements without human review.

Rationale: The tool does not deploy architectures or replace human review; it provides assessments and guidance rather than automatic deployment.

B ✗ — To monitor real-time application performance metrics and send alerts for operational incidents.

Rationale: Real-time monitoring and alerting are functions of services like CloudWatch, not the primary role of the Well-Architected Tool.

C ✗ — To manage AWS billing and apply reserved pricing recommendations across accounts.

Rationale: Cost management and reserved pricing are handled by billing tools such as Cost Explorer and Savings Plans; the Well-Architected Tool focuses on architectural review and improvement guidance.

D ✓ — To evaluate a workload against the Well-Architected pillars, highlight risks, and get guidance to prioritise fixes.

Rationale: This describes the main function: assessing workloads versus pillars, identifying risks, and providing actionable remediation recommendations and prioritization.

Explanation

The Well-Architected Tool exists to review a workload's architecture against a set of best-practice areas (the Well-Architected pillars). Its job is to reveal design or operational risks, provide prioritized, actionable recommendations to address those risks, and help teams track improvements over time. It is a review and guidance instrument, not a deployment, monitoring, or billing automation tool.

Why the correct choice is right and others are wrong. The correct option states that the tool evaluates workloads against the Well-Architected pillars, identifies risks, and supplies guidance to prioritise fixes; this matches the tool's core purpose as an assessment and improvement framework. The incorrect options describe functions outside that scope: one claims automatic deployment and compliance without review, which misrepresents the tool because it does not provision architectures; another attributes real-time monitoring and alerting, which are responsibilities of observability services; and the final option assigns billing and reserved pricing management, which belong to AWS billing and cost tools rather than the Well-Architected Tool.

Key Concepts

- **Architectural assessment:** Comparing a workload's design to established pillars identifies gaps and weaknesses to address.
- **Risk identification and prioritization:** The tool surfaces risks and recommends remediation steps so teams can focus on the most important fixes first.
- **Guidance versus execution:** The tool provides actionable advice and improvement plans, while other AWS services perform deployment, monitoring, or cost management.

Common Pitfalls

- Mistaking the Well-Architected Tool for an automatic deployment or monitoring service leads to expecting real-time metrics or push-button builds, which it does not provide.
- Confusing cost management features with the tool's recommendations can cause users to search the wrong AWS service for billing actions.

Glossary

- **Well-Architected pillars:** The core domains of best practices used to evaluate architectures (for example, security, reliability, performance efficiency, cost optimization, and operational excellence).
- **Remediation guidance:** Practical suggestions the tool provides to reduce identified risks and improve the workload design.

13.38 — Innovation speed via services & global infra (D1.T1.1)

Domain 1 • Task 1.1 ↑ Back to Question 13.38

Which AWS service is purpose-built to improve network path performance and availability for globally distributed TCP/UDP applications?

Options & Rationales

A ✓ — AWS Global Accelerator

Rationale: Designed to improve path performance for TCP/UDP applications by leveraging managed acceleration across AWS's global infrastructure.

B ✗ — Amazon CloudFront

Rationale: Optimizes delivery of content from edge locations; it is a CDN for serving content, not a general TCP/UDP path acceleration service.

C ✗ — Amazon Route 53

Rationale: Provides highly available DNS and routing policies; it does not accelerate the end-to-end network path for applications.

D ✗ — Amazon CloudWatch

Rationale: Delivers monitoring and observability; it does not affect data-plane network performance.

Explanation

The question asks which managed AWS service is intended to improve network path performance for TCP/UDP applications used by a global audience. Among AWS's global and edge services, one is specifically identified for path performance of TCP/UDP traffic. That service is AWS Global Accelerator. It provides acceleration for application traffic by optimizing how user traffic reaches application endpoints, improving performance and availability for TCP/UDP-based workloads.

Amazon CloudFront is often confused with Global Accelerator because both improve global user experience. However, CloudFront serves content from edge locations as a content delivery network, focusing on caching and distribution of web content rather than optimizing generic TCP/UDP application paths. Amazon Route 53 supplies highly available DNS and routing policies, which help direct users to endpoints but do not themselves accelerate the data path. Amazon CloudWatch provides monitoring and logging capabilities for observability and does not impact the performance of the application traffic path.

Therefore, the only service in this list whose purpose aligns directly with improving path performance for TCP/UDP applications is AWS Global Accelerator.

Key Concepts

- **AWS Global Accelerator:** Improves path performance and availability for TCP/UDP applications accessed globally.
- **Amazon CloudFront:** Serves content from edge locations as a CDN to reduce latency for content delivery.
- **Amazon Route 53:** Highly available DNS with routing policies; directs traffic but does not accelerate the path.
- **Amazon CloudWatch:** Monitoring and observability; no direct effect on network path performance.

Common Pitfalls

- Confusing CloudFront's content caching at the edge with Global Accelerator's TCP/UDP path optimization. CloudFront accelerates content delivery, not general application network paths.

13.39 — AWS cloud value proposition (D1.T1.1)

Domain 1 • Task 1.1

↑ Back to Question 13.39

A startup is launching a new mobile API with unpredictable traffic spikes. The team wants to minimize operational overhead (no servers to patch), scale automatically within minutes, pay only for actual usage, and have high availability across multiple Availability Zones by default. Which approach best meets these goals while avoiding lift-and-shift overprovisioning?

Options & Rationales

A ✗ — Preprovision peak-capacity servers and run them 24/7 to handle spikes.

Rationale: Overprovisioned and cost-inefficient; not elastic and still requires server management.

B ✗ — Use AWS Global Accelerator alone to absorb spikes without changing the backend.

Rationale: Improves global performance, but does not add backend capacity or autoscaling.

C ✗ — Rely on AWS Budgets and AWS Cost Explorer to control costs while keeping the current design.

Rationale: These provide visibility and forecasts, not elasticity or architectural scaling.

D ✓ — Amazon CloudFront + Amazon S3 for static content; Amazon API Gateway + AWS Lambda for API logic; Amazon DynamoDB with automatic scaling.

Rationale: Serverless, managed, and inherently Multi-AZ; scales automatically and uses pay-per-use pricing, removing undifferentiated heavy lifting.

Explanation

The requirements call for elasticity, pay-as-you-go cost alignment, Multi-AZ high availability, and minimal operational effort. In AWS, managed and serverless services deliver these benefits by removing undifferentiated heavy lifting (such as server patching and cluster management) and scaling to meet demand. A serverless pattern using Amazon CloudFront and Amazon S3 for static assets, Amazon API Gateway and AWS Lambda for API logic, and Amazon DynamoDB with automatic scaling provides automatic, near-instant elasticity and Multi-AZ resilience by default. Because these services are fully managed, the team avoids maintaining servers, and costs closely track real usage (for example, pay-per-request or consumption-based models). This directly addresses unpredictable spikes without pre-purchasing or running idle capacity.

Preprovisioning peak capacity and running it 24/7 mimics on-premises, capacity-constrained infrastructure. It increases cost when demand is low and contradicts the pay-as-you-go model. It also retains operational burden for patching and scaling. Using AWS Global Accelerator can improve global performance and reduce latency by routing users more efficiently, but it does not create or scale backend capacity; without a scalable backend, spikes still overwhelm the system. Finally, AWS Budgets and AWS Cost Explorer are financial management tools for forecasting, monitoring, and alerting on spend. They do not provide elasticity or availability and cannot fix an overprovisioned lift-and-shift design.

Key Concepts

- **Elasticity and pay-as-you-go:** Provision only what you need and scale up or down in minutes so spend aligns with actual usage.
- **Managed/serverless services:** Offload patching, backups, and infrastructure tasks to AWS to reduce operational overhead and improve reliability by default.
- **Multi-AZ high availability:** Using AWS managed services typically provides built-in resilience across Availability Zones without extra setup.
- **Cost visibility vs. architecture:** Budgets and Cost Explorer inform spending; they do not replace architectural choices that deliver elasticity and efficiency.

Common Pitfalls

- Assuming a network accelerator will handle traffic spikes without a scalable backend.
- Keeping a lift-and-shift, overprovisioned design and expecting budgets or forecasts to deliver cost savings or elasticity.

13.40 — Pay-per-use aligns cost with consumption (D1.T1.4)

Domain 1 • Task 1.4 ↑ Back to Question 13.40

Which statement best describes how metered pricing helps organizations pay only for the resources they use?

Options & Rationales

A ✗ — Customers pay a fixed monthly fee regardless of how much they use services, providing predictable billing but not usage alignment.

Rationale: A fixed subscription does not vary with consumption and therefore does not align cost to actual resource use.

B ✗ — Organizations reserve capacity for a long term and receive a discount, which charges based on committed capacity rather than measured consumption.

Rationale: Reserved capacity ties cost to pre-purchased resources and reduces variability, but charges are based on commitment, not precise measured use.

C ✓ — Billing is calculated from measured resource consumption (for example, compute-hours or data transferred), so charges reflect actual use over time.

Rationale: Metered pricing records units of usage (such as instance-hours or GB transferred) and invoices based on those measured units, aligning cost with actual consumption and enabling rightsizing decisions.

D ✗ — Costs are allocated per named user account so each individual pays only for their own activity, independent of resource metrics.

Rationale: Per-user licensing assigns costs to identities rather than to measured resource units and does not inherently meter resource consumption like compute-hours or GB-hours.

Explanation

Metered pricing charges customers according to measured units of resource use (for example, compute-hours, storage gigabyte-months, or data transfer). A metering system records consumption metrics and maps them to billing units so invoices reflect what was actually consumed during the billing period. This model helps organizations avoid paying for idle capacity and supports decisions to rightsize or shift to more efficient options.

Why the correct choice is right. It describes charging based on measured resource consumption over time (such as instance-hours or GB transferred), which is the central mechanism that aligns costs with actual use.

Why the incorrect choices are wrong. The fixed-fee option describes subscription pricing, which provides predictability but not per-use alignment. The reserved-capacity option explains committed discounts tied to purchased capacity rather than dynamic measurement. The per-user allocation option ties cost to identities or seats rather than resource usage metrics, so it does not achieve the same consumption-based billing.

Key Concepts

- **Metered usage:** Recording measurable units (for example, compute-hours or gigabytes) so billing corresponds to resources consumed.
- **Rightsizing:** Using consumption data to adjust resource capacity and reduce waste when demand is lower than provisioned capacity.
- **Subscription vs. consumption models:** Subscriptions give fixed charges; metered models vary charges with actual use.

Common Pitfalls

- Confusing reserved or subscription pricing with metered billing will lead to thinking discounts equal usage alignment; they do not.
- Assuming per-user charges equal resource metering; user-based fees do not necessarily reflect resource units consumed.

Glossary

- **Metering:** The process of measuring resource consumption in defined units for billing.
- **Rightsizing:** Adjusting resource size or quantity to match observed demand and reduce cost.

13.41 — Regions vs Availability Zones vs Local Zones (D3.T3.2)

Domain 3 • Task 3.2

↑ Back to Question 13.41

A company runs customer-facing APIs in an AWS Region. They must: 1) deliver single-digit millisecond latency for a real-time processing component to an on-premises system in a nearby metro area, and 2) increase resilience against

data center failures within the Region. Which combination of actions will best meet these requirements?

Options & Rationales

A ✗ — Choose the Region with the most Availability Zones to guarantee the lowest latency to users and on-premises systems.

Rationale: Region choice should be based on latency to users, compliance, available services, and cost. Having more AZs does not guarantee lower latency.

B ✗ — Create multiple subnets within one Availability Zone to improve resilience against data center failures.

Rationale: An AZ is the failure boundary. Multiple subnets inside a single AZ do not protect against an AZ outage.

C ✗ — Deploy duplicate environments in two separate AWS Regions to achieve high availability within the Region.

Rationale: Cross-Region architectures are a disaster recovery strategy and do not replace Multi-AZ high availability within one Region; they also don't address metro ultra-low latency.

D ✗ — Run all production resources in a single Local Zone to meet both latency and availability needs.

Rationale: A single Local Zone is not a high-availability substitute. For resilience, use multiple AZs in the parent Region.

E ✓ — Deploy the application across at least two Availability Zones in the parent Region using an Elastic Load Balancer and Amazon RDS Multi-AZ.

Rationale: AZs are fault-isolation boundaries. Spreading resources across AZs with load balancing and RDS Multi-AZ provides automatic in-Region failover and higher availability.

F ✓ — Opt in to the nearest AWS Local Zone and run the real-time component on Amazon EC2 with Amazon EBS in that Local Zone.

Rationale: Local Zones extend a Region to metro areas and offer selected services (e.g., EC2 and EBS) for single-digit millisecond latency; they are opt-in per account.

Explanation

Foundational AWS building blocks have different scopes and purposes. An AWS Region is a separate geographic area and a data residency boundary. Regions are isolated from each other, and you choose one based on proximity/latency to users, compliance, available services, and cost. Within a Region, Availability Zones (AZs) are one or more discrete data centers engineered for physical separation and connected by high-throughput, low-latency links. AZs act as fault-isolation boundaries; deploying across multiple AZs increases availability and resilience to a single facility failure. Typical high-availability patterns

include placing Amazon EC2 instances in subnets across at least two AZs behind an Elastic Load Balancer and using Amazon RDS Multi-AZ for automatic failover. Some services like Amazon S3 and Amazon EFS are regional and replicate across multiple AZs by design.

AWS Local Zones extend a Region by placing selected compute, storage, and networking services closer to large metro areas, delivering single-digit millisecond latency to end users or on-premises environments. Local Zones are anchored to a parent Region, have an opt-in model per account, and offer a limited set of services such as EC2 and EBS. A common pitfall is treating a Local Zone like an AZ for high availability: a single Local Zone does not replace multi-AZ designs in the parent Region.

Applying these principles, the mixed requirement calls for two complementary moves. To meet the ultra-low-latency need to a metro-area on-premises system, use a Local Zone and run the real-time component on EC2 with EBS in that Local Zone. To increase resilience against data center failures within the Region, deploy the application across at least two AZs in the parent Region, fronted by an Elastic Load Balancer and with database tiers (if used) configured for Amazon RDS Multi-AZ. This design uses the Local Zone only where latency is critical and relies on multi-AZ architecture for availability.

Proposals to place everything in a single Local Zone conflate latency with availability and ignore that Local Zones are not an HA substitute. Spreading stacks across different Regions targets disaster recovery, not in-Region high availability, and also doesn't inherently solve the metro latency requirement. Choosing a Region solely because it has more AZs misunderstands that AZ count does not guarantee lower latency; proximity and network path do. Creating multiple subnets within one AZ does not protect against an AZ outage because the AZ itself is the failure boundary.

Key Concepts

- **Region vs. AZ vs. Local Zone:** Regions are isolated geographic areas; AZs are fault-isolated facilities within a Region; Local Zones bring selected services closer to metro areas for low latency.
- **Availability Zones as HA boundaries:** Designing across at least two AZs (e.g., EC2 behind an Elastic Load Balancer, RDS Multi-AZ) improves in-Region availability and automatic failover.
- **Local Zones for ultra-low latency:** Opt-in, anchored to a parent Region, limited services (e.g., EC2/EBS); not a replacement for multi-AZ resilience.

Common Pitfalls

- Treating a single Local Zone as an HA solution, or assuming more AZs in a Region guarantee lower latency. Multi-AZ within the Region is the primary HA mechanism; Local Zones address latency-specific needs.

13.42 — Edge locations in CloudFront (D3.T3.2)

Domain 3 • Task 3.2 ↑ Back to Question 13.42

A media company needs to speed up delivery of images and videos to viewers worldwide. Which CloudFront feature best meets this requirement?

Options & Rationales

A ✗ — Large regional storage centers intended for long-term archival of original media files.

Rationale: Regional archival storage is for long-term retention and does not speed up global content delivery to viewers.

B ✗ — The origin servers that permanently host the original images and videos.

Rationale: Origin servers store the primary copy but serving from them directly increases latency for distant users compared to nearest caches.

C ✗ — Private network endpoints used to create VPN connections between data centers.

Rationale: Private connectivity endpoints provide secure links but do not cache or accelerate public content delivery to end users.

D ✓ — Physical sites that cache frequently requested files near viewers to lower latency and reduce load on the origin.

Rationale: Edge locations cache content close to users so repeat requests are served faster and the origin receives fewer direct requests.

Explanation

Edge points of presence are distributed sites that keep copies of frequently accessed objects close to end users. By serving cached responses from these nearby locations, user-perceived latency decreases and the origin infrastructure handles fewer requests. This model is specifically designed for improving performance of static and cacheable web assets such as images and video segments.

Why the correct choice fits. it describes the distributed cache behavior that speeds repeated content delivery and reduces origin load, which directly addresses global performance needs.

Why the other choices are incorrect. the regional archival option refers to long-term storage, not low-latency delivery; the origin server option describes the authoritative content store, which is further from many users and increases latency if used for every request; and the private endpoint option relates to secure networking rather than content caching or acceleration.

Key Concepts

- **Content caching:** Storing responses closer to users to serve repeated requests faster and avoid repeated origin fetches.

- **Latency reduction**: Serving content from a nearby location decreases round-trip time and improves user experience.
- **Origin offload**: Caching reduces the number of requests that reach the primary storage or server, lowering origin load and bandwidth use.

Common Pitfalls

- Confusing the origin (source of truth) with the cache; using the origin for all requests increases latency.
- Thinking private network endpoints accelerate public content delivery; they secure connectivity but do not provide distributed caching.

Glossary

- **Edge location**: A geographically distributed site that stores cached copies of content to serve users with lower latency.
- **Origin**: The primary storage or server where original content is kept.

13.43 — Org change management for adoption (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 13.43

A company moving applications to AWS faces slow decision-making and stalled migration milestones. Which action most directly removes organizational barriers and ensures timely cloud adoption decisions?

Options & Rationales

A ✗ — Run a comprehensive role-based cloud training program for all engineers before migrating any workloads.

Rationale: Training improves skills but does not by itself resolve slow executive decisions or remove governance bottlenecks.

B ✓ — Assign a senior leader as visible sponsor with clear decision authority and regular status reviews.

Rationale: A senior sponsor with authority speeds decisions, aligns stakeholders, and enforces priorities to prevent stalls during migration.

C ✗ — Implement automated technical guardrails and IAM policies to enforce best practices across accounts.

Rationale: Guardrails help compliance and security, yet they do not address leadership-level decision delays that block migration progress.

D ✗ — Create financial incentives tied to cost savings and publish adoption KPIs monthly.

Rationale: Incentives and metrics can motivate teams but may be too slow to clear immediate decision obstacles without leadership direction.

Explanation

Organizational obstacles during cloud adoption often stem from unclear authority and competing priorities. Appointing a senior leader as a visible

sponsor who has explicit decision-making power and regularly reviews progress directly addresses these governance bottlenecks. This role can resolve conflicts, prioritize resources, and communicate executive expectations across teams, enabling migration milestones to move forward.

Assigning only technical measures, training, or incentives can help in parallel but do not by themselves remove the need for decisive leadership when cross-team trade-offs or resource reallocation are required.

Why the other choices are incorrect. The option proposing comprehensive role-based training is valuable for skills but does not assign accountability to resolve policy or resource conflicts. The suggestion to implement automated guardrails improves security and compliance, yet such measures cannot arbitrate competing business priorities. Creating incentives and publishing KPIs may encourage adoption but is unlikely to remove immediate decision-making roadblocks without an empowered sponsor.

Key Concepts

- **Visible executive sponsorship:** A senior leader actively supports the initiative, communicates priorities, and signals organizational commitment, which aligns stakeholders and reduces resistance.
- **Decision authority:** Clear delegation of who can make binding choices prevents delays caused by escalation chains and competing managers.
- **Governance with cadence:** Regular status reviews create an enforced rhythm for decisions, enabling timely unblock and adjustments.

Common Pitfalls

- Assuming training or technical controls will fix pace issues without assigning accountability and authority for cross-team decisions.
- Relying solely on metrics or incentives before establishing who resolves disputes about scope, budget, or risk.

13.44 — RI scope: Regional vs AZ (D4.T4.1)

Domain 4 • Task 4.1 ↑ Back to Question 13.44

An online learning platform runs a stateless web tier on Amazon EC2 Auto Scaling across two Availability Zones in one Region. Daily traffic shifts between AZs. The team uses Linux/Unix with default tenancy and sometimes changes instance sizes within the same family. They want to reduce costs, keep elasticity, and avoid locking capacity to a specific AZ. Which purchase choice best meets these goals?

Options & Rationales

A ✗ — Zonal Reserved Instances in one AZ for a specific instance size; includes a capacity reservation and discount only in that AZ.

Rationale: This locks both capacity and the discount to a single AZ and exact size. With shifting traffic, utilization can drop in that AZ, increasing the

effective rate.

B ✗ — Zonal Reserved Instances split evenly across both AZs for the exact size; provides capacity reservations in each AZ with discounts tied to each AZ.

Rationale: Discounts remain AZ-bound and size-specific, so workload shifts can still leave one AZ underutilized. This also adds capacity reservations the team does not need.

C ✓ — Regional Reserved Instances for Linux/Unix (default tenancy) in the chosen instance family; discount applies across all AZs with instance size flexibility; no capacity reservation.

Rationale: Regional RIs apply the billing discount to matching usage in any AZ in the Region and provide instance size flexibility for Linux/Unix default tenancy, maximizing utilization without reserving capacity.

D ✗ — Use only On-Demand Instances and rely on Auto Scaling to shift load between AZs to minimize cost.

Rationale: Auto Scaling improves elasticity, not the unit price. On-Demand has no RI discount, so it does not meet the primary cost-reduction goal.

Explanation

Reserved Instances (RIs) can be scoped Regionally or to a specific Availability Zone (Zonal). Scope determines how the pricing benefit applies and whether capacity is also reserved. A Regional RI provides a billing discount that automatically applies to all matching EC2 usage in the chosen Region, across any AZ. For Linux/Unix with default tenancy, Regional RIs also include instance size flexibility within the same instance family (for example, discounts can apply to different sizes like large or xlarge). Regional scope does not include a capacity reservation. By contrast, a Zonal RI ties the reservation to one AZ and guarantees capacity for the exact attributes you select. Its discount applies only to matching usage in that AZ, and size flexibility does not apply.

Given the scenario's requirements—traffic shifts between AZs, Linux/Unix default tenancy, occasional size changes within a family, and no need for guaranteed capacity—the best fit is a Regional RI. It allows the discount to follow the workload regardless of which AZ is busier on a given day and automatically accommodates instance size changes within the family, improving utilization and lowering effective cost. Zonal RIs, whether placed in a single AZ or split across both AZs, keep the discount tied to each AZ and the exact size, which risks underutilization as traffic shifts and provides unnecessary capacity reservations. On-Demand pricing retains elasticity but offers no RI discount.

Underutilization matters because RI costs are fixed over the term. The simple relationships are $\text{Utilization} = \frac{\text{discounted hours}}{\text{purchased hours}}$ and $\text{Effective Rate} = \frac{\text{total RI cost}}{\text{hours actually applied}}$. As utilization drops, the effective rate rises. Regional scope typically increases utilization for AZ-shifting workloads by letting the discount

apply wherever the usage appears.

Variables used:

- **Utilization:** Fraction of purchased RI hours that matched running instances [unitless]
- **discounted hours:** Hours where the RI discount applied [hours]
- **purchased hours:** Total RI hours bought over the term [hours]
- **Effective Rate:** Average cost per discounted hour [cost/hour]
- **total RI cost:** Fixed spend for the RI term [cost]
- **hours actually applied:** Same as discounted hours [hours]

Key Concepts

- **RI Scope (Regional vs. Zonal):** Regional = discount across all AZs, no capacity; Zonal = AZ-bound discount with capacity reservation.
- **Instance Size Flexibility:** For Linux/Unix default tenancy, Regional RIs apply across sizes within the same instance family.
- **Utilization and Effective Rate:** Higher utilization lowers the RI's effective per-hour cost.

Common Pitfalls

- Assuming splitting Zonal RIs across AZs mimics Regional flexibility; the discount still remains AZ-bound.
- Believing Auto Scaling alone reduces price; it manages capacity but does not add an RI discount.

13.45 — AWS Config rules for compliance (D2.T2.2)

Domain 2 • Task 2.2 ↑ Back to Question 13.45

A compliance team uses AWS Config to evaluate resource conformity and wants to enforce fixes when violations occur. Which statement about Config remediation is NOT correct?

Options & Rationales

A ✗ — You can configure automatic remediation to run a Lambda function when a resource is marked noncompliant.

Rationale: AWS Config can trigger a Lambda-based remediation action automatically for noncompliant resources.

B ✓ — All remediation requires manual approval before any corrective action can be executed by Config.

Rationale: Config supports automatic remediation; manual approval is optional, not mandatory.

C ✗ — Remediation actions can be grouped and deployed as reusable policy packages for consistent enforcement across accounts.

Rationale: Collections of rules and their remediation can be packaged for consistent deployment across environments.

D ✗ — Remediation can be used to restore desired configurations when drift is detected by an evaluation.

Rationale: Remediation is intended to correct resources that deviate from the declared desired state after evaluations detect drift.

Explanation

AWS Config evaluates resource settings against declared expectations and can initiate corrective steps when a resource is found out of compliance. Remediation refers to actions that restore or correct resource configuration so the environment matches the intended state. These actions may be automated (for immediate correction) or set to require human approval depending on how the team configures the response. Organizations often package related rules and their corrective actions together to apply consistent policies across accounts.

Why the incorrect statement is wrong. The claim that every remediation must have manual approval is inaccurate. While manual approval can be chosen for safety-sensitive actions, AWS Config supports automatic remediation workflows that run without human intervention when a noncompliant evaluation occurs.

Why the other statements are correct. The statement describing Lambda-based automatic remediation is valid because AWS Config can invoke Lambda functions as remediation steps. The note about grouping rules and remediations into reusable policy packages aligns with the concept of packaging consistent rule sets and corrective measures for deployment. The description that remediation restores desired configurations after drift matches the purpose of remediation: correcting deviations discovered by evaluations.

Key Concepts

- **Remediation action:** A corrective operation (automatic or manual) that attempts to return a resource to the declared desired configuration after a noncompliant evaluation.
- **Automatic remediation:** A configured workflow that executes corrective steps without human approval; useful for low-risk, repeatable fixes.
- **Conformance packaging:** Bundling rules and their remediation for consistent policy deployment across environments.

Common Pitfalls

- Assuming remediation is always manual leads to overly cautious designs and missed opportunities for automation.
- Believing automatic remediation is appropriate for every situation can introduce risk; choose automation only for safe, well-tested actions.

Glossary

- **Noncompliant evaluation:** The result of comparing a resource's actual settings to the declared desired state and finding a mismatch.
- **Lambda remediation:** Using an AWS Lambda function as the corrective step invoked by Config.

13.46 — Avoid SPOF with multi-AZ (D3.T3.2)

Domain 3 • Task 3.2 ↑ Back to Question 13.46

A company wants to reduce the chance that a single datacenter failure makes its web application unavailable. Which design choice best addresses this single point of failure concern?

Options & Rationales

A ✓ — Deploy redundant application instances in multiple Availability Zones and keep data synchronized.

Rationale: Running instances across separate AZs provides independent fault domains; synchronized copies let traffic fail over without a single datacenter causing total outage.

B ✗ — Run all application servers in one AZ but use larger instance types for capacity.

Rationale: Larger instances increase capacity but still depend on a single AZ, so they don't remove the single-point-of-failure risk from a datacenter outage.

C ✗ — Store a backup of the database in the same AZ and perform daily manual restore tests.

Rationale: Same-AZ backups are vulnerable to the same outage; daily manual restores may be slow and do not prevent downtime during the AZ failure window.

D ✗ — Use a single, highly available EC2 instance with local storage and scheduled snapshots to S3.

Rationale: A single instance with local storage is still a single point of failure; snapshots help recovery but don't avoid immediate outage if the instance or AZ fails.

Explanation

To avoid a single point of failure caused by a datacenter outage, distribute active resources across separate failure domains. Placing redundant application instances in different Availability Zones ensures that if one AZ becomes unavailable, other AZs can continue serving traffic. Keeping data synchronized across those AZs allows seamless failover and preserves availability.

Why the other approaches fail:

- Relying on larger instances in one AZ improves capacity but does not protect against AZ-level faults; the workload still depends on a single physical location.
- Backups stored in the same AZ remain exposed to the same outage; restoring from those backups can take significant time and does not provide continuous availability.
- A single EC2 instance, even with snapshots, is still one running endpoint; snapshots are useful for recovery but do not prevent service interruption

when the instance or its AZ fails.

Key Concepts

- **Multi-AZ redundancy:** Run active resources in multiple isolated AZs so an outage in one zone does not bring down the entire service.
- **Fault domain isolation:** Availability Zones are separate location groups designed to limit a failure's blast radius.
- **Synchronized data copies:** Keeping data consistent across AZs supports fast failover and continued operations during an outage.

Common Pitfalls

- Assuming bigger hardware in one AZ equals higher availability; it does not mitigate AZ-level failures.
- Treating backups as a substitute for an active cross-AZ replication strategy; backups help recovery but not immediate availability.

13.47 — Permissions for programmatic automation (D3.T3.1)

Domain 3 • Task 3.1

↑ Back to Question 13.47

Which statement best describes the purpose of an IAM role's trust policy in AWS?

Options & Rationales

A ✗ — It defines the actions, resources, and conditions the role is allowed to perform in AWS.

Rationale: This describes the role's identity-based permission policies, not the trust policy. Permission policies state what the role can do after it is assumed.

B ✗ — It encrypts temporary credentials and manages automatic rotation for the role.

Rationale: Credential issuance and rotation are handled by AWS STS via short-lived credentials. Trust policies do not perform encryption or key rotation tasks.

C ✗ — It allows target services like Amazon S3 or AWS Lambda to grant the role access using a resource-based policy.

Rationale: Resource-based policies live on the resource (e.g., S3 bucket policy, Lambda resource policy). They are separate from a role's trust policy.

D ✓ — It specifies the trusted principals (for example, lambda.amazonaws.com, ec2.amazonaws.com, or another AWS account) that can assume the role via AWS STS.

Rationale: A trust policy defines who or what can obtain temporary credentials for the role by assuming it. It does not grant permissions to perform actions on AWS resources.

Explanation

IAM roles are designed for temporary access via AWS Security Token Service (STS), which is recommended for programmatic automation instead of long-term access keys. A role has two distinct policy layers with different purposes. The trust policy declares the set of trusted principals—such as AWS services (lambda.amazonaws.com, ec2.amazonaws.com) or external AWS accounts—that are allowed to assume the role. This controls who can obtain temporary credentials for the role. Separately, identity-based permission policies attached to the role define which actions, on which resources, and under what conditions the role can perform once assumed.

Resource-based policies are another policy type applied directly to resources like Amazon S3 buckets, AWS Lambda functions, and AWS KMS keys. These policies explicitly authorize (or deny) callers at the resource. When combined, access is granted only if the principal is allowed to assume the role (trust policy), and the resulting identity has permissions to act (permission policies), and the resource policy permits the call. Any explicit deny—such as within a resource policy—overrides allows. Least privilege is achieved by scoping actions to specific ARNs and using conditions (for example, aws:SourceAccount, aws:SourceArn, aws:RequestTag).

The correct statement is the one that defines a trust policy as specifying which principals can assume the role via STS. The statement about defining actions, resources, and conditions describes permission policies, not the trust policy. The claim that a trust policy encrypts or rotates credentials is incorrect; STS issues short-lived credentials and rotation is not a trust policy function. The statement asserting that trust policies allow resources like S3 or Lambda to grant access conflates resource-based policies, which are attached to the resource, with a role's trust configuration.

Key Concepts

- **IAM Role Trust Policy:** Declares who or what can assume the role via STS; controls the trusted principal set.
- **Identity-based Permission Policies:** Specify allowed actions, resources, and conditions for the role after assumption; enforce least privilege.
- **Resource-based Policies:** Applied to resources (S3, Lambda, KMS) to explicitly authorize or deny access; explicit deny prevails.

Common Pitfalls

- Confusing trust policies with permission policies, assuming the trust policy grants resource actions rather than only controlling who can assume the role.

13.48 — Post-migration optimization for cost/perf (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 13.48

Which post-migration activity focuses on changing the sizes or types of compute and storage resources so they match actual workload demand and

reduce cost?

Options & Rationales

A ✗ — Purchasing long-term capacity commitments such as Reserved Instances or Savings Plans to get discounts.

Rationale: Buying discounted commitments reduces cost for steady usage, but does not directly adjust provisioned resource sizes to match changing demand.

B ✓ — Adjusting instance sizes and storage configurations to better reflect observed usage patterns.

Rationale: This is the core right-sizing activity: resize or change resource types to remove over-provisioning and lower costs while meeting performance needs.

C ✗ — Configuring Auto Scaling to add or remove instances automatically based on load.

Rationale: Auto Scaling dynamically changes capacity in response to load, which complements right-sizing but is a different activity focused on elasticity rather than resizing existing resources.

D ✗ — Moving infrequently accessed objects to a cheaper storage tier to lower storage charges.

Rationale: This is storage tiering (lifecycle management) to optimize storage cost; it targets data placement rather than resizing compute or primary storage allocations.

Explanation

Right-sizing is the practice of modifying allocated compute and storage resources so they match actual workload needs. The goal is to eliminate excess capacity that causes unnecessary expense while preserving required performance. Typical actions include changing virtual machine types, reducing over-large volumes, or switching to more efficient instance families after observing usage patterns.

Why the correct choice fits. Adjusting instance sizes and storage configurations directly targets provisioned capacity and aligns resources with measured demand. This reduces wasteful spending caused by over-provisioning and can improve cost efficiency without changing application design.

Why the other choices are not the best answer. Purchasing long-term capacity commitments lowers cost for predictable usage but does not change the size or type of currently provisioned resources; it assumes steady demand rather than addressing mismatched capacity. Configuring Auto Scaling provides automatic elasticity by adding or removing instances based on load; it complements right-sizing but focuses on dynamic scaling, not permanently resizing existing resources. Moving rarely accessed objects to cheaper storage tiers optimizes data placement and storage cost, which is a different optimization activity than resizing compute or primary storage.

Key Concepts

- **Right-sizing:** Continuously compare allocated resources to actual metrics and reduce or change resources to remove over-provisioning.
- **Elasticity vs. provisioning:** Elastic features (for example, automatic scaling) adjust capacity dynamically; right-sizing changes the baseline allocation to be more efficient.
- **Resource telemetry:** Use monitoring data (metrics and utilization reports) to make informed resizing decisions rather than guessing.

Common Pitfalls

- Reducing sizes without monitoring can harm performance; always verify workload behavior and test changes. Over-relying on discounts (Reserved Instances/Savings Plans) without first eliminating waste can lock in payment for oversized resources.

Glossary

- **Right-sizing:** The activity of resizing or changing resource types to better match workload demand and cost objectives.
- **Auto Scaling:** A mechanism to add or remove capacity automatically based on defined policies and observed load.

13.49 — TLS & mechanisms for data in transit (D2.T2.2)

Domain 2 • Task 2.2 ↑ Back to Question 13.49

A company exposes a partner-facing API over the internet. Security requires that both the client and the server present X.509 certificates during the HTTPS handshake. Which AWS-supported mechanism directly satisfies this two-way certificate authentication requirement?

Options & Rationales

A ✗ — Standard TLS with only a server certificate

Rationale: Typical TLS validates the server but not the client by certificate, failing the requirement for two-way authentication.

B ✗ — MACsec on an AWS Direct Connect link

Rationale: MACsec secures a Direct Connect physical link at Layer 2; it does not provide mutual certificate checks for HTTPS clients.

C ✓ — Mutual TLS (mTLS) for the API connection

Rationale: mTLS enforces two-way certificate exchange during the TLS handshake, so both client and server are authenticated while encrypting the session.

D ✗ — IPSec site-to-site VPN between the networks

Rationale: An IPsec VPN encrypts traffic at the network layer but does not provide per-client certificate authentication at the HTTPS session level.

Explanation

Encrypting data in transit can occur at different layers. Transport Layer Security (TLS) protects application sessions such as HTTPS. In its common form, the server presents a certificate so clients can authenticate the server and establish an encrypted channel. Mutual TLS (mTLS) extends this by requiring the client to present its own certificate during the same handshake, enabling the server to authenticate the client using certificate-based identity. This directly fulfills scenarios where both parties must prove identity at the application layer while encrypting the connection.

The correct choice is mutual TLS because it implements two-way certificate authentication as part of the HTTPS handshake. Standard TLS with only a server certificate encrypts the connection and authenticates the server but does not authenticate clients by certificate, so it fails the discriminator. An IPsec site-to-site VPN provides network-layer encryption between endpoints but does not establish per-client certificate identity for individual HTTPS connections. MACsec secures link-layer traffic on dedicated Direct Connect circuits and is unrelated to client/server certificate exchange for internet-facing APIs.

Key Concepts

- **TLS vs. mTLS**: TLS commonly authenticates the server; mTLS requires certificates from both client and server for two-way authentication.
- **Layer of protection**: Application-layer TLS/mTLS secures HTTPS sessions; network/link-layer methods (IPsec, MACsec) do not provide per-client certificate identity.
- **Certificate-based identity**: X.509 certificates enable cryptographic proof of identity during the handshake.

Common Pitfalls

- Assuming any encrypted tunnel (e.g., IPsec) automatically provides client certificate authentication for HTTPS; it does not.

13.50 — Lightsail for simple VPS use cases (D3.T3.3)

Domain 3 • Task 3.3

↑ Back to Question 13.50

A startup runs a WordPress site on Amazon Lightsail. To increase availability, they will add a Lightsail load balancer and place instances in different Availability Zones. They also schedule one automated snapshot per day for each instance. During a 28-day month, what is the minimum number of Lightsail instances they should run, and how many instance snapshots will be created for the month?

Options & Rationales

A ✗ — Two Amazon Lightsail instances across two Availability Zones; 28 snapshots in 28 days

Rationale: The snapshot schedule is per instance. Two instances over 28 days

produce 56, not 28, snapshots.

B ✗ — Three Amazon Lightsail instances across three Availability Zones; 84 snapshots in 28 days

Rationale: Arithmetic is correct (3 × 28), but this is not the minimum. Multi-AZ is satisfied with two instances.

C ✓ — Two Amazon Lightsail instances in different Availability Zones behind a Lightsail load balancer; 56 snapshots in 28 days

Rationale: Multi-AZ needs at least two instances (one per AZ). With one snapshot per instance per day: 2 × 28 = 56 total.

D ✗ — One Amazon Lightsail instance with a static public IP; 28 snapshots in 28 days

Rationale: A single instance cannot meet a Multi-AZ requirement and remains a single point of failure.

Explanation

Amazon Lightsail includes a managed load balancer that can distribute traffic across instances placed in different Availability Zones (AZs). To achieve high availability across AZs, there must be at least one healthy instance in each AZ receiving traffic. Therefore, the minimum configuration that satisfies a Multi-AZ goal is two Lightsail instances, each in a different AZ, registered behind the Lightsail load balancer.

The snapshot requirement states one automated snapshot per day for each instance. Over a 28-day month, the total number of instance snapshots is the product of the number of instances and the number of days. With two instances, this is 2 × 28 = 56 snapshots. This meets the backup goal while maintaining the minimal instance count for Multi-AZ availability.

The proposal with two instances but only 28 snapshots undercounts because it ignores that the schedule applies per instance. A design with three instances would create 84 snapshots but is not the minimum needed to satisfy Multi-AZ. A single instance, even with a static public IP, cannot be Multi-AZ and remains a single point of failure, which fails the availability objective the load balancer is intended to address.

Formula:

$$S = I \times D$$

Variables used:

- **I:** Number of Lightsail instances [count]
- **D:** Number of days in the month [days]
- **S:** Total instance snapshots created in the month [count]

Key Concepts

- **Multi-AZ high availability:** Requires capacity in at least two Availability Zones so the workload survives the loss of one AZ.
- **Lightsail load balancer:** Simplifies distribution and TLS termination across Lightsail instances placed in different AZs.
- **Snapshot scheduling is per resource:** Daily snapshots per instance aggregate linearly with the number of instances.

Common Pitfalls

- Assuming a load balancer alone provides Multi-AZ without multiple instances across AZs.
- Counting one daily snapshot total instead of one per instance per day, leading to half the correct total.

13.51 — S3 storage classes & use cases (D3.T3.6)

Domain 3 • Task 3.6 ↑ Back to Question 13.51

Which Amazon S3 storage class is designed for infrequently accessed objects that are stored in a single Availability Zone to reduce cost?

Options & Rationales

A ✗ — S3 Standard (frequent-access, multi-AZ high availability)

Rationale: Optimized for frequent access with multi-AZ redundancy and low latency, not for lower-cost single-AZ infrequent storage.

B ✗ — S3 Standard-IA (infrequent access with rapid retrieval across multiple AZs)

Rationale: Provides infrequent-access storage with multi-AZ resilience and quick retrieval, so it is not the single-AZ, lower-cost option.

C ✓ — S3 One Zone-IA (infrequent access in a single AZ)

Rationale: Stores infrequently accessed objects in one Availability Zone at a lower cost than multi-AZ classes; suitable when data can be recreated if an AZ is lost.

D ✗ — S3 Glacier Instant Retrieval (archival with millisecond retrieval)

Rationale: An archival class that supports very low-latency retrieval but is intended for long-term archive, not single-AZ infrequent storage for cost savings.

Explanation

Amazon S3 offers multiple storage classes that balance cost, availability, and retrieval characteristics. A class that places objects in a single Availability Zone reduces redundancy and therefore costs compared with multi-AZ classes, so it is suitable when data can be reproduced if that AZ becomes unavailable. This class is intended for objects that are not accessed often but still need occasional, rapid retrieval.

Why the correct choice fits. S3 One Zone-IA stores infrequently accessed objects in one Availability Zone to deliver lower storage pricing than multi-AZ infrequent classes. It maintains the durability of objects within the single AZ but does not provide the multi-AZ resilience of other S3 classes, so it should be chosen only when lower cost is more important than AZ-level fault tolerance.

Why the other choices are incorrect. S3 Standard is designed for frequent-access workloads and provides multi-AZ availability and low latency, making it inappropriate when the primary goal is lower-cost, single-AZ infrequent storage. S3 Standard-IA offers infrequent-access storage with multi-AZ redundancy and rapid retrieval, so it does not match the single-AZ cost-optimized requirement. S3 Glacier Instant Retrieval is an archival option with very low-latency retrieval but is intended for archive purposes rather than the specific single-AZ, infrequent-access cost trade-off.

Key Concepts

- **Availability Zone (AZ) placement:** Stores data within one physical AZ or across multiple AZs; single-AZ placement reduces cost but lowers fault tolerance.
- **Infrequent access storage:** Designed for objects accessed rarely; balances lower storage cost with retrieval charges and sometimes minimum storage durations.
- **Durability vs. availability trade-off:** Higher redundancy (multi-AZ) increases availability and resilience but also increases cost compared with single-AZ options.

Common Pitfalls

- Assuming lower-cost classes always replicate across AZs; some low-cost options intentionally limit replication to one AZ.
- Confusing archival classes with infrequent-access classes; archival classes target long-term retention and different retrieval patterns.

Glossary

- **AZ (Availability Zone):** An isolated location within an AWS Region that provides independent power, networking, and cooling.
- **Infrequent access:** A usage pattern where objects are stored for long periods but are retrieved rarely.

13.52 — Security guidance: Blog & Knowledge Center (D2.T2.4)

Domain 2 • Task 2.4 ↑ Back to Question 13.52

A regulated business must keep an immutable, searchable record of every API call and account activity across its AWS account for 365 days to support audits. Which AWS service is designed to provide these immutable API activity and event logs for that retention requirement?

Options & Rationales

A ✗ — Amazon CloudWatch Logs — collects metrics and application logs for monitoring and short-term troubleshooting.

Rationale: CloudWatch Logs focuses on metrics, operational logs, and monitoring; it is not the primary service for immutable, account-level API activity auditing.

B ✗ — AWS Config — records resource configuration changes to evaluate compliance over time.

Rationale: AWS Config tracks configuration state changes of resources for compliance, not the full, immutable stream of API call events needed for account-level audit trails.

C ✗ — AWS Security Hub — aggregates security findings and compliance posture from multiple services.

Rationale: Security Hub centralizes security findings and standards checks; it does not itself provide the authoritative, immutable API call logs required for audit retention.

D ✓ — AWS CloudTrail — records and preserves account API activity and events as immutable logs.

Rationale: CloudTrail captures and stores AWS API calls and related events so they can be reviewed and retained for audits; it is intended for immutable activity logging.

Explanation

The requirement is for an authoritative, immutable record of API calls and account activity retained to satisfy audits for 365 days. AWS CloudTrail is the service purpose-built to capture, record, and store AWS API calls and related events across an account. It provides an audit trail of who did what, when, and from where; CloudTrail logs are the standard source for forensic review and compliance investigations.

Why the other services are not the best fit. The option describing CloudWatch Logs refers to collecting metrics and application or system logs for monitoring and troubleshooting. While CloudWatch Logs can store log data, it is primarily used for operational metrics and application logs rather than as the canonical immutable audit trail of AWS API activity. The option describing AWS Config focuses on recording resource configuration states and changes over time to assess compliance; it does not capture the full API call event stream as an audit trail. The option describing AWS Security Hub explains aggregation of security findings and compliance posture from multiple services; Security Hub consumes findings but is not the source of raw API call events for audits.

Key Concepts

- **API activity audit trail:** A chronological record of API calls and account actions used for forensic review and compliance verification. CloudTrail

captures these events.
- **Immutability for audits:** Audit trails should be tamper-evident and preserved so historical activity cannot be altered; CloudTrail provides a durable record suitable for audits when combined with secure storage.
- **Service purpose clarity:** Use the service whose primary design matches the requirement—CloudTrail for API-call logging; Config for resource configuration history; CloudWatch for operational metrics and logs; Security Hub for aggregated security findings.

Common Pitfalls

- Assuming CloudWatch Logs or Security Hub replace the audit-trail function of the API event service leads to gaps in forensic capability. Confusing configuration history (AWS Config) with API call auditing misses the distinct data each service provides.

Glossary

- **Audit trail:** A protected sequence of records that documents system activity and user actions for compliance and investigation.
- **Immutable logs:** Records stored in a way that prevents undetected modification, ensuring integrity for audits.

13.53 — Tags for cost allocation/chargeback (D4.T4.2)

Domain 4 • Task 4.2

↑ Back to Question 13.53

Which statement best defines an AWS cost allocation tag?

Options & Rationales

A ✓ — Key–value metadata on AWS resources that, when activated in Billing and Cost Management, become reporting dimensions in Cost Explorer, AWS Budgets, and the Cost and Usage Report (CUR).

Rationale: Cost allocation tags are standard resource tags. After activation, they appear as filterable dimensions in Cost Explorer, Budgets, and CUR to attribute and analyze spend.

B ✗ — An AWS Organizations policy type that enforces required tag keys and allowed values across accounts to standardize tagging.

Rationale: That describes AWS Organizations tag policies. They standardize tagging but do not turn tags into cost-reporting dimensions.

C ✗ — A Cost Management feature that groups multiple tag values and AWS accounts into logical business views for reporting and chargeback.

Rationale: This is AWS Cost Categories. They aggregate tags and accounts into business views; they are not the definition of a cost allocation tag.

D ✗ — A security control that automatically blocks creation of untagged EC2, RDS, and S3 resources at launch time.

Rationale: Tag enforcement can be achieved with templates and service control policies, but that is not what cost allocation tags are or do.

Explanation

Tags in AWS are key–value pairs applied to resources (for example, CostCenter=FIN, Project=MobileApp). When these tags are activated as cost allocation tags in the Billing and Cost Management console, they become dimensions you can use to group, filter, and attribute spend in AWS Cost Explorer, AWS Budgets, and the AWS Cost and Usage Report (CUR). Both AWS-generated tags (such as aws:createdBy) and user-defined tags can be activated. This enables showback or chargeback by mapping costs to business units, projects, or environments without building custom billing pipelines.

It is important to distinguish cost allocation tags from adjacent features. AWS Organizations tag policies help you standardize required tag keys and allowed values across accounts, improving data quality, but they do not make tags show up as cost dimensions. AWS Cost Categories provide higher-level business views by grouping multiple tag values and even multiple accounts into logical categories; they complement tags but are a different construct. Finally, enforcing that resources are tagged at creation is typically handled via deployment templates and Service Control Policies (SCPs), not by cost allocation tags themselves.

By activating appropriate tags and ensuring consistent application with governance, finance teams can filter Cost Explorer, configure tag-based AWS Budgets, and analyze CUR to achieve transparent chargeback. For shared resources that cannot be uniquely tagged per team, organizations often allocate costs proportionally using measured usage.

Key Concepts

- **Cost allocation tag:** A resource tag that, once activated, becomes a cost reporting dimension in Cost Explorer, Budgets, and CUR.
- **AWS Organizations tag policies:** Governance to standardize tag keys/values across accounts; they do not create cost dimensions.
- **AWS Cost Categories:** Logical groupings that aggregate tags and accounts into business views for reporting.
- **Chargeback/showback:** Mapping spend to owners using tag-based filters and reports.

Common Pitfalls

- Assuming tags appear in cost reports without activation or that tags themselves enforce tagging at resource launch. These require activation and separate governance controls.

13.54 — Data transfer pricing within/across Regions (D4.T4.1)

Domain 4 • Task 4.1

↑ Back to Question 13.54

A team sees sudden monthly network charges after moving a web API to another AWS Region. Which cause most likely explains the increased per-GB billable traffic?

Options & Rationales

A ✓ — Clients in one Region are calling an API deployed in a different Region, causing cross-Region data transfer charges.

Rationale: Data sent between different AWS Regions is billed at higher per-GB rates; calling an API across Regions generates inter-Region transfer costs.

B ✗ — Traffic between instances in the same Availability Zone is being billed per GB because private IP addresses were used.

Rationale: Inbound/outbound traffic inside the same AZ typically does not incur per-GB charges; using private IPs does not make intra-AZ links billable.

C ✗ — All data leaving AWS to the public internet is free when sourced from an EC2 instance in any Region.

Rationale: Egress to the public internet is billed; data leaving AWS is not free regardless of source service.

D ✗ — VPC peering automatically routes traffic over a dedicated Direct Connect link, avoiding transfer fees.

Rationale: VPC peering uses AWS network and does not use Direct Connect; it does not automatically eliminate transfer charges or use a dedicated link.

Explanation

Moving a service to a different geographical Region can change how network traffic is billed. When clients or other AWS resources located in one Region communicate with resources in another Region, AWS treats that data movement as cross-Region transfer and applies higher per-gigabyte charges than intra-Region traffic. This is why relocating the API without also moving clients or adjusting architecture often increases the network portion of the bill.

Why the correct choice fits. Cross-Region data movement is the primary driver of higher per-GB network costs because network hops between Regions traverse AWS backbone segments that are billed differently than traffic confined to a single Region.

Why the other choices are incorrect. Saying same-AZ traffic is billed because of private IP use is incorrect; traffic kept inside the same Availability Zone generally avoids per-GB transfer fees. Claiming data leaving AWS to the internet is free contradicts billing norms because internet egress is billed. Asserting that VPC peering uses a Direct Connect link is wrong; VPC peering uses AWS's internal network between VPCs and does not automatically route over a customer's Direct Connect or eliminate transfer charges.

Key Concepts

- **Cross-Region data transfer:** Charges apply when data moves between different AWS Regions and are typically higher than intra-Region charges.
- **Intra-AZ traffic:** Communication that stays inside the same Availability Zone is usually not billed per gigabyte.

Common Pitfalls

Assuming that moving only the backend reduces all network costs can be misleading; client location relative to resources determines whether traffic becomes cross-Region. Another common mistake is confusing VPC peering with private customer connections like Direct Connect.

13.55 — When to require MFA (D2.T2.3)

Domain 2 • Task 2.3 ↑ Back to Question 13.55

Select TWO AWS-native interactive sign-in activities from the list where adding multi-factor authentication (MFA) is most appropriate to reduce the risk of credential compromise and protect sensitive AWS access.

Options & Rationales

A ✓ — Signing in as the account root user to perform billing or account-level changes

Rationale: Root has full, persistent privileges; MFA prevents unauthorized full-account control if the password is compromised.

B ✗ — Using long-lived programmatic access keys for automated scripts and services

Rationale: Programmatic keys are non-interactive; protecting them focuses on rotation and least privilege rather than human MFA at sign-in.

C ✓ — Interactive sign-in by an IAM user who manages account-wide administrative policies

Rationale: Privileged administrative users can change policies and access; MFA reduces risk from stolen credentials.

D ✗ — Accepting a federated login session from an external identity provider for a short-lived user

Rationale: Federated sessions inherit the external provider's authentication; MFA may be applied there, but the AWS-side session itself is not the primary MFA point.

E ✗ — Assuming a cross-account role for a brief, limited-privilege task

Rationale: Cross-account role assumption uses temporary credentials; MFA is recommended for sensitive escalation but not required for every short-lived role.

F ✗ — Temporarily rotating a user password or updating non-sensitive profile settings

Rationale: Routine, low-risk profile updates typically do not require MFA; high-risk credential changes do, but non-sensitive edits do not.

Explanation

Multi-factor authentication (MFA) requires a second verification factor beyond a password, which significantly reduces the chance an attacker can use stolen credentials to gain access. Protecting the highest-impact sign-ins is the priority: the account root user controls the entire AWS account and cannot be limited by IAM policies, so adding MFA for root prevents complete account takeover. Interactive IAM users with administrative or policy-management privileges can make broad configuration changes; requiring MFA for those sign-ins reduces risk from credential compromise.

Why the other activities are less appropriate as primary MFA targets.
Using long-lived programmatic keys for automation requires secure handling, rotation, and least-privilege access rather than human MFA at sign-in because these credentials are non-interactive. Routine, non-sensitive profile edits or brief password rotations are lower risk compared to full administrative actions; MFA is most valuable for high-impact credential changes rather than every trivial setting. Federated logins depend on the external identity provider to enforce MFA, so the AWS session reflects that provider's assurance rather than an additional AWS-side MFA step. Temporary cross-account role assumption uses short-lived credentials; while MFA can be part of a sensitive escalation flow, it is not the default control for every short-lived role.

Key Concepts

- **Multi-Factor Authentication (MFA):** Adds a second factor (something you have or are) in addition to a password to strengthen authentication and reduce unauthorized access.
- **Account Root vs. IAM Administrative Users:** The root account has unrestricted control and should have the strongest protections; privileged IAM users can alter policies and resources and should also use enhanced protections like MFA.
- **Programmatic Credentials vs. Interactive Sign-in:** Programmatic access uses keys for services and automation and is protected by rotation and least privilege; MFA targets human interactive logins.

Common Pitfalls

- Treating every action as equally risky: MFA is most effective when applied to high-impact, interactive sign-ins rather than all automated keys or low-risk profile edits.
- Assuming federated sessions automatically provide AWS-side MFA: the external identity provider must enforce MFA for federated sessions to be effective.

13.56 — When to use ElastiCache (D3.T3.4)

Domain 3 • Task 3.4 ↑ Back to Question 13.56

Select TWO options that are NOT appropriate uses of an in-memory caching service like Amazon ElastiCache.

Options & Rationales

A ✗ — Cache frequent database read results to reduce load on the primary database and lower read latency.

Rationale: This matches a common, correct pattern: caching read-heavy query results improves response times and offloads the database.

B ✗ — Store ephemeral user session state for a web application to speed access.

Rationale: This is an appropriate use because in-memory caches provide fast, temporary storage for session data and reduce backend database reads.

C ✓ — Keep long-term archival records (compliance retention) as the primary copy of data.

Rationale: Not appropriate: using an in-memory cache as the long-term archive violates durability requirements because caches are ephemeral and not designed for permanent, compliant storage.

D ✓ — Use the cache as the authoritative, durable store for transactional data requiring strong consistency.

Rationale: Not appropriate: relying on an in-memory cache as the primary durable store is an anti-pattern since caches do not guarantee durable, strongly consistent storage for transactions.

E ✗ — Offload many concurrent, low-latency read requests from a busy database to improve throughput.

Rationale: This is an appropriate use: in-memory caching supports high-throughput, low-latency read patterns and reduces primary database contention.

Explanation

An in-memory caching service is intended to provide fast, temporary access to frequently read data and to reduce load on a primary database. It is managed for speed and scale, not for long-term durability or authoritative persistence. Choosing a cache is appropriate when the system can tolerate ephemeral storage and eventual refreshes from the primary store.

Why the incorrect uses are violations. Using a cache as the long-term archive violates durability and retention requirements because cached entries can be evicted or lost. Using a cache as the authoritative transactional store is an anti-pattern because in-memory caches do not guarantee the durable, strongly consistent behavior expected of a primary database.

Key Concepts

- **Ephemeral, low-latency storage:** Caches store data in memory for quick reads but are not built to preserve data permanently across failures.
- **Offloading reads:** Caching reduces read IOPS and contention on databases by serving repeated queries from memory.
- **Durability vs. performance trade-off:** Durable, compliance-grade archival storage requires services designed for long-term retention and data integrity, which is different from in-memory caching.

Common Pitfalls

- A frequent mistake is treating cached data as the only copy; failure to maintain a durable primary source risks data loss. Another is assuming caches provide the same consistency and retention guarantees as databases.

Glossary

- **In-memory cache:** A fast data store kept in RAM for quick reads, typically used for temporary or replicated data.
- **Durability:** The assurance that data will survive restarts, failures, and meet retention/compliance needs.

13.57 — AWS Detective analysis/visualization (D2.T2.4)

Domain 2 • Task 2.4 ↑ Back to Question 13.57

A security analyst needs a quick visual map that shows how AWS resources and identities interacted during a suspicious incident to help find the root cause. Which Detective capability best provides an interactive relationship map of entities and their activities?

Options & Rationales

A ✓ — Behavior graphs — interactive relationship maps that link resources, principals, and activities for investigation.

Rationale: Behavior graphs create visual links between entities and actions so investigators can trace relationships and root cause efficiently.

B ✗ — CloudWatch metrics dashboards showing aggregated performance and utilization over time.

Rationale: CloudWatch provides metrics and dashboards, which show performance trends but do not produce relationship maps between entities and activities.

C ✗ — CloudTrail event logs that list API calls for auditing and compliance purposes.

Rationale: CloudTrail records API activity as logs for auditing, but it does not automatically generate interactive visual relationship maps for investigations.

D ✗ — GuardDuty alerts that prioritize suspicious findings detected across the account.

Rationale: GuardDuty detects and prioritizes threats but does not itself produce an interactive relationship map linking entities and their actions.

Explanation

The investigator requested an interactive visual map that links resources, identities, and actions to trace a security incident's root cause. A behavior-graph capability provides relationship mapping: it models entities (for example, instances, IP addresses, or principals) and the activities that connect them and displays those links visually so investigators can follow chains of activity.

Why this is correct. Behavior graphs aggregate linked events into a graph of entities and interactions, making it straightforward to see which resources communicated, which principals performed actions, and how events relate over time. That visual structure speeds root-cause analysis because investigators can focus on connected nodes rather than parsing long lists of unrelated records.

Why the other choices are not the best fit. CloudWatch dashboards show metrics and trends (CPU, latency, error rates), useful for performance and operational monitoring, but they do not create relational maps of entities and actions. CloudTrail provides detailed API call records for auditing and forensic timelines, which are essential raw data, but it does not present the information as an interactive relationship graph by itself. GuardDuty generates prioritized security findings to surface potential threats, helping triage, but those findings are alerts rather than a navigable map of linked entities and activities.

Key Concepts

- **Behavior graph:** A visual representation that models entities (resources and principals) as nodes and their interactions as edges, enabling investigators to trace relationships and paths during analysis.
- **Relationship-driven investigation:** Focusing on how entities interact helps identify lateral movement, compromised principals, or suspicious communication patterns more quickly than inspecting isolated logs.

Common Pitfalls

- Treating raw logs as a visual map: Assuming that reading CloudTrail or alerts alone provides immediate relationship context can slow investigations; raw records often need correlation or visualization.
- Confusing detection with visualization: Detection (alerts) and visualization (relationship graphs) serve different roles; one surfaces suspicious activity, the other helps explore how that activity propagated.

Glossary

- **Entity:** A resource or principal (for example, an EC2 instance, IP address, or IAM user) involved in activity.
- **Relationship map:** A graphical view showing connections and interactions between entities to reveal paths and dependencies.

13.58 — Cross-account access with roles & external ID (D2.T2.3)

Domain 2 • Task 2.3 ↑ Back to Question 13.58

Which purpose best describes an External ID when granting another AWS account temporary access to your resources via an IAM role?

Options & Rationales

A ✗ — A long-lived secret key stored in the role that authenticates the requester for ongoing access.

Rationale: External IDs are not long-lived secrets stored in roles; they are single values used during the temporary assume-role exchange and do not replace secrets or credentials.

B ✓ — A value supplied by the role requester that helps prevent the confused-deputy problem during cross-account role assumption.

Rationale: An External ID is provided by the caller (requester) and checked by the role's trust policy to ensure the request originates from the intended third party, mitigating confused-deputy attacks.

C ✗ — A tag added to resources that automatically grants permissions to a specific external account.

Rationale: Resource tags do not grant cross-account permissions by themselves; permission delegation uses role trust and assume-role checks rather than tagging to authorize another account.

D ✗ — A policy condition that encrypts temporary credentials returned by STS for the requester.

Rationale: External ID is not an encryption mechanism for STS credentials; it is a value evaluated during AssumeRole to validate the request context, while encryption is handled by other services like KMS.

Explanation

When a resource owner allows another AWS account to assume an IAM role, a small, caller-provided value called an External ID can be required as part of the assume action. The External ID is included in the role's trust policy as a condition; the security token service verifies that the incoming assume request carries the expected value before issuing temporary credentials. This pattern prevents a third party from unintentionally using its permissions on behalf of an unrelated caller — a scenario known as the confused-deputy problem.

Why the correct choice fits. The correct description states that the External ID is supplied by the requester and helps prevent the confused-deputy problem during cross-account role assumption. That captures the purpose: it is a validating token checked during the short-lived credential issuance flow, not a secret stored in the role or an encryption control.

Why the other choices are wrong. The idea that the External ID is a long-lived secret stored in the role is incorrect because External IDs are transient validation values used at assume time, not persistent authentication credentials. Saying it is a resource tag that grants permissions is incorrect because tags do not grant cross-account access; trust policies and assume-role interactions govern delegation. Finally, describing the External ID as a policy condition that encrypts STS credentials is wrong because it does not perform encryption; encryption of credentials uses cryptographic services like KMS, whereas the External ID is only a conditional check.

Key Concepts

- **External ID purpose:** A caller-supplied value validated during role assumption to confirm the request originates from the intended third party and to reduce confused-deputy risks.
- **Trust policy condition:** The role's trust policy can require the External ID as a condition when allowing a principal to call AssumeRole, enforcing requester context checks before temporary credentials are issued.
- **Temporary credentials via STS:** AWS STS issues short-lived credentials when an AssumeRole request passes trust policy checks; the External ID is evaluated during this process, not used as a stored secret.

Common Pitfalls

- Treating the External ID as a password or long-lived secret leads to misuse; it is only a validation value checked at role assumption time.
- Assuming tags or resource metadata alone can implement cross-account authorization is incorrect; delegation requires proper trust policy and STS-based credential issuance.

13.59 — Migration phases: assess, mobilize, M&M (D1.T1.3)

Domain 1 • Task 1.3

↑ Back to Question 13.59

An enterprise rushed from assess to rehosting and skipped mobilize. After a few migration waves, they now have many AWS accounts with inconsistent IAM, ad hoc VPC designs, and logs scattered across accounts. Leadership fears rework and security gaps will erode migration ROI. What is the most effective next step to optimize outcomes before scaling migration?

Options & Rationales

A ✗ — Consolidate all workloads into a single AWS account and apply a centralized IAM policy to reduce complexity.

Rationale: Contradicts the multi-account landing zone goal. Centralizing in one account reduces isolation and does not fix guardrails, networking standards, or logging at scale.

B ✗ — Proceed with rehosting using AWS Application Migration Service now and defer IAM, VPC, and logging standardization until after cutover.

Rationale: Rehosting first amplifies sprawl and rework. MGN does not solve governance issues; skipping mobilize is the pitfall that erodes ROI.

C ✗ — Purchase Compute Savings Plans immediately to improve costs while keeping current account structures unchanged.

Rationale: Improves compute spend but leaves governance and security risks unresolved. ROI can still erode due to rework and inconsistent controls.

D ✓ — Establish an AWS Control Tower landing zone with AWS Organizations, IAM Identity Center, SCP guardrails, centralized CloudTrail/CloudWatch logging, and KMS encryption to standardize security and access across accounts.

Rationale: Implements the mobilize phase: secure multi-account foundations, least-privilege access, guardrails, and centralized logging. Directly addresses sprawl and security gaps before scaling.

Explanation

This scenario highlights the consequences of skipping the mobilize phase of the migration journey. The migration journey consists of assess, mobilize, and migrate-and-modernize. Mobilize turns intent into an executable plan and foundation. A key outcome of mobilize is a secure multi-account landing zone that standardizes access, guardrails, logging, encryption, and network design before scaling migration waves. The recommended way to establish this foundation is using AWS Control Tower and AWS Organizations, with IAM Identity Center for access, service control policies (SCPs) for preventive guardrails, centralized logging via AWS CloudTrail and Amazon CloudWatch, and encryption managed by AWS Key Management Service (AWS KMS). Skipping mobilize often causes account sprawl, inconsistent security, and rework—exactly what the scenario describes—and those issues threaten the return on investment (ROI).

ROI can be framed as ROI $= (B - C)/C$, where increasing rework and security incidents increase costs C, thereby reducing ROI. To protect ROI and enable scale, the most effective next step is to complete mobilize outcomes: stand up the Control Tower landing zone, enforce SCP guardrails, unify workforce access with Identity Center, centralize logs, and align VPC, tagging, and cost allocation designs. With this foundation in place, subsequent rehosting or modernization efforts proceed consistently and with lower operational risk.

Variables used:

- B: Expected benefits [USD]
- C: Migration and run costs [USD]

Why the other ideas fall short:

- Consolidating into a single account reduces isolation and still leaves governance, tagging, and logging gaps unresolved. It conflicts with the objective of a secure multi-account landing zone.

- Continuing to rehost with AWS Application Migration Service (MGN) before fixing access, guardrails, and logging just spreads inconsistent patterns, creating more rework—the very pitfall that erodes ROI.
- Buying Savings Plans is useful for cost optimization later, but it does not address the root causes (governance and security). Costs might drop, yet risk and rework remain.

Key Concepts

- **Mobilize phase and landing zone:** Use AWS Control Tower and AWS Organizations to standardize accounts, access (IAM Identity Center), guardrails (SCPs), logging (CloudTrail/CloudWatch), and encryption (KMS).
- **Account sprawl pitfall:** Skipping mobilize causes inconsistent security and rework that undermines migration velocity and outcomes.
- **ROI sensitivity:** ROI = $(B-C)/C$; governance gaps increase C through rework and incidents, lowering ROI.

Common Pitfalls

- Optimizing spend first (e.g., Savings Plans) without fixing governance; this lowers unit costs but leaves risks and rework that continue to erode ROI.

13.60 — S3 durability & availability guarantees (D3.T3.6)

Domain 3 • Task 3.6

↑ Back to Question 13.60

Which durability level does Amazon S3 promise for objects stored across multiple Availability Zones?

Options & Rationales

A ✗ — 99.99% durability

Rationale: 99.99% is commonly used for availability SLAs, not S3's long-term object durability guarantee, so this underestimates S3's durability claim.

B ✗ — Six nines (99.9999%) durability

Rationale: Six nines is a high durability level but still lower than S3's documented eleven nines design target; it understates S3's durability.

C ✗ — No durability guarantee—customers must replicate data themselves

Rationale: While customers can add replication for additional resilience, S3 itself provides a documented high durability design rather than leaving durability entirely to the customer.

D ✓ — 11 nines (99.999999999%) durability

Rationale: Amazon S3 is designed to provide extremely high object durability, commonly described as eleven nines (99.999999999%), meaning a very low likelihood of permanent data loss when stored across AZs.

Explanation

Amazon S3 is engineered to keep stored objects safe over long periods by redundantly storing data across multiple Availability Zones within a Region. The service's durability metric expresses the probability that data will persist without loss. S3's commonly cited durability target is eleven nines (99.999999999%), indicating an extremely low likelihood of object loss when using standard S3 storage with its multi-AZ architecture.

Why the correct answer is right. The eleven-nines figure refers to the design goal for object persistence in S3 and reflects redundant storage and internal checks that protect against hardware failures and data degradation. This is a durability statement (data won't be lost), not an availability SLA about access times.

Why the other choices are wrong. Stating 99.99% confuses availability SLAs with durability and therefore understates S3's durability design. Six nines is higher than typical service-level references but still below S3's eleven-nines durability target, so it is incorrect. Saying there is no durability guarantee ignores that S3's architecture and documented durability target already provide strong protection; replication options add geographic resilience but do not replace the base durability promise.

Key Concepts

- **Durability:** Measures the likelihood that stored data will remain intact over time; higher durability means lower chance of permanent data loss.
- **Redundancy across AZs:** Storing multiple copies in separate Availability Zones reduces the risk that a single hardware or facility failure causes data loss.

Common Pitfalls

- Confusing durability with availability leads to selecting an availability percentage as the durability value.
- Assuming replication is required for any durability—replication improves geographic resilience but S3 already provides high durability within a Region.

Glossary

- **Availability:** The proportion of time a service is reachable and able to respond to requests.
- **Availability Zone (AZ):** An isolated location within an AWS Region used to separate redundant resources.

13.61 — Infra vs managed service responsibilities (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 13.61

Which statement best describes the operational responsibility difference between running a database on Amazon EC2 versus using Amazon RDS?

Options & Rationales

A ✗ — Both EC2 and RDS require the customer to install and patch the underlying host operating system manually.

Rationale: This is incorrect because AWS manages the OS and patches for managed RDS instances, while EC2 requires customer management of the guest OS.

B ✓ — On EC2 you manage the operating system and database software; with RDS AWS handles the database engine and OS patching for you.

Rationale: EC2 provides virtual servers where the customer is responsible for the guest OS and installed applications; Amazon RDS is a managed database service where AWS performs engine management and operating system patching.

C ✗ — RDS gives full root access to the database host so customers can patch the OS, while EC2 does not allow OS changes.

Rationale: This is wrong: RDS does not provide full root access to the database host; EC2 does allow customers full control of the guest OS.

D ✗ — EC2 automatically updates database engines, whereas RDS requires customers to update engine versions and operating system packages.

Rationale: This reverses responsibilities: RDS automates engine updates and OS patching options, while EC2 leaves engine and OS updates to the customer.

Explanation

At a high level, running a database on a virtual server versus a managed database service changes who is responsible for operating system and database maintenance. On a virtual server, the customer provisions the instance, installs the OS and database software, and must apply patches and updates. A managed database service delegates engine lifecycle tasks and host-level patching to the cloud provider, reducing the customer's operational burden.

Why the correct choice is right. The correct statement says the customer manages the operating system and database software on EC2, while AWS handles the database engine and operating system patching for RDS. This reflects that EC2 provides raw compute where guest-level responsibilities (OS, runtime, applications) belong to the customer. In contrast, RDS is a managed service where AWS performs routine maintenance of the engine and underlying host, including OS patching, so customers focus on data and schema instead of host maintenance.

Why the other choices are wrong. One option claims both services require customers to patch the OS; that ignores RDS managed host maintenance. Another claims RDS grants full root access to patch the OS; managed RDS restricts host-level root access. The last option flips responsibilities by saying EC2 auto-updates engines and RDS requires customers to update—this is the opposite of how managed services and virtual servers operate.

Key Concepts

- **Virtual server responsibility:** Customers control the guest OS, installed software, and patching when using virtual machines such as EC2.
- **Managed service responsibility:** The provider performs host-level maintenance (including operating system patching) and engine management for services like Amazon RDS.

Common Pitfalls

- Confusing who controls the host vs. the service; assuming managed services expose host-level root access; reversing which option performs automated maintenance.

Glossary

- **Guest OS:** The operating system installed and managed by the customer on a virtual machine.
- **Managed database service:** A cloud offering where the provider handles infrastructure and routine maintenance tasks for the database engine.

13.62 — Security Hub centralizes findings (D2.T2.2)

Domain 2 • Task 2.2 ↑ Back to Question 13.62

A company has 30 AWS accounts in AWS Organizations. All workloads run only in us-east-1. Security leadership wants: 1) a single, centralized view of security findings across all accounts, and 2) automatic isolation of EC2 instances when high-severity findings are generated by integrated services like Amazon GuardDuty or Amazon Inspector. The team prefers native AWS integrations. Which TWO actions should the team take?

Options & Rationales

A ✓ — Designate a delegated administrator for AWS Security Hub in AWS Organizations to centrally manage and view findings across member accounts.

Rationale: Security Hub supports governance at scale via a delegated administrator, enabling centralized visibility and management across multiple member accounts.

B ✗ — Enable Amazon Macie automatic quarantine in Security Hub to remove sensitive S3 data exposures without additional configuration.

Rationale: Security Hub provides visibility and governance; it does not automatically remediate. Macie identifies sensitive data, while remediation requires automation (for example, via EventBridge to runbooks or Lambda).

C ✗ — Set a global aggregation Region in AWS Security Hub so you do not need to enable Security Hub in other Regions.

Rationale: Security Hub is Regional. An aggregation Region helps consolidate multi-Region views, but it does not replace per-Region enablement and is unnecessary in a single-Region setup.

D ✗ — Configure AWS Config conformance packs so failed Security Hub controls are resolved automatically by the service.

Rationale: Conformance packs and rules support evaluation, while Security Hub surfaces failed controls and a compliance score. Automated fixes require EventBridge-driven actions, not Security Hub or conformance packs alone.

E ✓ — Create an Amazon EventBridge rule that filters high-severity Security Hub findings and invokes an AWS Systems Manager Automation runbook to isolate affected EC2 instances.

Rationale: Security Hub findings can be routed with EventBridge to trigger automated remediation, such as Systems Manager Automation runbooks that isolate EC2 instances.

F ✗ — Use AWS CloudTrail event rules to route Security Hub findings directly to remediation workflows.

Rationale: The native integration for routing Security Hub findings is Amazon EventBridge. EventBridge filters findings and triggers automation; CloudTrail is not used for this purpose here.

Explanation

AWS Security Hub centralizes security visibility across AWS accounts and Regions by aggregating findings from native services (for example, Amazon GuardDuty, Amazon Inspector, Amazon Macie, IAM Access Analyzer, AWS Config) into a normalized format. It also evaluates environments against built-in security standards and produces a compliance score that highlights failed controls for remediation. In the shared responsibility model, Security Hub helps with visibility and governance, while customers are responsible for enabling it, interpreting results, and remediating issues.

For multi-account governance at scale, Security Hub integrates with AWS Organizations. By designating a delegated administrator, security teams can centrally manage Security Hub across member accounts. Security Hub is Regional; for multi-Region consolidation, you can set an aggregation Region. Findings can be de-duplicated, enriched, and surfaced via dashboards and Insights. To reduce mean time to remediation, Security Hub integrates with Amazon EventBridge, which can route specific findings to automation such as AWS Systems Manager Automation runbooks or AWS Lambda.

Given the scenario (all workloads in us-east-1 and a need for cross-account visibility and automated isolation of EC2 instances), the combination that meets both requirements is: use a delegated administrator to centrally manage and view findings across all accounts, and create an EventBridge rule that filters high-severity findings and invokes a Systems Manager Automation runbook to isolate the affected instances. This uses native integrations for both governance and remediation.

Other proposals do not meet the stated needs. Setting an aggregation Region is aimed at multi-Region consolidation; in a single-Region environment it

does not add value, and it does not remove the Regional nature of the service. Claims of automatic quarantine or auto-resolution (for example, via Macie or conformance packs alone) conflict with Security Hub's governance-focused role; remediation is customer-driven and typically automated via EventBridge to Systems Manager or Lambda. Using CloudTrail to route findings is not the described mechanism; EventBridge is the native event routing for Security Hub findings.

Key Concepts

- **Security Hub delegated administrator:** Centralizes enablement and management across AWS Organizations member accounts for consistent governance.
- **Regional scope and aggregation:** Security Hub is Regional; an aggregation Region provides a consolidated view across Regions when needed.
- **EventBridge remediation:** EventBridge filters Security Hub findings and triggers automation (Systems Manager Automation or Lambda) for remediation.
- **Shared responsibility:** Security Hub improves visibility; customers must enable it, interpret results, and implement fixes.

Common Pitfalls

- Assuming Security Hub or Macie automatically remediates findings without automation. Security Hub surfaces findings; remediation must be orchestrated (for example, via EventBridge).
- Configuring multi-Region aggregation when operating in a single Region, adding complexity without meeting additional requirements.

13.63 — Six Well-Architected pillars & focus (D1.T1.2)

Domain 1 • Task 1.2 ↑ Back to Question 13.63

A retailer serves its static website directly from a single Amazon S3 bucket in one Region. Global users report high latency. The company also wants to lower S3 origin load to reduce cost and prefers minimal changes. Which is the BEST solution?

Options & Rationales

A ✗ — Enable S3 Intelligent-Tiering on the bucket to auto-move objects between access tiers for better latency and lower transfer costs.

Rationale: Intelligent-Tiering optimizes storage cost based on access patterns. It does not provide edge caching or lower request/transfer latency.

B ✓ — Put Amazon CloudFront in front of the S3 bucket to cache objects at edge locations, cutting latency and reducing direct S3 requests.

Rationale: CloudFront edge caching improves global performance and reduces origin load/cost for S3-backed static sites with minimal change.

C ✗ — Serve files through Amazon API Gateway backed by AWS Lambda for automatic scaling without servers.

Rationale: API Gateway + Lambda is ideal for business logic, not static asset delivery; it adds overhead and does not cache files at the edge.

D ✗ — Use AWS WAF to filter requests so fewer reach S3, improving latency for global users.

Rationale: AWS WAF provides security controls. It does not accelerate delivery or cache content; performance and origin offload require a CDN like CloudFront.

Explanation

To improve global performance and reduce origin load for static content hosted on Amazon S3, the most effective pattern is to place a content delivery network (CDN) in front of the origin. Amazon CloudFront integrates with S3 and caches objects at edge locations. Serving objects from those edges lowers round-trip time for users worldwide (Performance Efficiency) and reduces the number of requests that reach S3 (Cost Optimization). Fewer origin fetches also reduce S3 request and data transfer from the origin. Creating a CloudFront distribution with S3 as the origin is a minimal-change solution aligned with the Well-Architected principles described.

Enabling S3 Intelligent-Tiering is a storage class feature that automatically moves objects between access tiers to optimize storage cost as access patterns change. It does not provide edge caching or improve network latency. Using Amazon API Gateway with AWS Lambda is excellent for serverless business logic because it scales automatically and removes server management, but it introduces additional hops and compute cost for static asset delivery and does not cache content at the edge. AWS WAF is a security control deployed at the edge to help block malicious requests; it does not accelerate delivery or offload the origin for general traffic.

Key Concepts

- **Amazon CloudFront with S3 origin**: Edge caching lowers latency and reduces S3 origin requests, improving both performance and cost.
- **S3 Intelligent-Tiering:** Optimizes storage cost only; it does not affect delivery latency or provide caching.
- **API Gateway + Lambda:** Suited for serverless business logic, not for static file distribution and caching.
- **AWS WAF at the edge:** Security and request filtering; not a performance accelerator or cache.

Common Pitfalls

- Assuming S3 Intelligent-Tiering improves latency or reduces network transfer by itself.
- Using compute or security services to solve a content delivery performance

problem instead of a CDN.

13.64 — Forecasting & controlling spend (multi-account) (D4.T4.2)

Domain 4 • Task 4.2 ↑ Back to Question 13.64

Select TWO actions that directly use budget-based, threshold-driven forecasting to help control spend across multiple AWS accounts:

Options & Rationales

A ✓ — Create budgets that send alerts when forecasted month-end spend exceeds a chosen percentage of budgeted amount.

Rationale: Budgets can compare forecasted spend to a budgeted target and notify stakeholders when projected spend passes a defined percentage threshold.

B ✓ — Configure automated responses tied to budget thresholds to trigger operational controls when forecasts predict overspend.

Rationale: Budgets support actions or notifications that enable automated steps (for example, workflows or approvals) when a forecasted or actual threshold is reached, aligning forecasts with operational limits.

C ✗ — Use Trusted Advisor checks exclusively to forecast next month's spend for all accounts.

Rationale: Trusted Advisor provides optimization and best-practice checks but is not the primary tool for budgets-based forecasting and threshold alerts.

D ✗ — Apply chargeback tags to resources and then wait for end-of-month cost allocation reports to allocate spend.

Rationale: Tagging supports cost allocation and reporting but does not itself provide threshold-driven forecast alerts to control spend proactively.

E ✗ — Rely only on the monthly consolidated invoice to detect unexpected account-level overruns after the billing cycle closes.

Rationale: Monthly invoices report charges after they accrue and do not provide threshold-driven, forecast alerts that enable proactive control during the cycle.

F ✗ — Depend on manual review of CloudWatch metrics as the only mechanism to forecast account billing.

Rationale: CloudWatch monitors resource and application metrics but is not designed to provide billing forecasts or budget threshold alerts without integration to budgeting tools.

Explanation

Budget-driven forecasting and threshold alerts let organizations detect and act on projected overspend before charges finalize. A budget compares actual and

forecasted costs to a planned target and can notify people or kick off automated responses when a percentage threshold is crossed, enabling proactive cost control across accounts. Relying solely on post-cycle invoices or ad hoc metric reviews delays action; similarly, tagging and optimization checks are useful for reporting and recommendations but do not replace a forecasting-and-alert mechanism.

Key Concepts

- **Cost Forecasting with Budgets:** Budgets can project month-end spend based on current consumption and compare it against a budgeted amount to produce forecasted alerts.
- **Threshold-Driven Alerts and Actions:** Budgets support threshold definitions (for example, 80% of budget) which can trigger notifications or automated operational responses to constrain spending.
- **Proactive vs. Reactive Controls:** Forecast alerts are proactive—providing lead time to remediate—whereas invoices and end-of-period reports are reactive.

Common Pitfalls

- Treating tagging or invoice review as a substitute for forecast alerts leads to late detection of overruns. Confusing monitoring metrics with billing forecasts can miss cost drivers that budgeting tools identify.

Glossary

- **Budget:** A defined spending target that can monitor actual and forecasted costs.
- **Forecasted spend:** A projection of expected charges at the end of the billing period based on current usage trends.

13.65 — Responsibility shifts for serverless (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 13.65

Under the shared responsibility model for AWS Lambda, which task is the customer's responsibility?

Options & Rationales

A ✓ — Scan and remediate vulnerabilities in the AWS Lambda deployment package or container image and keep dependencies updated.

Rationale: Customers must secure their code and supply chain by fixing package/image vulnerabilities and updating third-party libraries.

B ✗ — Patch the underlying operating system and managed runtimes for AWS Lambda functions.

Rationale: AWS patches the OS and managed runtimes for Lambda; customers do not manage host-level patching.

C ✗ — Provide multi-AZ availability and automatic scaling for the Lambda compute fleet.

Rationale: AWS provides multi-AZ availability and handles capacity and scaling for Lambda.

D ✗ — Create and manage the elastic network interfaces (ENIs) when a Lambda function is attached to a VPC.

Rationale: When a function uses a VPC, AWS creates the ENIs; customers design subnets, route tables, and security groups.

Explanation

In serverless architectures using AWS Lambda, the shared responsibility model shifts more infrastructure duties to AWS but does not remove customer obligations. AWS operates the compute fleet, provides multi-AZ availability, patches the underlying operating system and managed runtimes, and handles capacity and scaling. AWS also isolates execution environments and, when a function connects to a VPC, creates the elastic network interfaces (ENIs).

Customers remain responsible for securing their application logic and data paths. This includes scanning and remediating vulnerabilities in the function's deployment package or container image and keeping third-party dependencies current. Customers also define identity and access (execution role and resource-based policies), design VPC components when used (subnets, route tables, security groups, and any NAT gateways or VPC endpoints), enforce encryption in transit and at rest (often with customer managed AWS KMS keys and appropriate key policies/rotation), and configure monitoring, log retention, and tracing.

Therefore, the customer-owned task is remediating vulnerabilities in the deployment package or container image and keeping dependencies updated. Patching the OS/managed runtimes, ensuring multi-AZ and automatic scaling, and creating ENIs for VPC-attached functions are handled by AWS.

Key Concepts

- **Shared responsibility in serverless:** AWS runs and secures the platform; customers secure code, access, and data.
- **Lambda patching split:** AWS patches the OS and managed runtimes; customers patch dependencies and remediate code/image vulnerabilities.
- **Networking boundary:** AWS creates ENIs for VPC-attached functions; customers design subnets, route tables, and security groups.
- **Availability and scaling:** Multi-AZ availability and capacity/scaling for Lambda are provided by AWS.

Common Pitfalls

- Assuming "serverless" means customers have no patching duties, leading to unpatched dependency vulnerabilities.
- Confusing ENI creation (AWS) with broader VPC design and access controls (customer).

Exam 14

Question 14.1

Which of the following statements about granting user permissions in AWS is NOT correct?

○ **A.** Assign roles or policies that allow only the actions a user needs to perform their job.

○ **B.** Use temporary credentials or role assumption for tasks that need elevated permissions for a limited time.

○ **C.** Regularly review and remove unused permissions or accounts to reduce unintended access.

○ **D.** Give administrators broad, permanent full-access policies to simplify troubleshooting and avoid permission errors.

Explanation→

Question 14.2

An organization runs application servers in private subnets across two Availability Zones in an IPv4-only VPC. The servers must download OS patches and packages from various internet repositories, but must not be reachable from the internet. An Internet Gateway is already attached to the VPC, and a web tier in a public subnet has internet access.

Which actions will provide outbound-only internet access for the private subnets while keeping inbound blocked?

☐ **A.** Add a 0.0.0.0/0 route in the private subnets' route table that targets the Internet Gateway.

☐ **B.** Add a 0.0.0.0/0 route in the private subnets' route table that targets the NAT Gateway.

☐ **C.** Attach an Egress-Only Internet Gateway and add ::/0 in the private subnets' route table.

☐ **D.** Enable auto-assign public IPv4 addresses on the private subnets.

☐ **E.** Create a NAT Gateway in a public subnet.

☐ **F.** Create a VPC endpoint for Amazon S3.

Explanation→

Question 14.3

A security engineer needs short-lived credentials for an IAM user who must use multi-factor authentication (MFA) before accessing AWS APIs. Which TWO statements correctly describe how AWS Security Token Service (STS) GetSessionToken supports this requirement?

☐ **A.** GetSessionToken creates a new IAM role which the user assumes to gain temporary permissions.

☐ **B.** GetSessionToken returns temporary access keys and a session token that expire after a configurable short duration.

☐ **C.** GetSessionToken returns permanent long-term credentials usable until explicitly revoked.

☐ **D.** GetSessionToken is intended for identity federation with external SAML or OIDC providers.

☐ **E.** GetSessionToken modifies an IAM user's inline policies to restrict access during the session.

☐ **F.** When an MFA serial number and one-time code are provided, GetSessionToken can require MFA before issuing temporary credentials.

Explanation→

Question 14.4

A retailer runs an order-processing microservice that stores data in Amazon DynamoDB. During flash sales, customers place many orders concurrently. For each order, the application must create an order record and decrement inventory items. The company must guarantee that either all updates are applied together or none are applied to avoid overselling. Which is the BEST solution?

○ **A.** Use DynamoDB Streams to detect oversells and trigger an asynchronous workflow to reverse conflicting updates.

○ **B.** Use DynamoDB ACID transactions to group the order write and inventory decrements into a single all-or-nothing operation.

○ **C.** Add DynamoDB Accelerator (DAX) to cache inventory reads for microsecond latency before writing updates.

○ **D.** Create a Global Secondary Index (GSI) on sku to query inventory updates quickly across items and maintain consistency.

Explanation→

Question 14.5

A company serves customer PII from an Amazon S3 bucket through Amazon CloudFront. A security review requires encryption in transit end to end (viewer to CloudFront and CloudFront to S3). Review the abbreviated configuration and choose the single change that best meets the requirement with minimal architecture changes.

Table 2: Exhibit: CloudFront distribution (abbreviated)

Setting	Current value
Viewer protocol policy	Allow HTTP and HTTPS
Origin	S3 bucket (my-bucket)
Origin protocol policy	HTTP Only
S3 encryption at rest	SSE-KMS: Enabled

○ **A.** Update CloudFront to enforce HTTPS: set Viewer Protocol Policy to Redirect HTTP to HTTPS and Origin Protocol Policy to HTTPS Only.

○ **B.** Enable S3 default encryption with SSE-KMS and rely on CloudTrail logs for key usage.

○ **C.** Create a VPC interface endpoint (AWS PrivateLink) for S3 so CloudFront uses private network paths.

○ **D.** Set up an AWS Site-to-Site VPN between your data center and AWS to encrypt traffic.

Explanation→

Question 14.6

A company wants external contractors to access AWS resources for a limited time without creating long-lived IAM users. Which approach is the best solution to provide temporary, limited AWS credentials?

○ **A.** Create IAM users for each contractor with password rotation every 30 days.

○ **B.** Use AWS STS to issue short-lived credentials that contractors assume via an IAM role.

○ **C.** Share a single IAM access key among contractors to simplify onboarding.

○ **D.** Place contractor accounts in a separate AWS Region to limit exposure.

Explanation→

Question 14.7

A finance team wants a single place to buy third-party software for multiple AWS accounts and to track who bought what for chargeback. Which Marketplace benefits help meet this need?

☐ **A.** Offers unified purchase records that map transactions to accounts for billing and reconciliation.

☐ **B.** Installs and configures purchased software in each account automatically without customer action.

☐ **C.** Automatically rotates customer encryption keys for marketplace products without customer involvement.

☐ **D.** Provides a single catalog where teams can find and purchase approved software across accounts.

☐ **E.** Replaces IAM and centrally enforces all runtime access controls for purchased software.

Explanation→

Question 14.8

Your company wants to convert recorded customer support calls into searchable text for analytics, and needs each utterance to show when it occurred and which participant said it. Select TWO features that meet these requirements using Amazon Transcribe.

☐ **A.** Automatically extracts named entities (people, locations) from the audio.

☐ **B.** Detects and labels individual speakers within the audio (speaker identification).

☐ **C.** Reports transcription quality using word error rate so you can evaluate accuracy.

☐ **D.** Provides sentiment scores for each sentence in the transcript.

☐ **E.** Produces transcripts that include timestamps for words and phrases.

Explanation→

Question 14.9

A finance leader needs an automated way to detect and notify teams when aggregated monthly charges across multiple AWS accounts approach their limit. Which single AWS capability best provides threshold alerts and projected end-of-month spend tracking across accounts?

○ **A.** Rely only on consolidated billing in AWS Organizations and review the monthly invoice.

○ **B.** Create an AWS Budgets budget configured for forecasted spend with alerts sent to owners.

○ **C.** Use cost allocation tags and analyze daily CSV exports to spot overruns manually.

○ **D.** Apply IAM policies to prevent users from creating resources once a spend limit is reached.

Explanation→

Question 14.10

Select TWO use cases that are appropriate for a service that extracts printed or handwritten text from images or video frames for automated processing (for example, OCR and text search)

☐ **A.** Automatically extracting line-item text from photographed receipts to populate expense forms.

☐ **B.** Flagging explicit visual content in images to prevent inappropriate material from being displayed.

☐ **C.** Identifying objects such as chairs, trees, and buildings in a photo for inventory tagging.

☐ **D.** Determining a person's emotional state from a still image to personalize content.

☐ **E.** Reading vehicle license plates from traffic camera frames to match plate numbers against a database.

Explanation→

Question 14.11

A company wants to keep an up-to-date copy of on-premises servers in AWS so that, if a primary site fails, recent data is available for quick boots of recovery hosts. Which AWS Elastic Disaster Recovery capability best meets this requirement?

○ **A.** Continuous block-level replication that streams ongoing disk changes to AWS for near-current copies of servers.

○ **B.** Manual snapshot exports where administrators periodically copy full disk images to S3 and restore when needed.

○ **C.** Traffic routing with a Global Accelerator that redirects users to alternate regions during failures.

○ **D.** Using read replicas in a managed database service to offload read traffic during peak load.

Explanation→

Question 14.12

A team wants a service that spreads incoming requests across instances in different AZs and avoids sending traffic to unhealthy servers. Which two capabilities of Elastic Load Balancing meet these requirements?

☐ **A.** Auto Scaling — automatically adds or removes instances based on demand to maintain capacity across AZs.

☐ **B.** Amazon S3 — stores objects across AZs to ensure availability and durability for static website content.

☐ **C.** Network Load Balancer — balances TCP/UDP traffic across instances in several AZs and performs health checks to remove unhealthy nodes.

☐ **D.** AWS Global Accelerator — optimizes global traffic paths to an endpoint, but it does not replace AZ-level health-based load balancing.

☐ **E.** Application Load Balancer — routes requests across targets in multiple AZs and checks target health before sending traffic.

☐ **F.** Route 53 latency routing — directs users to the Region with the lowest latency but does not perform health-based load distribution across instances within AZs.

Explanation→

Question 14.13

Which statements correctly describe how Amazon Route 53 health checks are used to influence DNS routing and availability?

☐ **A.** Health checks encrypt DNS queries between users and the application endpoints to improve data security.

☐ **B.** Using Route 53 health checks removes the need to run separate monitoring services such as CloudWatch.

☐ **C.** Health checks can be combined with routing policies to enable automatic failover between endpoints.

☐ **D.** Route 53 regularly probes endpoints and can stop sending DNS traffic to an endpoint that fails checks.

☐ **E.** Route 53 health checks require installing a software agent on each application server to report status.

☐ **F.** Health checks retain detailed historical performance and billing records for long-term cost allocation.

Explanation→

Question 14.14

A SaaS startup runs a stateless web tier on Amazon EC2 behind an Application Load Balancer across two Availability Zones. Instances are m5.large On-Demand with gp2 Amazon EBS volumes. Amazon CloudWatch shows average CPU near 20% with short weekday spikes to 60%. The team must lower costs this month without reducing resilience or doing a major refactor, and they want to stay on EC2. Which approach best aligns with rightsizing and matching supply with demand?

○ **A.** Replace m5.large with m6g.large (Graviton), switch EBS to gp3, and use EC2 Auto Scaling target tracking at 50% CPU.

○ **B.** Purchase a 1-year Compute Savings Plan sized to current m5.large usage and keep a fixed Auto Scaling desired capacity.

○ **C.** Upsize to m5.xlarge for headroom, switch EBS to gp3, and keep the Auto Scaling group at a fixed desired count.

○ **D.** Rebuild the web tier on AWS Lambda behind Amazon API Gateway to remove idle servers.

Explanation→

Question 14.15

Which security area can AWS Trusted Advisor automatically inspect and report prioritized findings for to help reduce public access and misconfiguration risk?

○ **A.** S3 bucket permissions and object access settings

○ **B.** VPC route tables and internet routing correctness

○ **C.** CloudWatch metric thresholds and alarm definitions

○ **D.** Application-level code vulnerabilities found in deployed apps

Explanation→

Question 14.16

Which statement best describes typical AWS billing for moving data between geographically separate Regions?

○ **A.** Data transferred within a single Availability Zone is more expensive than cross-Region transfers.

○ **B.** All data moving anywhere in AWS is billed at the same flat per-GB rate regardless of source or destination.

○ **C.** Data sent between two different AWS Regions usually incurs higher per-GB charges than traffic kept within the same Region.

○ **D.** Sending data from AWS to the public internet is generally cheaper than moving it between Regions.

Explanation→

Question 14.17

A web application is experiencing intermittent slow requests. Which two actions focus specifically on using end-to-end request traces to find the root cause?

☐ **A.** Visualize trace spans to identify which downstream dependency adds the most time to requests.

☐ **B.** Search application logs for error messages that match the slow request timestamps.

☐ **C.** Run synthetic tests from edge locations to measure global availability and response times.

☐ **D.** Aggregate CPU and memory metrics to detect overall resource exhaustion during slow periods.

☐ **E.** Create a dashboard of average request counts per minute to identify traffic spikes.

☐ **F.** Collect distributed traces to see latency across each microservice call in a single request path.

Explanation→

Question 14.18

Select TWO statements that describe how allocating large, upfront infrastructure costs across many customers affects cloud service pricing over time

☐ **A.** A broader subscriber pool lets the provider amortize capital outlays, enabling gradual price reductions for standard services.

☐ **B.** Unit prices fall because initial hardware and data center expenses are recovered across a large customer base.

☐ **C.** Pricing increases are inevitable because larger scale requires proportionally more staff to manage services.

☐ **D.** Spreading capital costs increases variability of monthly bills for each customer because of uneven allocation.

☐ **E.** Individual customers see immediate discounts only when they commit to long-term contracts, unrelated to shared capital costs.

☐ **F.** Higher per-customer fixed costs mean customers always pay more as usage grows.

Explanation→

Question 14.19

A startup stores 24 TB of application logs as raw CSV files in a single Amazon S3 prefix. Analysts use Amazon Athena; most queries filter by event_date and select only a few columns. Queries are slow and the Athena bill is high. The team wants to significantly reduce the data scanned per query while remaining serverless. What should they do?

○ **A.** Convert the dataset to Apache Parquet with Snappy compression and partition the S3 data by year/month/day; update the AWS Glue Data Catalog.

○ **B.** Enable Amazon S3 SSE-KMS on the bucket and encrypt Athena query results.

○ **C.** Run an AWS Glue crawler to catalog the existing CSV files in the AWS Glue Data Catalog.

○ **D.** Use AWS Lake Formation to grant analysts access to only required columns and tables.

Explanation→

Question 14.20

Select TWO choices that are NOT valid reasons to select multiple AWS Regions to improve end-user latency.

☐ **A.** Consolidates all user traffic through a single Region so every user accesses the same endpoint.

☐ **B.** Enables independent regional scaling so local demand can be met with reduced latency.

☐ **C.** Ensures lower network round-trip times by hosting resources close to users.

☐ **D.** Allows routing users to the Region that shows the lowest measured latency, improving user experience.

☐ **E.** Provides guaranteed zero-latency during geographic failover events.

Explanation→

Question 14.21

Which statement is NOT a way that economies of scale let AWS reduce customer unit prices?

○ **A.** Buying large quantities of hardware and negotiating better vendor pricing to lower per-unit costs.

○ **B.** Purchasing hardware in small quantities and keeping capacity isolated to drive up per-unit prices.

○ **C.** Pooling capacity across many customers so physical and virtual resources are shared and utilized efficiently.

○ **D.** Automating provisioning and operations so fewer manual tasks are needed and operational cost per unit falls.

Explanation→

Question 14.22

A company needs to move about 80 on-premises VMware VMs and a MySQL database to AWS within six weeks. They want to lift and shift the application servers to Amazon EC2 with minimal downtime, run test launches before cutover, and rightsize instances at launch. For the database, they want to migrate to Amazon RDS for MySQL using a full load followed by ongoing changes so the source stays online until cutover. Which TWO AWS services will perform these migrations with minimal downtime?

☐ **A.** AWS Schema Conversion Tool (AWS SCT) — converts database schema and code objects for heterogeneous migrations.

☐ **B.** AWS Systems Manager Automation — orchestrate post-launch configuration and runbooks across EC2.

☐ **C.** AWS Database Migration Service (AWS DMS) — full load plus CDC to Amazon RDS for MySQL with minimal downtime.

☐ **D.** AWS Migration Hub — centralized visibility and status tracking across migrations.

☐ **E.** Amazon S3 as an intermediate target for the database — use object storage as a landing zone during migration.

☐ **F.** AWS Application Migration Service (MGN) — agent-based, continuous block-level replication with test launches and rightsizing to EC2.

Explanation→

Question 14.23

Your team is evaluating procurement of third-party security tools through AWS Marketplace. Select TWO statements that represent anti-patterns or incorrect assumptions about using Marketplace for security products.

☐ **A.** A product bought from Marketplace will be fully pre-configured by AWS so no customer setup or tuning is required.

☐ **B.** Purchasing a security product from Marketplace removes the need to track who is billed; cost allocation and tagging are unnecessary.

☐ **C.** Marketplace can consolidate licensing and billing for third-party security solutions into a single AWS invoice.

☐ **D.** Marketplace offers a catalog of network and endpoint security products that organizations can evaluate and purchase through AWS.

☐ **E.** Vendors in Marketplace may provide automated deployment methods to speed installation, but customer configuration is still needed.

☐ **F.** Using Marketplace can simplify licensing terms compared with negotiating separate vendor contracts outside AWS.

Explanation→

Question 14.24

A company wants a single place to push and enforce consistent firewall and DDoS protection rules across many AWS accounts in its organization. Which AWS capability best meets this need?

○ **A.** A web application firewall you configure separately in each account for application-layer HTTP/S rules.

○ **B.** A central service that deploys and enforces protection policies across multiple accounts in an Organization.

○ **C.** A per-instance virtual firewall applied only to individual Elastic Network Interfaces.

○ **D.** A monitoring audit service that only records API activity for later review without enforcing rules.

Explanation→

Question 14.25

A startup just created a new AWS account to prototype a web API using two Amazon EC2 micro instances, Amazon S3, and AWS Lambda. They want to remain within the AWS Free Tier and receive an early warning before any overage charges occur. Which is the BEST solution?

○ **A.** Check AWS Cost Explorer at the end of each month to see which services consumed the Free Tier and adjust afterward.

○ **B.** Rely on Always Free benefits for AWS Lambda to guarantee no charges even if EC2 Free Tier hours are exceeded.

○ **C.** Stop nonessential instances and right-size resources without configuring alerts to stay within the monthly allowance.

○ **D.** Enable AWS Free Tier usage alerts and configure an AWS Budgets cost budget with email alerts at 80% (or $0) to notify when actual or forecasted charges approach the threshold.

Explanation→

Question 14.26

A company is deploying AWS Storage Gateway for on-premises applications. To align with security best practices, which action should they prioritize?

○ **A.** Right-size the local cache and plan network bandwidth.

○ **B.** Enable lifecycle transitions to S3 Intelligent-Tiering.

○ **C.** Choose Volume Gateway and use EBS snapshots for backup.

○ **D.** Use TLS in transit, SSE-KMS at rest, and IAM least privilege.

Explanation→

Question 14.27

A company wants to design a web application so a single physical data center failure does not cause an outage. Select TWO actions that use AWS Availability Zones to improve resilience

☐ **A.** Use edge locations to run primary application servers so users experience lower latency worldwide.

☐ **B.** Run all components in a single Availability Zone to simplify networking and speed up replication.

☐ **C.** Store backups only in the same Availability Zone as compute to reduce cross-zone data transfer costs.

☐ **D.** Rely on a single Availability Zone but increase instance size to handle peak load during failures.

☐ **E.** Host the primary database in one Availability Zone and deploy a synchronous standby replica in a different Availability Zone.

☐ **F.** Place application servers in at least two Availability Zones and use a load balancer to distribute traffic across them.

Explanation→

Question 14.28

A company is designing an encrypted VPN connection between its on-premises router and an AWS VPC. Which TWO statements about redundant VPN tunnels are correct?

☐ **A.** Redundant tunnels replace IPsec with TLS for stronger encryption.

☐ **B.** Paired tunnels allow the VPN connection to span multiple AWS Regions automatically.

☐ **C.** Each VPN connection includes a pair of independent IPSec tunnels to provide resilience.

☐ **D.** Only one tunnel can be active at a time, so paired tunnels provide no benefit during maintenance.

☐ **E.** Redundant tunnels eliminate the need to configure routing on the on-premises gateway.

☐ **F.** Having two tunnels improves availability by allowing automatic failover when one tunnel stops carrying traffic.

Explanation→

Question 14.29

A startup wants a managed database so they can recover from accidental data changes quickly without managing backup scripts. Which advantage of a managed database best matches this requirement?

○ **A.** Access to the underlying OS for full administrative control and custom backup tooling.

○ **B.** Ability to run on Spot instances to reduce costs for steady production workloads.

○ **C.** Direct control of network ACLs for segregating traffic between subnets.

○ **D.** Built-in automated backups with time-based recovery to restore data to a previous moment.

Explanation→

Question 14.30

A security team is updating an AWS sign-in policy to reduce account-takeover risk for high-privilege access across IAM and AWS IAM Identity Center. Which TWO actions are NOT recommended best practices for MFA?

☐ **A.** Use hardware security keys (FIDO2) or virtual MFA devices for second-factor verification.

☐ **B.** Add IAM deny conditions so actions like cloudtrail:StopLogging, kms:ScheduleKeyDeletion, and ec2:TerminateInstances require aws:MultiFactorAuthPresent.

☐ **C.** Always enable MFA for the AWS account root user.

☐ **D.** Require MFA in a role's trust policy to use AWS STS AssumeRole for an admin role (condition: aws:MultiFactorAuthPresent).

☐ **E.** Use long-lived access keys for privileged IAM users so they are not prompted for MFA.

☐ **F.** Make MFA optional for the break-glass group to speed up incident access to Billing and account settings.

Explanation→

Question 14.31

A company runs Amazon RDS, Amazon Aurora, and DynamoDB. They want a single place to define backup schedules, retention rules, and perform restores for these supported databases. Which is the best solution?

○ **A.** Use AWS Backup to centrally define policies and restore operations for the supported databases.

○ **B.** Manually create snapshots and on-demand backups within each database service's console when needed.

○ **C.** Deploy CloudFormation templates to create periodic snapshots for each database type and manage retention via template updates.

○ **D.** Install a third-party backup agent on every database instance and manage schedules from the agent's console.

Explanation→

Question 14.32

Which of the following statements about AWS Artifact is NOT correct?

○ **A.** It provides a central location to download AWS compliance reports and attestations.

○ **B.** It lets customers obtain AWS security and compliance documentation for regulatory review.

○ **C.** Access to its reports can help customers demonstrate AWS's controls during their compliance assessments.

○ **D.** It is a tool that issues compliance certificates for a customer's cloud workloads.

Explanation→

Question 14.33

A retail company wants to report how much its AWS-hosted applications reduced greenhouse gas emissions for its annual sustainability disclosure. Which AWS approach best provides quantifiable emissions attribution for their workloads?

○ **A.** Use the AWS Customer Carbon Footprint Tool to measure and attribute emissions from the company's AWS usage.

○ **B.** Enable AWS CloudTrail and analyze API call logs to calculate the data center energy savings for the workloads.

○ **C.** Run Cost Explorer reports and map cost savings to carbon reductions for each resource to infer emissions avoided.

○ **D.** Use AWS Trusted Advisor checks to identify idle resources and assume removed resources equal direct emissions reductions.

Explanation→

Question 14.34

Your security team finds sensitive customer files stored in an S3 bucket without server-side encryption enabled. Who is primarily responsible for enabling encryption for the objects stored in that bucket?

○ **A.** AWS — because AWS manages the physical storage hardware, it must encrypt customer objects automatically.

○ **B.** The customer's network team — they must encrypt data in transit before it reaches S3, so S3 object encryption is not needed.

○ **C.** AWS Support — open a support case and AWS Support will enable encryption for all existing objects.

○ **D.** The customer — they must enable and manage encryption settings for objects they place in S3.

Explanation→

Question 14.35

A financial services company must meet a strict data residency policy that requires all customer data, backups, and logs to remain in Germany. The team selected the EU (Frankfurt) Region and needs high availability and auditability. Which proposal would violate the residency requirement?

○ **A.** Enable Amazon S3 Cross-Region Replication from EU (Frankfurt) to EU (Paris) to improve disaster recovery.

○ **B.** Deploy Multi-AZ Amazon RDS in EU (Frankfurt) for high availability while keeping data in-Region.

○ **C.** Send AWS CloudTrail logs to an S3 bucket in EU (Frankfurt) and keep CloudWatch Logs in the same Region.

○ **D.** Use single-Region AWS KMS keys in EU (Frankfurt) and restrict other Regions with an AWS Organizations SCP using the aws:RequestedRegion condition.

Explanation→

Question 14.36

Your company uses a single AWS account. To reduce risk from compromise of the account's highest privilege identity, which single action is the best way to make accidental or unauthorized use of that identity significantly harder?

○ **A.** Create an administrative IAM user and use it for daily tasks instead of the root credentials.

○ **B.** Store the root account password in a secure vault and share it only with senior staff.

○ **C.** Require a hardware or virtual multi-factor authentication device for the account's root credentials.

○ **D.** Attach an IAM policy that denies sensitive actions unless a specific tag is present on requests.

Explanation→

Question 14.37

What primary benefit does AWS Security Hub provide when it *normalizes* security alerts from multiple sources?

○ **A.** It automatically blocks malicious traffic at the network edge before reaching AWS resources.

○ **B.** It stores long-term forensic logs for legal retention and e-discovery by default.

○ **C.** It converts varied tool alerts into a common format so teams can compare and act on them consistently.

○ **D.** It directly encrypts all customer data across AWS services using customer-managed keys.

Explanation→

Question 14.38

A company stores reports in an Amazon S3 bucket in Account A. A data analytics application in Account B uses an IAM role to download specific objects via the AWS SDK. The team already attached an identity-based policy to that role allowing s3:GetObject on the bucket ARN, but requests still fail with AccessDenied. The application cannot be modified to assume a role in Account A, and the company does not want to create users in Account A.

What is the simplest change to enable cross-account reads?

○ **A.** Attach another identity-based policy to the IAM role in Account B allowing s3:GetObject on the bucket.

○ **B.** Create an IAM role in Account A with S3 read permissions and a trust policy for Account B; update the app to assume this role.

○ **C.** Add an S3 bucket policy in Account A granting s3:GetObject to the IAM role ARN from Account B on the required bucket/prefix.

○ **D.** Create an IAM user in Account A with S3 read access and provide long-term access keys to the application.

Explanation→

Question 14.39

A company plans to move data-center servers to AWS and wants to identify which virtual machines can be right-sized to lower cost. Which Migration Evaluator capability directly provides measured CPU, memory, and I/O patterns to support sizing decisions?

○ **A.** Inventory analysis that collects system and dependency metadata.

○ **B.** TCO estimation that models total cost across on-premises and cloud.

○ **C.** Utilization profiling that captures CPU, memory and disk usage over time.

○ **D.** Licensing assessment that evaluates software portability and license costs.

Explanation→

Question 14.40

In AWS X-Ray, which feature provides a visual diagram that shows how services communicate and highlights latency between components for traced application requests?

○ **A.** A visual graph that displays service interactions and latency for traced requests.

○ **B.** A component that stores raw trace segment data for long-term archival and analysis.

○ **C.** An automated detector that flags anomalies and suggests root causes for traces.

○ **D.** A sampling controller that decides which requests are recorded to limit trace volume.

Explanation→

Question 14.41

A startup is re-architecting an image-processing workflow with the goals of paying only for actual usage, scaling automatically during spikes, and reducing

manual operations. Uploads land in Amazon S3, and processing should run only when needed. Which TWO choices are anti-patterns that would undermine these goals?

☐ **A.** Front the workflow with Amazon API Gateway invoking AWS Lambda so billing tracks requests and compute duration.

☐ **B.** Use an Amazon EventBridge schedule to run a Lambda function every minute to poll S3 for new objects.

☐ **C.** Maintain a fixed fleet of Amazon EC2 instances sized for peak traffic and handle scaling with manual scripts.

☐ **D.** Use Amazon DynamoDB on-demand for metadata writes from AWS Lambda to avoid capacity planning and pay per request.

☐ **E.** Configure Amazon S3 event notifications via Amazon EventBridge to invoke AWS Lambda when new objects are uploaded.

☐ **F.** Create Amazon CloudWatch alarms that start AWS Systems Manager Automation runbooks to retry failures and notify the team.

Explanation→

Question 14.42

A company needs to move dozens of virtual machines to AWS quickly with minimal changes to applications and operating systems. Which two actions align best with this migration approach?

☐ **A.** Use AWS Application Migration Service to replicate servers and cut over to AWS with little alteration.

☐ **B.** Refactor applications to use managed platform services and change application code for scalability.

☐ **C.** Recreate each application in a cloud-native architecture using microservices and serverless functions.

☐ **D.** Replace on-premises applications by subscribing to a new SaaS product and migrating users.

☐ **E.** Decommission unused systems and delete old data before migration to reduce migrated footprint.

☐ **F.** Lift existing VMs into AWS as-is, keeping current OS, configurations, and application binaries unchanged.

Explanation→

Question 14.43

A company runs a steady, 24x7 web/application tier whose instance attributes rarely change and also executes a nightly analytics batch that can tolerate

being stopped and resumed. They want to lower EC2 compute costs while meeting reliability needs. Which TWO purchasing choices best align with these workloads?

☐ **A.** Amazon EC2 Standard Reserved Instances (Regional, 1–3 years) for the steady 24x7 web tier; billing discount and instance size flexibility

☐ **B.** Bid above the On-Demand price on Spot to avoid interruptions and ensure availability

☐ **C.** Purchase Zonal Reserved Instances for the batch jobs to guarantee capacity in a specific Availability Zone

☐ **D.** Use Convertible Reserved Instances for batch jobs to switch instance families or OS during the term

☐ **E.** Amazon EC2 Spot Instances for nightly analytics; spare capacity at steep discount with 2-minute interruption notice

☐ **F.** Run the 24x7 web tier On-Demand to avoid commitment and scale instantly

Explanation→

Question 14.44

A media company uses Amazon S3 object-created events to invoke an AWS Lambda function that processes images and writes results to a downstream system. During marketing spikes, Lambda scales out quickly and the downstream system starts throttling. The team must prevent overload by limiting how many Lambda invocations run in parallel. They must keep the existing S3-to-Lambda integration and avoid adding new services. Which action is the MOST appropriate?

○ **A.** Enable provisioned concurrency on the Lambda function to keep execution environments warm and reduce cold starts.

○ **B.** Increase the function's memory configuration to reduce execution time and lower the chance of throttling.

○ **C.** Insert an Amazon SQS queue between S3 and the function and use an event source mapping to batch and buffer events.

○ **D.** Configure reserved concurrency on the Lambda function to cap its maximum concurrent executions and protect the downstream system.

Explanation→

Question 14.45

A retail web application experiences large traffic spikes during promotions. The team wants capacity to increase and decrease automatically to meet demand

while avoiding manual server changes. Which approach is the best fit?

○ **A.** Buy a fixed number of large reserved instances sized for peak traffic and use them throughout the year.

○ **B.** Configure Auto Scaling with scaling policies for the application fleet so instances launch and terminate based on load.

○ **C.** Manually add or remove on-demand instances each time an expected promotion begins or ends.

○ **D.** Use a single very large instance to handle all peak traffic and accept idle cost between promotions.

Explanation→

Question 14.46

Which alerting method delivers a direct message to named people when a cost threshold is exceeded in an AWS budget?

○ **A.** Sending email alerts configured on the budget

○ **B.** Triggering an automated IAM permission change

○ **C.** Publishing budget events only to an internal metrics store

○ **D.** Relying solely on scheduled billing reports

Explanation→

Question 14.47

A Solutions Architect is performing a structured architecture review. Which two outcomes are most likely from this review?

☐ **A.** A fully implemented, production-ready infrastructure deployed into the account.

☐ **B.** A report mapping the current design to best-practice principles with suggested improvements.

☐ **C.** A prioritized list of technical risks and recommendations to improve the design.

☐ **D.** A signed project plan and timeline for delivering code-level changes across teams.

☐ **E.** A completed cost forecast that guarantees lower monthly charges if recommendations are applied.

Explanation→

Question 14.48

An analytics team runs AWS Glue ETL jobs in private subnets to read and write large datasets in Amazon S3. Finance reports unexpectedly high NAT Gateway charges during the ETL windows. The team wants the smallest change to eliminate these data transfer costs while keeping the subnets private (no internet access). What is the BEST solution?

○ **A.** Enable S3 Intelligent-Tiering with lifecycle policies on the data lake buckets.

○ **B.** Add an Amazon S3 Gateway VPC endpoint and update private subnet route tables so ETL traffic to S3 stays private.

○ **C.** Use AWS Glue to convert JSON to Parquet and partition by date before querying with Amazon Athena.

○ **D.** Buy a Savings Plan to cover compute during the ETL windows.

Explanation→

Question 14.49

Your company wants to permanently close a standalone AWS account that is not part of AWS Organizations. The billing console does not show the option for any administrator IAM user. Which identity must perform the account closure to complete this irreversible action?

○ **A.** An IAM user with AdministratorAccess policy attached.

○ **B.** An IAM role assumed by the account's billing administrator.

○ **C.** The AWS Organizations management account owner (if the account is in an organization).

○ **D.** The AWS account root user (the original account owner credentials).

Explanation→

Question 14.50

A team uses AWS CloudFormation and wants to keep resources aligned with their templates over time. Which actions are INCORRECT when using drift detection?

☐ **A.** Automate periodic drift checks and alerts using scheduled jobs to catch divergence early.

☐ **B.** Make routine configuration changes directly in the console and skip drift checks to move faster.

☐ **C.** Run drift detection on the stack to identify resources whose configurations differ from the template.

☐ **D.** Assume CloudFormation automatically fixes any manual console edits, so drift detection is unnecessary.

☐ **E.** Review drift results and update the template, then perform a stack update to realign resource state.

Explanation→

Question 14.51

A company must move a production database from an on-premises Oracle engine to an Amazon Aurora PostgreSQL instance with minimal downtime and without losing in-flight transactions. Which solution best supports continuous replication and preserves data consistency during cutover?

○ **A.** Export a nightly full dump from Oracle, import into Aurora, and schedule a short maintenance window for final sync.

○ **B.** Create an RDS Read Replica of the on-premises Oracle instance and promote it to primary after replication completes.

○ **C.** Use AWS Database Migration Service with ongoing change replication from the source to the target, validate data, then cut over when synchronized.

○ **D.** Use AWS Schema Conversion Tool to migrate data in one step directly into Aurora without separate replication.

Explanation→

Question 14.52

Which statement about idempotence in infrastructure as code is NOT correct?

○ **A.** Applying the same template multiple times results in the same infrastructure state without unintended side effects.

○ **B.** Repeated runs of IaC always create duplicate resources, so manual cleanup is required after each deployment.

○ **C.** Idempotent deployments allow safe, repeatable updates and reduce configuration drift between environments.

○ **D.** Idempotence enables automation pipelines to redeploy templates safely as part of CI/CD workflows.

Explanation→

Question 14.53

Select TWO statements that correctly describe a capability of Convertible Reserved Instances for Amazon EC2 in a changing capacity requirement scenario

☐ **A.** Family changes are automatic when you stop and restart instances of a different instance family.

☐ **B.** You can exchange the unused value of a reservation to obtain reservations for different EC2 instance families.

☐ **C.** The reservation exchange preserves the benefit of lower hourly pricing compared with running instances entirely On-Demand.

☐ **D.** You must cancel the original reservation and wait for a refund before purchasing different instance family reservations.

☐ **E.** Exchanges are only allowed to the same instance family but a different availability zone.

Explanation→

Question 14.54

A company with dozens of Amazon S3 buckets across multiple teams discovers during a pre-audit that several buckets became publicly listable due to permissive bucket policies. The compliance lead asks for the fastest, account-wide way to prevent any existing or future buckets from becoming publicly accessible, with minimal ongoing management. What should a cloud practitioner do?

○ **A.** Encrypt all S3 buckets with SSE-KMS using a customer managed key.

○ **B.** Enable Amazon S3 Block Public Access at the account level.

○ **C.** Turn on AWS Security Hub to identify and alert on public S3 buckets.

○ **D.** Set each bucket's ACL to private and rely on IAM to control access.

Explanation→

Question 14.55

Which statement best describes how responsibility for security and compliance is handled when an organization runs a mix of on-premises and AWS resources (a hybrid deployment)?

○ **A.** Both the cloud provider and the customer share parts of security; the customer must apply matching controls across on-premises and cloud resources.

○ **B.** The cloud provider is fully responsible for all security and compliance for any resource the customer connects to the cloud.

○ **C.** The customer is solely responsible for physical security of the cloud provider's data centers when using hybrid architectures.

○ **D.** Security controls only need to be applied in the cloud; on-premises systems are outside the shared model.

Explanation→

Question 14.56

A company runs a web service that performs heavy numerical simulations using many CPU threads and keeps only small temporary state in memory. They must minimize cost while delivering high CPU throughput. Which EC2 instance family is the best choice?

○ **A.** Choose a memory-optimized instance family (large memory per vCPU).

○ **B.** Choose a general-purpose instance family (balanced vCPU and memory).

○ **C.** Choose a compute-optimized instance family (high vCPU-to-memory ratio).

○ **D.** Choose a storage-optimized instance family (high local I/O throughput).

Explanation→

Question 14.57

Select TWO options that describe appropriate uses or characteristics of a Gateway Load Balancer when integrating third-party virtual network appliances into an AWS VPC

☐ **A.** Provide transparent forwarding that lets the appliance receive and handle traffic without replacing client IP addresses.

☐ **B.** Act as a global content distribution service to cache and accelerate web assets for users worldwide.

☐ **C.** Optimize high-volume TCP passthrough with extremely low latency for static content delivery at the network edge.

☐ **D.** Replace the need for virtual appliances by providing built-in intrusion detection and custom traffic inspection rules.

☐ **E.** Insert a third-party virtual firewall inline so all traffic flows through the appliance before reaching backend instances.

☐ **F.** Terminate HTTPS connections and perform application-layer routing based on URL paths.

Explanation→

Question 14.58

A company plans to run containerized applications on AWS. Select TWO statements that correctly guide when to choose AWS Fargate versus Amazon EC2 for the container compute layer.

☐ **A.** Use Amazon EC2 when you require full control of the underlying instances, including custom AMIs or specialized kernel configuration.

☐ **B.** Choose AWS Fargate when you need direct SSH access to the container host for troubleshooting.

☐ **C.** Use AWS Fargate only for stateless containers; stateful containers must run on EC2.

☐ **D.** Choose Amazon EC2 when you want a serverless, per-task compute model without managing capacity.

☐ **E.** Use AWS Fargate when you want AWS to handle the container host lifecycle (provisioning, patching, and scaling) to reduce operational effort.

Explanation→

Question 14.59

A company uses AWS Organizations with consolidated billing and several linked accounts. Last month it purchased Savings Plans with upfront fees to reduce Amazon EC2 costs across all accounts. In AWS Cost Explorer, Finance is reviewing a monthly view using Unblended cost and still sees EC2 spend appear unchanged. They want the report to show the true effective cost of the commitments across the period and linked accounts. What should they do in Cost Explorer?

○ **A.** Use AWS Cost Explorer with daily granularity and group by API operation to reveal Savings Plans coverage.

○ **B.** Use AWS Cost Explorer and switch the metric to Amortized cost to spread Savings Plans/RI upfront fees over time across linked accounts.

○ **C.** Activate the Project tag as a cost allocation tag and rerun the same monthly view for last quarter.

○ **D.** In AWS Cost Explorer, group by Purchase option while keeping Unblended cost to separate On-Demand from Savings Plans.

Explanation→

Question 14.60

Which statement best describes the primary purpose of AWS Artifact?

○ **A.** It gives customers on-demand access to AWS audit and compliance documents they can use for their own audits.

○ **B.** It schedules and runs new third-party audits of a customer's AWS account on demand.

○ **C.** It stores a customer's own internal compliance reports and manages their retention policies.

○ **D.** It is a support channel for opening compliance-related service tickets with AWS Support.

Explanation→

Question 14.61

Which statement about a service that records resource configurations and checks them against desired states for continuous compliance is NOT correct?

○ **A.** It replaces API call auditing tools and provides the same detailed user activity logs as an API audit trail.

○ **B.** It records configuration histories and evaluates changes to identify drift from desired states.

○ **C.** It continuously evaluates resources using rules and can initiate automatic remediation when evaluations fail.

○ **D.** It offers prebuilt rule sets maintained by the provider and the ability to author custom rule logic for specialized checks.

Explanation→

Question 14.62

A company runs a web application on Amazon EC2 with Auto Scaling configured to add instances during traffic spikes and remove them when idle. Which statement about this design is WRONG?

○ **A.** Auto Scaling helps meet increased demand while avoiding payment for idle capacity.

○ **B.** Auto Scaling guarantees zero downtime for any infrastructure failure without additional architectural changes.

○ **C.** Auto Scaling can improve performance only when combined with appropriate scaling policies and monitoring.

○ **D.** Auto Scaling supports cost optimization by reducing the number of running instances when load decreases.

Explanation→

Question 14.63

A company must show who changed critical IAM settings during a recent incident and provide evidence for a regulatory audit. Which two outcomes are best supported by enabling AWS CloudTrail?

☐ **A.** Directly reduces monthly compute charges by optimizing instance usage.

☐ **B.** Encrypts application data stored in S3 buckets to ensure data-at-rest confidentiality.

☐ **C.** Creates tamper-evident, time-ordered records of API activity to support investigations.

☐ **D.** Automatically blocks malicious API requests in real time to prevent further unauthorized changes.

☐ **E.** Provides verifiable event logs that auditors can review to confirm compliance activities and shorten audit cycles.

Explanation→

Question 14.64

A small business plans to store customer documents in Amazon S3. Which security task is primarily the customer's responsibility to ensure sensitive files are handled correctly?

○ **A.** Maintaining the physical servers and data center power systems that host S3.

○ **B.** Assign sensitivity levels and handling rules to each dataset (data classification).

○ **C.** Automatically rotating AWS-managed encryption keys used by S3 without customer involvement.

○ **D.** Operating global edge locations and networking for Amazon's CDN services.

Explanation→

Question 14.65

A healthcare startup is building a patient portal on Amazon EC2, Amazon RDS, and Amazon S3. They will handle protected health information (PHI) and are preparing for an external audit. The team wants to use AWS Artifact to do two things: (1) present auditor-verified evidence of AWS-owned controls for the underlying AWS infrastructure and managed services they use, and (2) accept any required account-level agreement to build under HIPAA. Which actions should they take using AWS Artifact?

☐ **A.** Download the SOC 2 Type II report and program summaries from AWS Artifact Reports to evidence AWS-owned controls.

☐ **B.** Review and accept the HIPAA Business Associate Addendum in AWS Artifact Agreements for the AWS account handling PHI.

☐ **C.** Enable AWS CloudTrail and AWS Config, then present those logs and configuration histories as customer-owned control evidence.

☐ **D.** Use AWS Artifact to automatically validate and monitor the startup's EC2, RDS, and S3 configurations for HIPAA compliance.

☐ **E.** Use AWS Artifact to generate least-privilege IAM policies and enable KMS encryption for the application.

☐ **F.** Publish the downloaded SOC reports on the company's public website so all stakeholders can access them.

Explanation→

Exam 14 — Answer Key

Question	Correct answer(s)
14.1	D
14.2	B, E
14.3	B, F
14.4	B
14.5	A
14.6	B
14.7	A, D
14.8	B, E
14.9	B
14.10	A, E
14.11	A
14.12	C, E
14.13	C, D
14.14	A
14.15	A
14.16	C
14.17	A, F
14.18	A, B
14.19	A
14.20	A, E
14.21	B
14.22	C, F
14.23	A, B
14.24	B
14.25	D
14.26	D
14.27	E, F

Question	Correct answer(s)
14.28	C, F
14.29	D
14.30	E, F
14.31	A
14.32	D
14.33	A
14.34	D
14.35	A
14.36	C
14.37	C
14.38	C
14.39	C
14.40	A
14.41	B, C
14.42	A, F
14.43	A, E
14.44	D
14.45	B
14.46	A
14.47	B, C
14.48	B
14.49	D
14.50	B, D
14.51	C
14.52	B
14.53	B, C
14.54	B
14.55	A
14.56	C
14.57	A, E

Question	Correct answer(s)
14.58	A, E
14.59	B
14.60	A
14.61	A
14.62	B
14.63	C, E
14.64	B
14.65	A, B

Exam 14 Explanations

14.1 — Well-Architected security best practices (D1.T1.2)

Domain 1 • Task 1.2 ↑ Back to Question 14.1

Which of the following statements about granting user permissions in AWS is NOT correct?

Options & Rationales

A ✗ — Assign roles or policies that allow only the actions a user needs to perform their job.

Rationale: This follows the principle of granting minimal necessary permissions and is a correct practice.

B ✗ — Use temporary credentials or role assumption for tasks that need elevated permissions for a limited time.

Rationale: Using short-lived credentials for temporary elevation limits exposure and aligns with least-privilege guidance.

C ✗ — Regularly review and remove unused permissions or accounts to reduce unintended access.

Rationale: Periodic pruning of permissions reduces risk and supports least-privilege practices.

D ✓ — Give administrators broad, permanent full-access policies to simplify troubleshooting and avoid permission errors.

Rationale: Granting unrestricted, long-lived full access violates the principle of least privilege and increases risk.

Explanation

The key concept tested is granting only the permissions required for each identity to reduce the chance of accidental or malicious misuse. Least-privilege means giving the smallest set of privileges necessary, limiting how long elevated access exists, and removing access that is no longer needed.

Why the incorrect option is wrong. One option recommends assigning broad, permanent full-access rights to administrators to avoid permission problems. That practice increases the attack surface because a compromise or mistake by an administrator would have extensive impact. Permanent unrestricted access also makes auditing and accountability harder, since many actions are permitted without fine-grained control.

Why the other options are correct. The other choices reflect limiting permissions to required actions, using temporary credentials or role assumption to provide time-bound elevation, and routinely reviewing and removing unused

permissions. These steps reduce exposure, make audits meaningful, and limit the blast radius of compromised credentials.

Key Concepts

- **Least privilege:** Only grant the minimal actions and resources an identity needs to perform tasks; reduces risk from mistakes and compromises.
- **Temporary elevation:** Use short-lived credentials or assume-role patterns for tasks that require higher privileges, so elevated rights do not persist unnecessarily.
- **Access hygiene:** Periodic reviews and removal of unused permissions or accounts maintain a smaller, more auditable permission set.

Common Pitfalls

- Assuming broad permissions are acceptable because they 'save time' ignores increased security risk and makes incident investigations more difficult. Another mistake is failing to rotate or expire elevated credentials, which extends exposure.

Glossary

- **Identity and Access Management (IAM):** AWS service to create and manage users, roles, and policies that control access to AWS resources.
- **Role assumption / temporary credentials:** Mechanisms to grant short-lived permissions without creating long-lived credentials.

14.2 — VPC components: subnets, routes, IGW (D3.T3.5)

Domain 3 • Task 3.5 ↑ Back to Question 14.2

An organization runs application servers in private subnets across two Availability Zones in an IPv4-only VPC. The servers must download OS patches and packages from various internet repositories, but must not be reachable from the internet. An Internet Gateway is already attached to the VPC, and a web tier in a public subnet has internet access.

Which actions will provide outbound-only internet access for the private subnets while keeping inbound blocked?

Options & Rationales

A ✗ — Add a 0.0.0.0/0 route in the private subnets' route table that targets the Internet Gateway.

Rationale: Pointing private subnets to an Internet Gateway makes them public by routing definition and still doesn't enable outbound for private IPs without public addresses.

B ✓ — Add a 0.0.0.0/0 route in the private subnets' route table that targets the NAT Gateway.

Rationale: Private subnets need a default route to the NAT Gateway to send outbound traffic while preventing unsolicited inbound connections.

C ✗ — Attach an Egress-Only Internet Gateway and add ::/0 in the private subnets' route table.

Rationale: An Egress-Only Internet Gateway is for IPv6 outbound-only traffic. It does not help in an IPv4-only environment.

D ✗ — Enable auto-assign public IPv4 addresses on the private subnets.

Rationale: Assigning public IPs would allow direct internet reachability, violating the requirement to avoid inbound access and still requires an IGW route.

E ✓ — Create a NAT Gateway in a public subnet.

Rationale: A NAT Gateway must reside in a public subnet to provide outbound-only internet access for instances in private subnets.

F ✗ — Create a VPC endpoint for Amazon S3.

Rationale: VPC endpoints give private access to AWS services only. They do not provide general internet access to various external repositories.

Explanation

To give instances in private subnets outbound-only internet access in an IPv4 VPC, use a NAT Gateway. Subnets are considered public or private based on routing, not their names. A public subnet has a route to an Internet Gateway (IGW); a private subnet does not. Instances in private subnets have only private IP addresses, so they cannot use an IGW directly for outbound traffic. Instead, they send traffic to a NAT Gateway placed in a public subnet. The NAT Gateway then uses the VPC's IGW to reach the internet, while unsolicited inbound connections from the internet cannot be initiated to the private instances.

The correct pattern therefore has two parts: place a NAT Gateway in a public subnet, and update the private subnets' route table to include a default route (0.0.0.0/0) that targets the NAT Gateway. This preserves the private status of the subnets (no direct IGW route from those subnets) and satisfies the requirement for outbound-only internet access for software updates.

Routing private subnets directly to an IGW would change them to public subnets by routing definition and still would not enable outbound communication for private IP addresses unless those instances also had public IPs. Enabling auto-assign public IPv4 addresses would make the instances directly reachable from the internet (if a route to an IGW existed), violating the "not reachable" requirement. VPC endpoints provide private connectivity to specific AWS services only; they do not provide general internet access to arbitrary external repositories. Finally, an Egress-Only Internet Gateway applies to IPv6 traffic and does not help for IPv4-only workloads.

Key Concepts

- **Public vs. private subnet:** Determined by routing. A route to an Internet Gateway makes a subnet public; no IGW route keeps it private.

- **NAT Gateway in a public subnet:** Enables outbound-only internet for instances in private subnets when those subnets route 0.0.0.0/0 to the NAT.
- **Internet Gateway (IGW):** Required for internet connectivity, but private instances need NAT or public IPs to use it.
- **VPC endpoints:** Provide private access to AWS services only, not general internet.

Common Pitfalls

- Adding a default route to an IGW from private subnets, which makes them public and still doesn't work for private IPs.
- Assigning public IPs to private subnets, which breaks the "no inbound internet" requirement.

14.3 — STS temporary credentials for roles (D2.T2.3)

Domain 2 • Task 2.3 ↑ Back to Question 14.3

A security engineer needs short-lived credentials for an IAM user who must use multi-factor authentication (MFA) before accessing AWS APIs. Which TWO statements correctly describe how AWS Security Token Service (STS) GetSessionToken supports this requirement?

Options & Rationales

A ✗ — GetSessionToken creates a new IAM role which the user assumes to gain temporary permissions.

Rationale: GetSessionToken does not create or assume roles; it issues temporary credentials for the existing IAM user, not a new role.

B ✓ — GetSessionToken returns temporary access keys and a session token that expire after a configurable short duration.

Rationale: GetSessionToken issues time-limited access key ID, secret access key, and a session token; callers use these temporary credentials until they expire.

C ✗ — GetSessionToken returns permanent long-term credentials usable until explicitly revoked.

Rationale: Credentials from GetSessionToken are intentionally temporary and expire automatically; they are not permanent or revocable like long-term keys.

D ✗ — GetSessionToken is intended for identity federation with external SAML or OIDC providers.

Rationale: Federation with SAML or web identity typically uses AssumeRoleWithSAML or AssumeRoleWithWebIdentity, not GetSessionToken.

E ✗ — GetSessionToken modifies an IAM user's inline policies to restrict access during the session.

Rationale: GetSessionToken does not change IAM policies; it issues temporary credentials whose permissions are the same as the IAM user's permissions.

F ✓ — When an MFA serial number and one-time code are provided, GetSessionToken can require MFA before issuing temporary credentials.

Rationale: Supplying an MFA device identifier and token to GetSessionToken enforces an MFA check and produces credentials tied to that authenticated session.

Explanation

AWS Security Token Service (STS) can issue short-lived credentials for IAM users using the GetSessionToken API. These credentials consist of an access key ID, a secret access key, and a session token; they are time-limited so access automatically expires after the session duration.

Why the correct statements are right. GetSessionToken issues a credential triplet that has a built-in expiry, matching the need for short-lived access. When callers include MFA details, STS validates the one-time code and issues credentials only after MFA succeeds, satisfying the requirement for MFA-protected access.

Why the incorrect statements are wrong. GetSessionToken does not create roles or change policy definitions; role-based temporary access and federation from external identity providers use different STS operations (for example, role-assumption APIs). The credentials returned are temporary by design and cannot be treated as permanent long-term keys.

Key Concepts

- **Temporary credentials:** Short-lived access keys plus a session token issued by STS, used instead of long-term credentials to reduce risk. These expire after a specified duration.
- **MFA enforcement with GetSessionToken:** Supplying the MFA device serial and the one-time code to GetSessionToken forces an MFA check; the returned temporary credentials are contingent on that successful authentication.
- **Scope of GetSessionToken:** It provides credentials for an existing IAM user and does not create or assume roles, nor does it alter IAM policies.

Common Pitfalls

- Confusing GetSessionToken with role-assumption or federation APIs leads to choosing the wrong STS operation. GetSessionToken is for IAM user sessions, while AssumeRoleWithSAML/WithWebIdentity are for federated identities.
- Assuming temporary credentials modify IAM policies: temporary credentials inherit the IAM user's permissions but do not change policy documents.

14.4 — DynamoDB NoSQL use cases (D3.T3.4)

Domain 3 • Task 3.4 ↑ Back to Question 14.4

A retailer runs an order-processing microservice that stores data in Amazon DynamoDB. During flash sales, customers place many orders concurrently. For each order, the application must create an order record and decrement

inventory items. The company must guarantee that either all updates are applied together or none are applied to avoid overselling. Which is the BEST solution?

Options & Rationales

A ✗ — Use DynamoDB Streams to detect oversells and trigger an asynchronous workflow to reverse conflicting updates.

Rationale: Streams are asynchronous change capture. They react after writes occur and cannot enforce pre-write atomicity, so they do not prevent oversell.

B ✓ — Use DynamoDB ACID transactions to group the order write and inventory decrements into a single all-or-nothing operation.

Rationale: ACID transactions provide atomic multi-item writes, ensuring all updates succeed or none apply—directly preventing oversell. Trade-off: slightly higher latency and capacity per transaction.

C ✗ — Add DynamoDB Accelerator (DAX) to cache inventory reads for microsecond latency before writing updates.

Rationale: DAX speeds reads but does not change write semantics or provide atomic multi-item updates; caching cannot prevent concurrent write conflicts.

D ✗ — Create a Global Secondary Index (GSI) on sku to query inventory updates quickly across items and maintain consistency.

Rationale: GSIs add query paths, not constraints. They do not provide atomic writes across multiple items and cannot prevent oversell.

Explanation

The requirement is strict, all-or-nothing behavior across multiple updates: either the order record and all inventory decrements succeed together, or nothing is applied. In DynamoDB, this is precisely the use case for ACID transactions. Transactions group multiple item operations into a single, atomic unit so that partial success is impossible. The learning example explicitly calls out preventing oversell as a scenario where multi-item consistency is required, making ACID transactions the best fit.

DynamoDB Streams capture item changes for event-driven workflows after writes have occurred. Because Streams are asynchronous, they cannot block or atomically coordinate the updates needed to prevent oversell; at best, they enable compensating actions after the fact, which doesn't meet the preventative guarantee.

DynamoDB Accelerator (DAX) is a managed, in-memory cache that improves read latency to microseconds. While valuable for read-heavy workloads, DAX does not alter DynamoDB's write behavior or provide atomic multi-item guarantees, so it cannot prevent concurrent oversell conditions.

Global Secondary Indexes (GSIs) add additional query paths across partitions to support access patterns without scans, but they do not enforce constraints or

provide atomicity across multiple items. An index cannot ensure that multiple related updates succeed or fail as a single unit.

Therefore, using ACID transactions in DynamoDB is the only option that directly enforces the required all-or-nothing behavior to prevent overselling during high-concurrency events like flash sales.

Key Concepts

- **ACID transactions:** Provide atomic, consistent, isolated, durable multi-item operations so related writes succeed or fail together.
- **DynamoDB Streams:** Asynchronous change data capture for event-driven workflows; not a pre-write enforcement or atomicity mechanism.
- **DynamoDB Accelerator (DAX):** In-memory cache for microsecond read performance; does not change write semantics or provide cross-item atomicity.
- **Global Secondary Index (GSI):** Additional query path for different access patterns; not a tool for enforcing multi-item consistency.

Common Pitfalls

- Assuming Streams can fix oversell in real time; they act after writes and cannot guarantee prevention. Believing DAX or a GSI can enforce multi-item atomicity is also incorrect.

14.5 — Model for encryption at rest/in transit (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 14.5

A company serves customer PII from an Amazon S3 bucket through Amazon CloudFront. A security review requires encryption in transit end to end (viewer to CloudFront and CloudFront to S3). Review the abbreviated configuration and choose the single change that best meets the requirement with minimal architecture changes.

Table 4: Exhibit: CloudFront distribution (abbreviated)

Setting	Current value
Viewer protocol policy	Allow HTTP and HTTPS
Origin	S3 bucket (my-bucket)
Origin protocol policy	HTTP Only
S3 encryption at rest	SSE-KMS: Enabled

Options & Rationales

A ✓ — Update CloudFront to enforce HTTPS: set Viewer Protocol Policy to Redirect HTTP to HTTPS and Origin Protocol Policy to HTTPS Only.

Rationale: Amazon CloudFront: enforce TLS to viewers and use TLS to S3, achieving end-to-end encryption in transit with minimal changes.

B ✗ — Enable S3 default encryption with SSE-KMS and rely on CloudTrail logs for key usage.

Rationale: Amazon S3 + AWS KMS: protects data at rest and audits key use; it does not enforce TLS for data in transit.

C ✗ — Create a VPC interface endpoint (AWS PrivateLink) for S3 so CloudFront uses private network paths.

Rationale: PrivateLink/VPC endpoints: secure VPC-to-service traffic; CloudFront does not route through your VPC endpoints and this doesn't enforce viewer/origin TLS.

D ✗ — Set up an AWS Site-to-Site VPN between your data center and AWS to encrypt traffic.

Rationale: Site-to-Site VPN: encrypts hybrid links; it does not affect viewer-to-CloudFront or CloudFront-to-S3 paths.

Explanation

The goal is encryption in transit for both legs of the delivery path: from end users (viewers) to the CDN edge and from the CDN edge to the S3 origin. In AWS, AWS provides TLS-capable service endpoints and managed certificates (via AWS Certificate Manager), but customers must configure services to require HTTPS. With Amazon CloudFront, two controls enforce this:

- Viewer Protocol Policy controls how viewers connect. Setting it to Redirect HTTP to HTTPS forces clients to use TLS (e.g., TLS 1.2/1.3) to reach CloudFront.
- Origin Protocol Policy controls how CloudFront connects to the origin. Setting it to HTTPS Only ensures TLS is used from CloudFront to the S3 origin.

In the exhibit, encryption at rest on S3 is already enabled with SSE-KMS, which addresses data-at-rest requirements and enables CloudTrail logging for KMS key usage, but it does not provide encryption in transit. Therefore, the minimal and correct change is to update CloudFront to require HTTPS for both viewer and origin connections.

Private networking options like VPC interface endpoints (AWS PrivateLink) keep traffic off the public internet for resources inside a VPC. CloudFront is a global edge service and does not route through your VPC endpoints, so creating an S3 interface endpoint would not change the CloudFront-to-S3 path or enforce TLS. Similarly, AWS Site-to-Site VPN is for hybrid connectivity between on-premises networks and a VPC; it does not affect viewer traffic to CloudFront or CloudFront's connection to S3.

Thus, changing CloudFront's viewer policy to Redirect HTTP to HTTPS and its origin policy to HTTPS Only satisfies the end-to-end encryption-in-transit

requirement with minimal architecture changes.

Key Concepts

- **Encryption in transit (TLS/HTTPS):** Protects data while moving between clients and services; must be enforced by configuration (e.g., CloudFront viewer/origin policies).
- **Encryption at rest (SSE-KMS):** Protects stored data and provides key-use audit via CloudTrail; does not secure data on the wire.
- **CloudFront policies:** Viewer Protocol Policy controls client connections; Origin Protocol Policy controls CloudFront-to-origin connections (e.g., HTTPS Only for S3).
- **VPC endpoints/PrivateLink vs. edge:** PrivateLink secures VPC-to-service traffic; CloudFront, as an edge service, does not use your VPC endpoints.

Common Pitfalls

- Assuming SSE-KMS (at rest) also encrypts data in transit. Transport security must be enforced separately with HTTPS/TLS.
- Believing a VPC endpoint or VPN will affect CloudFront's viewer path or enforce TLS to S3.

Glossary

- **TLS:** Transport Layer Security; protocol for encrypting data in transit.
- **SSE-KMS:** Server-side encryption with AWS KMS keys for data at rest in S3.

14.6 — Federation with external IdP (D2.T2.3)

Domain 2 • Task 2.3 ↑ Back to Question 14.6

A company wants external contractors to access AWS resources for a limited time without creating long-lived IAM users. Which approach is the best solution to provide temporary, limited AWS credentials?

Options & Rationales

A ✗ — Create IAM users for each contractor with password rotation every 30 days.

Rationale: Individual IAM users produce long-lived credentials that need management and do not provide automatic short expiry, increasing operational overhead.

B ✓ — Use AWS STS to issue short-lived credentials that contractors assume via an IAM role.

Rationale: AWS STS issues temporary access keys linked to an IAM role, enforcing time-limited, least-privilege access without long-lived user credentials.

C ✗ — Share a single IAM access key among contractors to simplify onboarding.

Rationale: Sharing one access key reduces accountability and creates a long-lived secret that cannot be scoped to each contractor or automatically

expire.

D ✗ — Place contractor accounts in a separate AWS Region to limit exposure.

Rationale: Using a different Region does not create temporary credentials or reduce credential longevity; it only changes resource location.

Explanation

When external personnel need temporary access to AWS, the secure and recommended pattern is to provide time-limited credentials rather than permanent IAM users or shared secrets. A service exists that issues short-duration security tokens which are associated with an IAM role; principals use those tokens to make API calls for the allowed period. This approach supports least-privilege by attaching role permissions and reduces risk from credential leakage because tokens automatically expire.

Why the correct choice fits. It leverages a short-lived token issuance service combined with IAM roles, giving contractors only the permissions defined by the role and removing the need to manage long-term credentials.

Why the other choices are not appropriate:

- Creating individual long-lived IAM users still requires managing passwords and keys, and those credentials persist until rotated or revoked, raising operational and security burden.
- Sharing a single access key destroys individual accountability, increases blast radius if compromised, and does not provide automatic expiry.
- Moving accounts or resources to another Region does not solve credential lifecycle or permission scoping; Region placement concerns latency and data residency, not temporary access.

Key Concepts

- **Short-lived credentials:** Temporarily valid security tokens that reduce risk from leaked credentials by expiring automatically.
- **IAM role:** A set of permissions that can be assumed by trusted principals to gain temporary access without permanent credentials.

Common Pitfalls

- Confusing resource location with credential lifetime; changing Regions does not limit credential validity.
- Thinking rotation alone equals temporary access; rotation still relies on long-lived secrets until replaced.

Glossary

- **IAM role:** A permission container that can be assumed to obtain access rights for a session.
- **Temporary token:** A short-duration credential used for authenticated AWS API calls.

14.7 — Marketplace benefits for procurement (D4.T4.3)

Domain 4 • Task 4.3 ↑ Back to Question 14.7

A finance team wants a single place to buy third-party software for multiple AWS accounts and to track who bought what for chargeback. Which Marketplace benefits help meet this need?

Options & Rationales

A ✓ — Offers unified purchase records that map transactions to accounts for billing and reconciliation.

Rationale: Consolidated purchase tracking enables accurate chargeback, cost reporting, and reconciliation by linking license or subscription purchases to specific accounts.

B ✗ — Installs and configures purchased software in each account automatically without customer action.

Rationale: Marketplace supports procurement and licensing; deployment and configuration are performed by the customer or deployment tools, not automatically by Marketplace.

C ✗ — Automatically rotates customer encryption keys for marketplace products without customer involvement.

Rationale: Key rotation is a data protection task managed by KMS or the customer; Marketplace does not automatically manage encryption keys for buyer data.

D ✓ — Provides a single catalog where teams can find and purchase approved software across accounts.

Rationale: A centralized software catalog simplifies procurement by giving authorized users one place to discover and buy approved products for organization-wide use.

E ✗ — Replaces IAM and centrally enforces all runtime access controls for purchased software.

Rationale: Identity and access control remain the customer's responsibility through IAM or Identity Center; Marketplace does not substitute for IAM governance.

Explanation

Centralized procurement gives organizations one place to acquire third-party solutions and maintain a clear record of purchases. A central catalog reduces procurement fragmentation so teams use approved vendors and offerings. Unified purchase records let finance and operations reconcile costs, enable chargeback or showback, and simplify license audits across multiple accounts or organizational units.

Why the correct choices are right. A central catalog reduces duplicate purchases and enforces approved sourcing by giving users one location

to find authorized software. Consolidated purchase records provide the transaction-level visibility finance needs to map costs back to teams or accounts for reporting and chargeback.

Why the incorrect choices are wrong. Automatic rotation of encryption keys is not a Marketplace feature—key lifecycle is handled by services like KMS and the customer. Marketplace does not replace IAM; identity and access controls remain a customer responsibility. Marketplace facilitates buying and licensing but does not automatically deploy or configure software across accounts; deployment requires customer actions or automation tools.

Key Concepts

- **Central catalog:** A single curated listing of vendor offerings that simplifies discovery and ensures consistent procurement choices across teams.
- **Purchase tracking:** Consolidated transaction records that associate purchases with accounts, products, and subscriptions to support billing reconciliation and cost allocation.
- **Separation of responsibilities:** Procurement and licensing visibility are provided by the marketplace, while runtime access control and encryption key management remain with the customer.

Common Pitfalls

- Assuming Marketplace performs runtime security or deployment tasks leads to incorrect choices; its primary role for procurement is discovery, purchasing, and purchase visibility.

Glossary

- **Catalog:** A centralized listing of available software products.
- **Chargeback:** Allocating costs to the teams or accounts that incurred them based on purchase records.

14.8 — NLP services: Comprehend & Transcribe (D3.T3.7)

Domain 3 • Task 3.7 ↑ Back to Question 14.8

Your company wants to convert recorded customer support calls into searchable text for analytics, and needs each utterance to show when it occurred and which participant said it. Select TWO features that meet these requirements using Amazon Transcribe.

Options & Rationales

A ✗ — Automatically extracts named entities (people, locations) from the audio.

Rationale: Entity extraction operates on text and is offered by a separate language-analysis service after transcription.

B ✓ — Detects and labels individual speakers within the audio (speaker identification).

Rationale: Transcribe can separate and tag multiple speakers in a recording,

enabling per-speaker analysis and attribution.

C ✗ — Reports transcription quality using word error rate so you can evaluate accuracy.

Rationale: Word error rate is a metric used to measure transcription performance, but it is not a feature that directly provides timestamps or speaker tags.

D ✗ — Provides sentiment scores for each sentence in the transcript.

Rationale: Transcribe Call Analytics can generate sentiment insights over call transcripts, but sentiment alone does not satisfy this scenario's requirements for word-level timestamps and per-speaker labels.

E ✓ — Produces transcripts that include timestamps for words and phrases.

Rationale: Amazon Transcribe outputs time-aligned text so analytics systems can locate when words or phrases occurred in audio.

Explanation

Amazon Transcribe is a managed speech-to-text service that converts audio into machine-readable text. For recorded calls where you need to know when words were spoken and which participant said them, two capabilities are essential: time-aligned transcripts and speaker separation. Time-aligned transcripts attach timestamps to words or phrases so downstream analytics can locate or index audio segments. Speaker separation (speaker identification or speaker diarization) groups speech by distinct voices so you can attribute statements to specific participants.

The other choices describe text-analysis functions or evaluation metrics. Sentiment scoring and named-entity extraction operate on the transcribed text and may be provided by services such as Amazon Comprehend or Amazon Transcribe Call Analytics, but these higher-level analytics do not themselves provide the core transcription features needed in this scenario (timestamps and per-speaker labels). Word error rate is a way to quantify how accurate a transcription is during evaluation, but it does not itself provide the operational features needed to make transcripts searchable or to attribute speech to speakers.

Key Concepts

- **Speech-to-text transcription:** Converting spoken audio into text so it can be searched, indexed, or analyzed.
- **Time-aligned transcripts:** Transcriptions that include timing information for words or phrases, enabling synchronization with the original audio.
- **Speaker diarization (speaker identification):** The process of distinguishing and labeling different speakers within a single audio file to attribute utterances.
- **Downstream NLP vs. transcription:** Higher-level text analyses (sentiment, entity extraction) are applied to transcript text by separate language-analysis

tools.

Common Pitfalls

- Assuming the speech-to-text service also performs sentiment or entity analysis; these are separate steps that require a text-analysis tool.
- Confusing evaluation metrics with features; metrics like word error rate help assess quality but do not provide functionality such as timestamps or speaker labels.

Glossary

- **Transcript:** The text output produced from audio after speech recognition.
- **Speaker diarization:** Automatic grouping of audio segments by speaker identity without prior speaker labels.

14.9 — Forecasting & controlling spend (multi-account) (D4.T4.2)

Domain 4 • Task 4.2

↑ Back to Question 14.9

A finance leader needs an automated way to detect and notify teams when aggregated monthly charges across multiple AWS accounts approach their limit. Which single AWS capability best provides threshold alerts and projected end-of-month spend tracking across accounts?

Options & Rationales

A ✗ — Rely only on consolidated billing in AWS Organizations and review the monthly invoice.

Rationale: Consolidated billing centralizes invoicing but does not provide automated threshold alerts or forecasted spend notifications before month end.

B ✓ — Create an AWS Budgets budget configured for forecasted spend with alerts sent to owners.

Rationale: AWS Budgets lets you set thresholds on actual and forecasted spend and send notifications to recipients across accounts, giving proactive alerts when projected month-end charges approach limits.

C ✗ — Use cost allocation tags and analyze daily CSV exports to spot overruns manually.

Rationale: Tags improve chargeback and reporting but require manual processing; they do not automatically generate threshold alerts or forecast projections.

D ✗ — Apply IAM policies to prevent users from creating resources once a spend limit is reached.

Rationale: IAM controls permissions, not billing forecasts; it cannot natively enforce spend limits or send forecast-based notifications across accounts.

Explanation

For proactive multi-account cost control, you want an automated mechanism that monitors both actual charges and projected month-end spend and notifies stakeholders before budgets are exceeded. AWS Budgets is purpose-built for setting cost thresholds based on current usage and forecasted trends, and it can send alerts to specified recipients across accounts. This enables teams to take corrective actions early, such as scaling back resources or investigating unexpected usage.

Consolidated billing centralizes invoices and simplifies payment, but it is a reporting and payment consolidation feature rather than an alerting or forecasting tool. Cost allocation tags are essential for attributing costs to teams or projects and improving reporting accuracy, yet they rely on downstream analysis and do not by themselves generate automated threshold notifications. IAM policies control who can perform actions on resources but do not provide budgeting forecasts or automated spend alerts; using IAM to block resource creation based on spend would be operationally complex and is not a native billing control.

Key Concepts

- **AWS Budgets:** A service to set cost or usage thresholds, track actual and forecasted metrics, and send notifications to stakeholders when thresholds are breached or projected to be breached.
- **Consolidated billing:** Combines charges into a single invoice for linked accounts, simplifying payment and volume discounts but not offering proactive alerting.
- **Cost allocation tags:** Labels applied to resources to attribute cost to owners, teams, or projects; useful for reporting and chargeback but not for automated budget enforcement.

Common Pitfalls

- Assuming consolidated billing will alert teams before overruns; it only aggregates costs for billing purposes.
- Treating tags as an alert mechanism; tags improve visibility but need Budgets or cost tools to generate notifications.
- Expecting IAM to act as a native budget control; IAM controls permissions, not billing forecasts or notifications.

14.10 — Rekognition for image/video analysis (D3.T3.7)

Domain 3 • Task 3.7 ↑ Back to Question 14.10

Select TWO use cases that are appropriate for a service that extracts printed or handwritten text from images or video frames for automated processing (for example, OCR and text search)

Options & Rationales

A ✓ — Automatically extracting line-item text from photographed receipts to populate expense forms.

Rationale: Extracting printed text from photos of receipts is a common OCR use case that supports automated data entry and search.

B ✗ — Flagging explicit visual content in images to prevent inappropriate material from being displayed.

Rationale: Content moderation assesses suitability of imagery; it is distinct from extracting printed or handwritten text.

C ✗ — Identifying objects such as chairs, trees, and buildings in a photo for inventory tagging.

Rationale: Object and scene identification finds visual items and activities rather than extracting written characters.

D ✗ — Determining a person's emotional state from a still image to personalize content.

Rationale: Emotion detection analyzes facial expressions, not text extraction, so it is not a text OCR use case.

E ✓ — Reading vehicle license plates from traffic camera frames to match plate numbers against a database.

Rationale: Detecting and extracting characters from images of license plates is a typical text-extraction scenario for automated matching.

Explanation

A text-extraction service (OCR) is designed to find printed or handwritten characters in images or video frames and return them as machine-readable text. The most natural use cases are situations where the written text itself is the important data: receipts, bills, forms, signs, or license plates that must be searched, indexed, or mapped into structured fields. Automatically pulling line-item text from photographed receipts so you can populate expense forms is a classic OCR workload. Reading license plates from traffic camera frames to match plate numbers against a database is another: the characters on the plate are the key information being extracted and processed.

The other choices describe visual recognition tasks that analyze the content of images without returning characters for automated text processing. Content moderation focuses on whether an image is appropriate, object detection finds objects and scenes such as furniture or buildings, and emotion analysis inspects faces to infer likely sentiment. These capabilities may work on the same underlying images, but they are not text extraction; they do not output the characters and words that an OCR pipeline needs for search, indexing, or downstream business logic.

Key Concepts

- **Text extraction (OCR):** Converts printed or handwritten characters in images into machine-readable text for search, indexing, or data entry.
- **Use-case fit:** OCR is best when the primary goal is to capture characters (receipts, forms, signs, license plates) rather than interpret faces or scene content.
- **Result handling:** Extracted text is typically returned with confidence metrics so applications can verify or post-process results.

Common Pitfalls

- Confusing visual recognition tasks (faces, labels, content moderation) with text extraction leads to choosing features that do not return characters for automated processing.

Glossary

- **OCR:** Optical character recognition, the process of converting images of text into machine-readable text.

14.11 — Elastic Disaster Recovery (DRS) use (D3.T3.6)

Domain 3 • Task 3.6 ↑ Back to Question 14.11

A company wants to keep an up-to-date copy of on-premises servers in AWS so that, if a primary site fails, recent data is available for quick boots of recovery hosts. Which AWS Elastic Disaster Recovery capability best meets this requirement?

Options & Rationales

A ✓ — Continuous block-level replication that streams ongoing disk changes to AWS for near-current copies of servers.

Rationale: This capability continuously copies disk-level changes from source servers to AWS, keeping recovery instances closely synchronized so they can start with minimal data loss.

B ✗ — Manual snapshot exports where administrators periodically copy full disk images to S3 and restore when needed.

Rationale: Periodic manual snapshots are not continuous and can result in larger gaps in data, increasing potential loss and recovery time.

C ✗ — Traffic routing with a Global Accelerator that redirects users to alternate regions during failures.

Rationale: Global Accelerator only manages network traffic direction; it does not replicate server disk state or keep copies of workloads.

D ✗ — Using read replicas in a managed database service to offload read traffic during peak load.

Rationale: Database read replicas handle database scalability and read availability but do not provide full server disk replication for failover of entire

servers.

Explanation

The scenario asks for a method to maintain an up-to-date copy of whole servers in AWS so recovery hosts can be launched with little data loss. The correct solution is the feature that continuously copies low-level disk changes from the source environment to AWS, producing near-current server images usable for rapid recovery. This approach minimizes the recovery point objective because changes are streamed as they occur rather than captured at infrequent intervals.

Continuous replication is appropriate because it operates at the block (disk) level and keeps the cloud-side copy synchronized with the source system. That allows automated recovery instances to boot with recent data when a failure occurs. Alternatives like periodic snapshot exports create time gaps between copies, increasing potential data loss and delaying recovery. Services focused on traffic routing or read-scaling do not address the need to preserve an entire server's disk state.

Why the correct choice fits. Continuous block-level replication directly matches the requirement to keep server disk contents current in AWS for quick recovery. It minimizes RPO by streaming changes rather than waiting for scheduled captures, and it supports launching recovery instances that reflect recent state. Manual snapshot approaches increase RPO and operational effort. Network routing features and read-replica patterns solve different problems and do not create full server disk copies necessary for the described failover scenario.

Key Concepts

- **Continuous block-level replication:** Ongoing copying of disk blocks from a source server to a target environment so the target mirrors recent storage state.
- **Recovery instances:** Virtual machines launched in the cloud from replicated data so workloads can resume after a site failure.
- **Recovery point objective (RPO):** The maximum tolerable amount of data loss, reduced by more frequent or continuous replication.

Common Pitfalls

- Assuming network routing or traffic management provides data replication; they only re-route users and do not synchronize storage.
- Confusing database read replicas or snapshots with full server replication; those solutions address specific needs but not whole-server, near-real-time copies.

Glossary

- **RPO:** Recovery Point Objective, the allowable data loss window.
- **Recovery instance:** A VM in the recovery environment launched from replicated storage.

14.12 — Scalability across multiple AZs (D3.T3.2)

Domain 3 • Task 3.2 ↑ Back to Question 14.12

A team wants a service that spreads incoming requests across instances in different AZs and avoids sending traffic to unhealthy servers. Which two capabilities of Elastic Load Balancing meet these requirements?

Options & Rationales

A ✗ — Auto Scaling — automatically adds or removes instances based on demand to maintain capacity across AZs.

Rationale: Auto Scaling adjusts capacity but does not itself perform request distribution or health-based routing; a load balancer is required to direct traffic.

B ✗ — Amazon S3 — stores objects across AZs to ensure availability and durability for static website content.

Rationale: Amazon S3 provides durable object storage across facilities but does not distribute incoming application requests or perform health checks on compute instances.

C ✓ — Network Load Balancer — balances TCP/UDP traffic across instances in several AZs and performs health checks to remove unhealthy nodes.

Rationale: Network Load Balancer (NLB) can span multiple Availability Zones to distribute network-level traffic and monitors target health to avoid routing to failing instances.

D ✗ — AWS Global Accelerator — optimizes global traffic paths to an endpoint, but it does not replace AZ-level health-based load balancing.

Rationale: Global Accelerator improves global network performance to endpoints, yet it relies on regional load balancing for distributing requests across AZs and health-checked instances.

E ✓ — Application Load Balancer — routes requests across targets in multiple AZs and checks target health before sending traffic.

Rationale: Application Load Balancer (ALB) supports distributing traffic across targets in multiple Availability Zones and uses health checks to ensure only healthy targets receive requests.

F ✗ — Route 53 latency routing — directs users to the Region with the lowest latency but does not perform health-based load distribution across instances within AZs.

Rationale: Route 53 can route traffic between Regions based on latency and health checks, but it does not balance requests across instances inside multiple Availability Zones like a load balancer does.

Explanation

A load balancing service is required to both spread incoming requests across compute resources in more than one Availability Zone and to avoid sending

traffic to unhealthy servers. Elastic Load Balancing provides multiple types of load balancers tailored to different traffic patterns: the Application Load Balancer operates at the application layer (HTTP/HTTPS) and supports routing to targets across multiple AZs with health checks, while the Network Load Balancer operates at the transport layer (TCP/UDP) and also spans AZs and uses health monitoring to stop routing to failed targets.

Why the correct choices fit. The Application Load Balancer is designed for HTTP/S workloads and performs health evaluations of targets so only healthy backends receive requests. The Network Load Balancer offers similar AZ-spanning and health-check capabilities for high-performance, low-latency network traffic. Both therefore satisfy the requirements to distribute traffic across AZs and prevent routing to unhealthy instances.

Why the incorrect choices do not fit. Auto Scaling changes the number of instances based on demand but does not itself route or perform health-based traffic distribution. Amazon S3 is durable object storage and is not a request router for compute instances. Route 53 can direct traffic between Regions and support simple health checks, but it does not perform fine-grained, AZ-level load distribution among instance targets the way a load balancer does. AWS Global Accelerator improves global path performance to endpoints but relies on regional load balancing for AZ-aware, health-based distribution.

Key Concepts

- **Load balancing types:** Different ELB types (Application, Network) serve distinct protocol needs and both can span multiple Availability Zones to improve capacity and resilience.
- **Health checks:** Load balancers use health probes to determine whether a target should receive traffic, preventing routing to unhealthy instances.
- **Availability Zones:** Spanning AZs reduces the impact of a single datacenter failure by distributing traffic across isolated locations.

Common Pitfalls

- Confusing scaling with distribution: Auto Scaling adjusts instance counts but does not replace a load balancer's routing and health-check responsibilities.
- Assuming DNS alone provides AZ-aware balancing: DNS routing can guide users to Regions but does not replace per-AZ target health checks and distribution.

Glossary

- **Elastic Load Balancing (ELB):** AWS service family that distributes incoming application or network traffic across multiple targets to improve availability and fault tolerance.
- **Health check:** A probe used by load balancers to verify that a target is functioning before routing traffic to it.

14.13 — Route 53 scalable DNS (D3.T3.5)

Domain 3 • Task 3.5 ↑ Back to Question 14.13

Which statements correctly describe how Amazon Route 53 health checks are used to influence DNS routing and availability?

Options & Rationales

A ✗ — Health checks encrypt DNS queries between users and the application endpoints to improve data security.

Rationale: Health checks verify endpoint health; they do not provide encryption of DNS queries or application traffic.

B ✗ — Using Route 53 health checks removes the need to run separate monitoring services such as CloudWatch.

Rationale: Health checks focus on endpoint reachability for routing decisions and do not replace broader monitoring and metrics services.

C ✓ — Health checks can be combined with routing policies to enable automatic failover between endpoints.

Rationale: Route 53 uses health check status with routing policies (for example, failover routing) so healthy endpoints receive traffic automatically when others fail.

D ✓ — Route 53 regularly probes endpoints and can stop sending DNS traffic to an endpoint that fails checks.

Rationale: Route 53 health checks actively monitor endpoints and, when an endpoint is marked unhealthy, DNS responses can be adjusted so traffic stops going to that endpoint (supporting failover).

E ✗ — Route 53 health checks require installing a software agent on each application server to report status.

Rationale: Health checks perform external probes (HTTP, HTTPS, TCP) and do not depend on installing agents on servers.

F ✗ — Health checks retain detailed historical performance and billing records for long-term cost allocation.

Rationale: Health checks provide current availability status and can trigger routing changes, but long-term performance and billing records are outside their purpose.

Explanation

Amazon Route 53 health checks are an availability mechanism that actively tests application endpoints and informs DNS routing choices. The primary role is to determine whether an endpoint is reachable and healthy; Route 53 can then stop returning that endpoint in DNS responses or shift traffic to alternatives when configured for failover.

Why the correct statements are right:

- The statement about regularly probing endpoints and stopping DNS traffic when checks fail describes the operational behavior of health checks: Route 53 makes periodic probes (for example, HTTP, HTTPS, or TCP) and marks endpoints unhealthy if probes fail, which influences which IPs or records are returned to users.
- The statement about combining health checks with routing policies reflects that health information is used by routing types such as failover routing so traffic is automatically directed to healthy endpoints.

Why the incorrect statements are wrong:

- Health checks do not encrypt DNS queries or application traffic; encryption is a separate concern (for example, TLS for application traffic or DNSSEC for DNS integrity), so claiming health checks provide encryption is incorrect.
- They are not intended to serve as long-term performance or billing records; their purpose is real-time availability detection, not cost allocation or archival metrics.
- Health checks complement, but do not replace, broader monitoring solutions. Comprehensive observability and operational metrics typically require dedicated monitoring services.
- Health checks are performed by external probes and do not require installing agents on application servers; agents would be a different monitoring approach.

Key Concepts

- **Health checks:** Periodic probes made by Route 53 to determine endpoint reachability and responsiveness used to decide DNS responses.
- **Failover routing:** A routing mode that uses health check status to direct traffic to a backup endpoint when the primary is unhealthy.

Common Pitfalls

- Confusing availability probes with security features (encryption) leads to incorrect assumptions about what health checks provide.
- Expecting health checks to replace full observability or long-term metrics can cause gaps in monitoring and incident postmortems.

14.14 — Cost optimization: rightsizing & demand match (D1.T1.2)

Domain 1 • Task 1.2

↑ Back to Question 14.14

A SaaS startup runs a stateless web tier on Amazon EC2 behind an Application Load Balancer across two Availability Zones. Instances are m5.large On-Demand with gp2 Amazon EBS volumes. Amazon CloudWatch shows average CPU near 20% with short weekday spikes to 60%. The team must lower costs this month without reducing resilience or doing a major refactor, and they want to stay on EC2. Which approach best aligns with rightsizing and matching

supply with demand?

Options & Rationales

A ✓ — Replace m5.large with m6g.large (Graviton), switch EBS to gp3, and use EC2 Auto Scaling target tracking at 50% CPU.

Rationale: Combines rightsizing and better price/performance (Graviton, gp3) with elasticity. Target tracking scales to load, reducing idle compute while preserving Multi-AZ resilience.

B ✗ — Purchase a 1-year Compute Savings Plan sized to current m5.large usage and keep a fixed Auto Scaling desired capacity.

Rationale: Reduces hourly rates but leaves overprovisioned, idle capacity (20% average). Fixed capacity does not adapt to spikes and does not rightsize first.

C ✗ — Upsize to m5.xlarge for headroom, switch EBS to gp3, and keep the Auto Scaling group at a fixed desired count.

Rationale: gp3 lowers EBS cost, but upsizing increases compute cost and keeps idle capacity. No elasticity to match demand.

D ✗ — Rebuild the web tier on AWS Lambda behind Amazon API Gateway to remove idle servers.

Rationale: Serverless scales per request, but this is a major re-architecture and replaces the ALB, which violates the stated constraints.

Explanation

Cost optimization at a foundational level relies on two principles: rightsizing and matching supply with demand. Rightsizing means selecting smaller, more efficient resource types and configurations that still meet performance needs. Matching supply with demand means using elasticity so capacity increases during spikes and decreases when idle, ensuring you pay only for what you use.

In the scenario, CloudWatch indicates low average utilization (about 20%) with brief spikes. This suggests overprovisioned compute most of the time. The most effective move is to improve price/performance and add elastic scaling. Replacing m5.large with Graviton-based m6g.large generally lowers cost and improves performance per dollar, and switching gp2 to gp3 reduces EBS cost while allowing performance to be provisioned independently. Enabling EC2 Auto Scaling with a target tracking policy (for example, 50% CPU) adjusts instance counts automatically based on CloudWatch metrics, so capacity grows for spikes and shrinks when idle. This preserves the Multi-AZ architecture and avoids a major refactor while delivering meaningful savings.

Choosing only a commitment-based discount (Compute Savings Plans) without changing capacity leaves the waste from idle instances. Savings Plans lower the rate you pay but do not rightsize or add elasticity by themselves; they are best applied to a stable, optimized baseline after you rightsize. Increasing instance size for headroom moves in the opposite direction and increases compute cost while still keeping idle capacity. Re-architecting to serverless (Lambda and API

Gateway) would remove idle infrastructure but conflicts with the requirement to avoid a major refactor and to remain on EC2.

Key Concepts

- **Rightsizing:** Use metrics (e.g., CloudWatch) and recommendations (e.g., Compute Optimizer) to choose smaller or more efficient instance families and storage (prefer gp3) that meet needs.
- **Matching supply with demand:** Use EC2 Auto Scaling target tracking to maintain a utilization goal so capacity scales up for spikes and down when idle.
- **Graviton-based instances:** Offer better price/performance for many workloads, lowering compute cost when supported by the application stack.
- **Savings Plans usage:** Apply to steady baseline after optimization; commitments do not fix overprovisioning and do not add elasticity.

Common Pitfalls

- Buying a Savings Plan before rightsizing, which locks in spend while idle capacity remains.
- Assuming an Application Load Balancer adds capacity automatically; ALB distributes traffic, but EC2 Auto Scaling is required to scale instances.

14.15 — Trusted Advisor security checks (D2.T2.4)

Domain 2 • Task 2.4 ↑ Back to Question 14.15

Which security area can AWS Trusted Advisor automatically inspect and report prioritized findings for to help reduce public access and misconfiguration risk?

Options & Rationales

A ✓ — S3 bucket permissions and object access settings

Rationale: Trusted Advisor evaluates S3 access controls and flags buckets or objects that allow public or overly permissive access, providing remediation suggestions.

B ✗ — VPC route tables and internet routing correctness

Rationale: Trusted Advisor does not focus on VPC route table contents; routing configuration is not a primary Trusted Advisor security check.

C ✗ — CloudWatch metric thresholds and alarm definitions

Rationale: CloudWatch metrics and alarm tuning are monitored by CloudWatch and related services, not a core Trusted Advisor security check.

D ✗ — Application-level code vulnerabilities found in deployed apps

Rationale: Trusted Advisor does not perform static or dynamic application security testing; it focuses on infrastructure configuration checks, not app code scanning.

Explanation

AWS Trusted Advisor runs automated checks that look for common infrastructure misconfigurations and then gives prioritized, actionable recommendations. One important inspection area is object storage access: Trusted Advisor analyzes access controls and policies attached to buckets and objects and highlights resources that permit public or overly broad permissions, helping reduce accidental data exposure. The other options represent monitoring or security activities that are outside Trusted Advisor's core checks: route table correctness is a network configuration detail, metric thresholds belong to CloudWatch, and application code scanning requires specialized security testing tools.

Key Concepts

- **Trusted Advisor security checks:** Automated inspections that find misconfigured resources and present prioritized remediation for common security controls.
- **S3 access control:** Bucket and object policies, ACLs, and public access settings determine who can read or write objects; misconfiguration can expose data.
- **Scope of tooling:** Infrastructure configuration checks differ from runtime monitoring (CloudWatch) and from application security testing; each uses different AWS tools.

Common Pitfalls

- Confusing infrastructure checks with application security scanning leads to selecting tools that do not provide the needed findings.
- Assuming Trusted Advisor manages runtime alarms or detailed network routing analysis; it instead focuses on high-level best-practice checks such as storage permissions.

Glossary

- **S3:** Amazon Simple Storage Service used to store objects and data.
- **Trusted Advisor:** AWS service that inspects accounts for cost, performance, security, and fault-tolerance best practices and returns recommendations.

14.16 — Data transfer pricing within/across Regions (D4.T4.1)

Domain 4 • Task 4.1 ↑ Back to Question 14.16

Which statement best describes typical AWS billing for moving data between geographically separate Regions?

Options & Rationales

A ✗ — Data transferred within a single Availability Zone is more expensive than cross-Region transfers.

Rationale: Traffic inside the same Availability Zone is commonly free or lower cost; it is not usually more costly than cross-Region transfers.

B ✗ — All data moving anywhere in AWS is billed at the same flat per-GB rate regardless of source or destination.

Rationale: AWS differentiates pricing by where data moves (for example, within an AZ, between AZs, between Regions, or to the internet) rather than using one uniform rate.

C ✓ — Data sent between two different AWS Regions usually incurs higher per-GB charges than traffic kept within the same Region.

Rationale: Inter-Region data transfer typically has higher per-gigabyte pricing compared to transfers that remain inside a single Region, making cross-Region movement more expensive.

D ✗ — Sending data from AWS to the public internet is generally cheaper than moving it between Regions.

Rationale: Egress to the public internet is normally billed and can be costly; it is not typically cheaper than inter-Region transfer.

Explanation

AWS categorizes data movement by location to reflect the underlying network and operational costs. Transfers between separate geographic Regions cross longer network paths and additional infrastructure, so these cross-Region transfers are priced higher per gigabyte than traffic that remains inside the same Region. Keeping data within a Region can reduce transfer fees and latency.

Why the correct choice is right and others are wrong. The correct statement describes that transfers crossing Region boundaries generally incur higher per-gigabyte charges due to longer network paths and added infrastructure. The other statements are incorrect because they claim either that within-AZ transfers are more expensive, that all transfers share a single flat rate, or that internet egress is cheaper than inter-Region movement; each contradicts the way AWS differentiates data transfer pricing by location.

Key Concepts

- **Data transfer pricing tiers:** AWS applies different per-GB charges depending on where data flows (same AZ, between AZs, between Regions, or to the internet). This means cost varies by destination and path.
- **Inter-Region traffic:** Movement across Regions traverses broader network links and so typically carries higher per-GB fees compared with intra-Region transfers.

Common Pitfalls

- Assuming a single flat network rate: Believing all AWS data movement costs the same leads to underestimating bills when workloads replicate across Regions.
- Confusing intra-AZ and inter-Region costs: Traffic inside the same Availability Zone is usually cheaper (often free) than sending data between

Regions.

14.17 — Observability for operational excellence (D1.T1.2)

Domain 1 • Task 1.2 ↑ Back to Question 14.17

A web application is experiencing intermittent slow requests. Which two actions focus specifically on using end-to-end request traces to find the root cause?

Options & Rationales

A ✓ — Visualize trace spans to identify which downstream dependency adds the most time to requests.

Rationale: Span visualization highlights where time is spent in the request flow so teams can focus remediation on the component with highest latency.

B ✗ — Search application logs for error messages that match the slow request timestamps.

Rationale: Logs provide event detail and error context but lack the chronological, component-level timing of a single request trace.

C ✗ — Run synthetic tests from edge locations to measure global availability and response times.

Rationale: Synthetic monitoring measures availability and broad response times but does not reveal the internal call sequence and latency per component.

D ✗ — Aggregate CPU and memory metrics to detect overall resource exhaustion during slow periods.

Rationale: CPU and memory metrics indicate resource usage trends but do not show per-request component timing the way traces do.

E ✗ — Create a dashboard of average request counts per minute to identify traffic spikes.

Rationale: Request count dashboards show traffic volume trends but do not expose detailed per-request execution paths needed to trace latency sources.

F ✓ — Collect distributed traces to see latency across each microservice call in a single request path.

Rationale: End-to-end traces show how long each service or component takes within a single request, helping pinpoint slow segments causing user-facing latency.

Explanation

End-to-end request tracing captures the sequence and timing of operations that occur while handling a single user request. This approach produces a trace composed of spans—each span represents work performed by a service or component and includes timing information. By collecting and visualizing traces, engineers can see which spans contribute the most latency and which

downstream dependency or microservice is responsible for delays. Traces are therefore the primary tool for diagnosing intermittent slow requests where the problem depends on interactions between components in a single transaction.

Why the correct choices are right and others are wrong. Collecting distributed traces and visualizing spans directly target per-request timing and call sequencing, which is needed to identify which microservice or external dependency adds latency. Options that focus on resource metrics, logs, synthetic tests, or request counts address important observability signals but do not provide the chronological, component-level timing across a single request. Metrics and dashboards reveal trends and potential windows for investigation; logs supply contextual details once a suspect component is identified. Synthetic tests show external availability and latency from specific locations but do not trace internal service interactions. For intermittent slow requests caused by interactions between services, traces are the most effective initial diagnostic tool.

Key Concepts

- **Distributed tracing:** Records the path of a request as it traverses multiple services, breaking the request into spans with start and end times to reveal per-component latency.
- **Span visualization:** A trace view that orders spans and shows duration so teams can compare component latencies within the same request.
- **Metrics vs. logs vs. traces:** Metrics provide aggregated numeric trends over time, logs record immutable events and messages, while traces reveal the chronological, per-request interactions and timing across components.

Common Pitfalls

- Treating high-level metrics as sufficient for root-cause when the issue is within a single request path; metrics may indicate there is a problem but not where it occurs inside the transaction.
- Assuming logs alone will show component timing; logs can help after locating the suspect span but do not inherently map the end-to-end call sequence with timing.

Glossary

- **Trace:** A collected record of a single request's path through a distributed system composed of ordered spans.
- **Span:** A unit within a trace representing work done by a single component, including its duration and metadata.

14.18 — Economies of scale on AWS pricing (D1.T1.4)

Domain 1 • Task 1.4 ↑ Back to Question 14.18

Select TWO statements that describe how allocating large, upfront infrastructure costs across many customers affects cloud service pricing

over time

Options & Rationales

A ✓ — A broader subscriber pool lets the provider amortize capital outlays, enabling gradual price reductions for standard services.

Rationale: Amortizing capital outlays across many customers reduces per-customer cost and supports gradual decreases in published prices for commodity services.

B ✓ — Unit prices fall because initial hardware and data center expenses are recovered across a large customer base.

Rationale: Recovering capital expenditures from many customers lowers the cost allocated to each customer, enabling lower unit pricing over time.

C ✗ — Pricing increases are inevitable because larger scale requires proportionally more staff to manage services.

Rationale: Scale often enables automation and operational efficiencies that offset staffing increases; it does not automatically force price rises.

D ✗ — Spreading capital costs increases variability of monthly bills for each customer because of uneven allocation.

Rationale: Allocating fixed costs across many customers typically stabilizes and reduces per-unit cost rather than making bills more variable.

E ✗ — Individual customers see immediate discounts only when they commit to long-term contracts, unrelated to shared capital costs.

Rationale: Long-term commitments can lower price for that customer, but they are a separate pricing model and not the same effect as distributing fixed infrastructure costs.

F ✗ — Higher per-customer fixed costs mean customers always pay more as usage grows.

Rationale: This reverses the usual effect: spreading fixed costs over more users normally reduces per-customer cost rather than increasing it.

Explanation

When a cloud provider builds data centers and purchases hardware, those are large upfront capital expenses. Rather than charging one customer for an entire investment, the provider distributes that expense across many subscribers. This distribution lowers the incremental cost assigned to each customer for the same capacity. Over time, as the customer base grows and the initial investments are recovered, the provider can reduce the unit price for common services.

Why the correct options are right. The two correct statements describe the same core economic effect: allocating capital expenditures across a large user base reduces the per-customer share of those fixed costs. Saying unit prices fall because the provider recovers hardware and facility costs from many customers captures the basic mechanism. Stating that a broader subscriber pool enables

amortization of capital outlays and gradual price reductions expresses the same idea in business terms.

Why the other options are wrong. One distractor claims per-customer fixed costs rise with usage growth; this is incorrect because spreading fixed costs across more customers reduces, not increases, each customer's share. Another suggests scale necessarily increases prices due to proportional staffing needs; in reality, scale commonly allows automation and efficiency gains that offset staffing growth. A choice that ties immediate discounts to long-term contracts describes a different pricing mechanism (commitment discounts) rather than the effect of distributing capital costs. The final distractor claims spreading capital costs increases billing variability, but allocating fixed costs across many customers usually stabilizes and lowers per-unit pricing.

Key Concepts

- **Capital amortization:** Converting a large upfront investment into smaller periodic cost allocations across customers so each pays a fraction of the total over time.
- **Per-unit cost reduction:** As the number of customers grows, the fixed cost assigned to each unit of service declines, enabling lower prices for standard offerings.
- **Difference from commitment discounts:** Discounts from long-term or reserved commitments are a separate pricing approach and do not explain the general price decline from spreading fixed infrastructure costs.

Common Pitfalls

- Confusing provider-level cost distribution with customer-specific discounts leads to incorrect answers.
- Assuming scale always raises staffing costs ignores automation and efficiency gains that reduce per-unit operational expense.

Glossary

- **Amortize:** To spread the cost of a capital expense over a number of periods or customers.
- **Unit price:** The cost charged for a single measurable amount of service (for example, per GB or per compute hour).

14.19 — Athena for interactive S3 querying (D3.T3.7)

Domain 3 • Task 3.7 ↑ Back to Question 14.19

A startup stores 24 TB of application logs as raw CSV files in a single Amazon S3 prefix. Analysts use Amazon Athena; most queries filter by event_date and select only a few columns. Queries are slow and the Athena bill is high. The team wants to significantly reduce the data scanned per query while remaining serverless. What should they do?

Options & Rationales

A ✓ — Convert the dataset to Apache Parquet with Snappy compression and partition the S3 data by year/month/day; update the AWS Glue Data Catalog.

Rationale: Columnar, compressed Parquet plus date-based partitioning enables column and partition pruning, so Athena scans far less data, reducing cost and improving performance.

B ✗ — Enable Amazon S3 SSE-KMS on the bucket and encrypt Athena query results.

Rationale: Encryption improves security but does not change how much data Athena scans, so it does not reduce query cost or latency.

C ✗ — Run an AWS Glue crawler to catalog the existing CSV files in the AWS Glue Data Catalog.

Rationale: Cataloging provides schema metadata but leaves raw, unpartitioned CSV. Athena would still scan most of the dataset, so cost and performance remain largely unchanged.

D ✗ — Use AWS Lake Formation to grant analysts access to only required columns and tables.

Rationale: Lake Formation provides fine-grained permissions, not scan reduction. Without changing file format or partitions, Athena still scans the same amount of data.

Explanation

Amazon Athena is a serverless interactive query service that reads data directly from Amazon S3 using a schema-on-read approach. Pricing is based on how much data each query scans, not on provisioned servers. The core optimization, therefore, is to reduce scanned data. The pricing relationship can be summarized as Estimated cost = Data scanned (TB) × Price per TB.

Variables used:

- **Estimated cost:** Calculated Athena charge for a query (currency)
- **Data scanned (TB):** Amount of data scanned from S3 for the query (in terabytes); using compressed, columnar formats reduces the scanned bytes
- **Price per TB:** Athena rate in the chosen Region (currency/TB)

Raw, unpartitioned CSV in a single S3 prefix is a common anti-pattern because Athena often must read the entire dataset to evaluate filters. Converting the data to a columnar, compressed format like Apache Parquet (for example, with Snappy compression) allows Athena to read only the selected columns. Partitioning the data by a frequently filtered key such as event_date (for example, year/month/day) enables partition pruning so only relevant date folders are read. Together, column pruning and partition pruning substantially reduce the amount of data scanned, which lowers cost and speeds queries. Updating the AWS Glue Data Catalog (often via a Glue crawler) ensures Athena knows the new schema and partitions.

By contrast, enabling Amazon S3 SSE-KMS and encrypting query results is a best practice for security but has no effect on the volume of data scanned. Running a Glue crawler on the existing CSV merely registers metadata; without converting format and adding partitions, Athena would still scan most files. Using AWS Lake Formation refines access control at the table or column level but does not alter file format or partition layout, so scan size and cost remain the same.

Key Concepts

- **Schema-on-read with Athena:** Define tables over data in S3; no servers to manage.
- **Pricing by data scanned:** Reducing scanned bytes directly reduces cost cost = scanned TB × rate.
- **Columnar formats (Parquet/ORC) with compression:** Enable reading only needed columns and fewer bytes.
- **Partitioning by date:** Partition pruning reads only relevant subsets, minimizing scans.

Common Pitfalls

- Leaving raw CSV in a single S3 prefix causes broad scans and higher cost.
- Assuming encryption or cataloging alone improves performance; they enhance security and metadata, not scan volume.

14.20 — Multi-Region choice for low latency (D1.T1.1)

Domain 1 • Task 1.1 ↑ Back to Question 14.20

Select TWO choices that are NOT valid reasons to select multiple AWS Regions to improve end-user latency.

Options & Rationales

A ✓ — Consolidates all user traffic through a single Region so every user accesses the same endpoint.

Rationale: Routing everyone to one Region increases distance for many users and raises latency, so centralizing traffic is not a step to reduce response times.

B ✗ — Enables independent regional scaling so local demand can be met with reduced latency.

Rationale: Scaling resources separately in different Regions lets capacity match local user needs, which helps keep latency low.

C ✗ — Ensures lower network round-trip times by hosting resources close to users.

Rationale: Placing resources nearer to users reduces network round-trip time and is a primary reason to use multiple Regions for lower latency.

D ✗ — Allows routing users to the Region that shows the lowest measured latency, improving user experience.

Rationale: Using measured latency to route traffic to the Regions that provide the best round-trip times is a valid way to improve user experience with a multi-Region architecture.

E ✓ — Provides guaranteed zero-latency during geographic failover events.

Rationale: Geographic failover can reduce downtime but cannot guarantee zero latency because network delays and recovery steps still occur.

Explanation

Choosing multiple geographic Regions is commonly used to bring compute and storage closer to users so network travel time is shorter, thereby improving perceived responsiveness. A valid strategy includes hosting resources in Regions nearer to user populations and using routing based on measured latency to direct users to the Regions that provide the best experience; scaling regionally to match local demand also helps keep response times low. However, the idea of centralizing all traffic into a single Region runs counter to the goal of lowering latency because many users would be farther from that centralized location. Similarly, geographic failover and redundancy improve availability but do not eliminate network delays; no architectural choice can guarantee zero latency during failover.

Why each choice is right or wrong. The statement that consolidates user traffic into one Region is not a reason to improve latency because it increases distance for many users. The statement about hosting resources close to users is a correct reason: proximity reduces round-trip time. Enabling independent regional scaling is also a correct reason because local capacity reduces queuing and network hops that add delay. Claiming guaranteed zero-latency during failover is incorrect: failover reduces downtime but cannot remove network propagation and recovery delays. Using latency-aware routing based on measurements is a correct operational practice to validate and achieve better user experience across Regions.

Key Concepts

- **Proximity reduces latency:** Locating resources physically closer to users shortens network paths and lowers round-trip times, improving responsiveness.
- **Latency measurements for routing:** Measuring delay in milliseconds provides an objective comparison between Regions and can be used with routing policies to send users to lower-latency Regions.
- **Regional scaling:** Independent scaling per Region lets capacity match local traffic patterns, which supports lower latency under load.

Common Pitfalls

- Assuming a single centralized Region will be fastest for everyone leads to higher latency for distant users.
- Expecting failover to eliminate latency ignores propagation and recovery

time; failover improves availability, not instant zero-delay performance.

14.21 — Economies of scale drive lower prices (D1.T1.1)

Domain 1 • Task 1.1 ↑ Back to Question 14.21

Which statement is NOT a way that economies of scale let AWS reduce customer unit prices?

Options & Rationales

A ✗ — Buying large quantities of hardware and negotiating better vendor pricing to lower per-unit costs.

Rationale: Bulk procurement reduces unit cost through volume discounts and supplier leverage; this supports lower customer prices.

B ✓ — Purchasing hardware in small quantities and keeping capacity isolated to drive up per-unit prices.

Rationale: This describes the opposite of economies of scale: small purchases and isolated capacity increase unit costs, so it does not explain how AWS lowers prices.

C ✗ — Pooling capacity across many customers so physical and virtual resources are shared and utilized efficiently.

Rationale: Sharing infrastructure spreads fixed costs over more users, raising utilization and lowering per-unit charges.

D ✗ — Automating provisioning and operations so fewer manual tasks are needed and operational cost per unit falls.

Rationale: Automation reduces labor and operational overhead, which lowers the cost to deliver each unit of service.

Explanation

Economies of scale lower unit prices by spreading fixed expenses and improving efficiency across many customers. Volume purchasing secures discounts from suppliers; pooling infrastructure increases utilization so capital and facility costs are amortized across more units; and automation reduces operational labor per unit. The incorrect statement describes buying in small quantities and isolating capacity, which increases per-unit cost and therefore violates the economies-of-scale principle.

Key Concepts

- **Volume purchasing:** Buying at scale often achieves supplier discounts and better contract terms, lowering unit hardware costs.
- **Capacity pooling:** Sharing physical and virtual resources across customers raises utilization and spreads fixed data-center costs.
- **Operational automation:** Automated provisioning and management reduce manual effort and lower ongoing operational cost per unit.

Common Pitfalls

- Confusing scale with complexity: assuming more customers always mean higher overhead rather than potential efficiency gains.
- Thinking automation increases cost; at scale it typically reduces per-unit operational expense.

14.22 — Migration tools: Application Migration, DMS (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 14.22

A company needs to move about 80 on-premises VMware VMs and a MySQL database to AWS within six weeks. They want to lift and shift the application servers to Amazon EC2 with minimal downtime, run test launches before cutover, and rightsize instances at launch. For the database, they want to migrate to Amazon RDS for MySQL using a full load followed by ongoing changes so the source stays online until cutover. Which TWO AWS services will perform these migrations with minimal downtime?

Options & Rationales

A ✗ — AWS Schema Conversion Tool (AWS SCT) — converts database schema and code objects for heterogeneous migrations.

Rationale: SCT helps convert schemas/code for heterogeneous moves but does not move data. For homogeneous MySQL-to-MySQL, SCT is unnecessary; DMS handles the data migration.

B ✗ — AWS Systems Manager Automation — orchestrate post-launch configuration and runbooks across EC2.

Rationale: Systems Manager can run post-launch actions integrated by MGN but is not the migration engine for servers or databases.

C ✓ — AWS Database Migration Service (AWS DMS) — full load plus CDC to Amazon RDS for MySQL with minimal downtime.

Rationale: DMS performs a full load then change data capture to keep the source online until cutover, enabling a low-downtime migration to RDS for MySQL.

D ✗ — AWS Migration Hub — centralized visibility and status tracking across migrations.

Rationale: Migration Hub provides unified tracking and status but does not perform server or database migrations.

E ✗ — Amazon S3 as an intermediate target for the database — use object storage as a landing zone during migration.

Rationale: While DMS can target S3 for analytics offloading, this scenario migrates directly to RDS for MySQL with CDC. S3 is not required and adds unnecessary steps.

F ✓ — AWS Application Migration Service (MGN) — agent-based, continuous block-level replication with test launches and rightsizing to EC2.

Rationale: MGN is the primary lift-and-shift tool for physical and virtual servers to EC2. It continuously replicates, supports test launches, and enables rightsizing at cutover.

Explanation

Successful AWS migrations often combine a rehost path for servers and a database path that minimizes downtime. For servers, the purpose-built service is AWS Application Migration Service (MGN). MGN installs a lightweight agent and performs continuous block-level replication into a low-cost staging area. It lets teams launch test instances to validate cutover readiness and automatically handles driver and configuration conversion for Amazon EC2. At launch, MGN supports rightsizing of EC2 instance types and Amazon EBS volumes and can trigger post-launch actions via AWS Systems Manager. This approach preserves the original OS and application stack while reducing downtime and risk compared to manual rebuilds.

For relational databases, AWS Database Migration Service (AWS DMS) is designed to migrate data with minimal downtime. A typical DMS task performs a full load of existing data and then enables change data capture (CDC) to replicate ongoing changes while the source remains online. DMS supports homogeneous migrations, such as MySQL to Amazon RDS for MySQL, and heterogeneous migrations when paired with AWS Schema Conversion Tool (AWS SCT) for schema and code translation. In this scenario—moving MySQL to RDS for MySQL—DMS alone is sufficient to execute the data migration with CDC and meet the low-downtime requirement.

Services like AWS Migration Hub provide centralized visibility and tracking across tools but do not perform the migrations. AWS Systems Manager can run post-launch scripts that MGN invokes, but it is not a migration engine. Although DMS can target Amazon S3 for analytics offloading, the requirement here is a direct migration to RDS with CDC, so introducing S3 would add complexity without benefit.

Key Concepts

- **AWS Application Migration Service (MGN):** Agent-based, continuous block-level replication with test launches and rightsizing for lift-and-shift to EC2.
- **AWS Database Migration Service (DMS):** Full load plus CDC keeps the source online, enabling minimal-downtime migrations to Amazon RDS.
- **Homogeneous vs. heterogeneous:** Homogeneous (e.g., MySQL to RDS for MySQL) does not require AWS SCT; heterogeneous requires SCT to convert schema/code.
- **Migration Hub's role:** Central tracking and status, not the execution of server or database migrations.

Common Pitfalls

- Selecting AWS SCT for a homogeneous MySQL-to-MySQL move or assuming it performs data transfer. SCT is only for schema/code conversion in heterogeneous migrations.
- Choosing Migration Hub expecting it to migrate resources; it only provides visibility and tracking, not replication or cutover.

14.23 — Marketplace third-party security products (D2.T2.4)

Domain 2 • Task 2.4 ↑ Back to Question 14.23

Your team is evaluating procurement of third-party security tools through AWS Marketplace. Select TWO statements that represent anti-patterns or incorrect assumptions about using Marketplace for security products.

Options & Rationales

A ✓ — A product bought from Marketplace will be fully pre-configured by AWS so no customer setup or tuning is required.

Rationale: This is a violation: Marketplace simplifies procurement and deployment options but does not replace customer responsibility for configuring and tuning security controls.

B ✓ — Purchasing a security product from Marketplace removes the need to track who is billed; cost allocation and tagging are unnecessary.

Rationale: This is a violation: while Marketplace can consolidate charges, customers remain responsible for cost allocation and tagging for visibility and accountability.

C ✗ — Marketplace can consolidate licensing and billing for third-party security solutions into a single AWS invoice.

Rationale: This is accurate: Marketplace commonly offers consolidated billing and license procurement for third-party offerings.

D ✗ — Marketplace offers a catalog of network and endpoint security products that organizations can evaluate and purchase through AWS.

Rationale: This is accurate: Marketplace centralizes access to third-party security offerings for evaluation and purchase.

E ✗ — Vendors in Marketplace may provide automated deployment methods to speed installation, but customer configuration is still needed.

Rationale: This is accurate: many listings include deployment automation, yet customers still perform final configuration and validation.

F ✗ — Using Marketplace can simplify licensing terms compared with negotiating separate vendor contracts outside AWS.

Rationale: This is accurate: Marketplace often streamlines license procurement and management compared to separate agreements.

Explanation

AWS Marketplace is primarily a procurement and billing integration platform. It provides a catalog of third-party products, standardizes how customers subscribe and pay for those offerings, and often consolidates licensing and usage charges onto the AWS bill. For security tools, Marketplace can simplify how organizations discover solutions, evaluate pricing, and purchase licenses, but it does not change who is responsible for configuring, tuning, and operating those tools safely.

The incorrect assumptions in this question treat Marketplace as if it were also an operations and governance service. Expecting AWS to fully pre-configure a security product so that no customer setup is needed ignores the shared responsibility model: customers must still integrate the tool into their environment, define policies, and validate that protections match their risk posture. Likewise, assuming that consolidated billing removes the need for tagging, chargeback, or cost allocation would erode financial accountability. Even when charges appear on a single AWS invoice, teams remain responsible for tracking which applications, business units, or owners consume which security products.

Key Concepts

- **Marketplace procurement:** Provides a catalog and billing integration to simplify buying and invoicing of third-party software, but does not transfer operational responsibilities.
- **Customer configuration responsibility:** After acquisition, the customer must configure, tune, and validate security products to meet organizational policies.
- **Cost visibility:** Consolidated billing aggregates charges, yet cost allocation (tags, chargeback reports) remains a customer activity for accountability.

Common Pitfalls

- Assuming consolidated billing removes the need for tagging or cost management causes blind spots in budgets and chargeback processes.
- Believing deployment automation replaces security validation leads to improperly tuned or insecure configurations.

14.24 — Firewall Manager centralizes WAF rules (D2.T2.4)

Domain 2 • Task 2.4 ↑ Back to Question 14.24

A company wants a single place to push and enforce consistent firewall and DDoS protection rules across many AWS accounts in its organization. Which AWS capability best meets this need?

Options & Rationales

A ✗ — A web application firewall you configure separately in each account for application-layer HTTP/S rules.

Rationale: A WAF provides HTTP/S protections but requires per-account or

per-resource configuration; it does not by itself centralize policy deployment across many accounts.

B ✓ — A central service that deploys and enforces protection policies across multiple accounts in an Organization.

Rationale: This describes a centralized policy-management service that attaches organization-wide protection rules and reports compliance across accounts.

C ✗ — A per-instance virtual firewall applied only to individual Elastic Network Interfaces.

Rationale: Instance-level virtual firewalls control traffic to specific network interfaces but are not a tool for centrally pushing policies across an organization.

D ✗ — A monitoring audit service that only records API activity for later review without enforcing rules.

Rationale: An auditing service captures actions and events for visibility, but it does not deploy or enforce network protection policies across accounts.

Explanation

The requirement is a managed capability that lets an administrator push consistent network and application protection rules to many accounts from a single control point and track compliance. The correct choice describes a service that centrally manages and enforces protection policies across an AWS Organization and provides compliance reporting and remediation tracking.

Why the correct choice fits. Centralized policy management allows administrators to define protection rules once and apply them across multiple accounts and resources in the organization. This reduces administrative effort, ensures consistent security posture, and produces aggregated compliance status so gaps can be identified and addressed.

Why the other choices are incorrect. The choice describing a web application firewall represents an HTTP/S protection tool that must be configured for each resource; it does not itself provide organization-wide policy deployment. The per-instance virtual firewall choice refers to controls bound to individual network interfaces, which cannot centrally push rules across accounts. The auditing-only choice refers to a service that records API calls and events for visibility and investigation but does not enforce protective rules.

Key Concepts

- **Centralized policy management**: Defining security or protection rules in one place and applying them across multiple accounts or resources to maintain consistency.
- **Per-resource vs. organization-wide controls:** Some protections must be set on each resource individually; organization-wide controls let administrators scale consistent settings across accounts.

- **Compliance reporting and remediation tracking:** Aggregated status information helps teams discover noncompliant resources and take corrective actions.

Common Pitfalls

- Confusing a protection enforcement tool with a monitoring-only service leads to expecting enforcement from a logging/audit product.
- Assuming per-resource firewalls automatically provide organization-wide consistency ignores the need for central policy deployment.

Glossary

- **Organization-wide:** Applied across accounts that belong to an AWS Organization.
- **Compliance reporting:** Aggregated view showing whether resources match defined policies.

14.25 — Free Tier eligible services & limits (D4.T4.2)

Domain 4 • Task 4.2 ↑ Back to Question 14.25

A startup just created a new AWS account to prototype a web API using two Amazon EC2 micro instances, Amazon S3, and AWS Lambda. They want to remain within the AWS Free Tier and receive an early warning before any overage charges occur. Which is the BEST solution?

Options & Rationales

A ✗ — Check AWS Cost Explorer at the end of each month to see which services consumed the Free Tier and adjust afterward.

Rationale: Useful for visibility, but retrospective; end-of-month review is too late to prevent charges.

B ✗ — Rely on Always Free benefits for AWS Lambda to guarantee no charges even if EC2 Free Tier hours are exceeded.

Rationale: Always Free applies to select allocations only; it does not protect EC2 Free Tier hours or other billable usage.

C ✗ — Stop nonessential instances and right-size resources without configuring alerts to stay within the monthly allowance.

Rationale: Good hygiene, but without alerts there is no proactive notification before billable overages occur.

D ✓ — Enable AWS Free Tier usage alerts and configure an AWS Budgets cost budget with email alerts at 80% (or $0) to notify when actual or forecasted charges approach the threshold.

Rationale: Combines service-specific Free Tier usage alerts with AWS Budgets forecasts to email early warnings at 80% or $0, enabling action before overages.

Explanation

AWS Free Tier includes three categories: 12-month Free Tier (activates on new accounts), Always Free (does not expire), and short-term Trials. Allowances are calculated per account per calendar month and typically aggregate across Regions. When usage exceeds included amounts, uses non-eligible options, or occurs in unsupported Regions, it is billed at On-Demand rates.

A common pitfall is running multiple EC2 micro instances continuously. The monthly Free Tier hours are shared; two micro instances running 24/7 can exceed the allowance, and attached EBS volumes and data transfer out may add costs. To avoid surprise charges, you need proactive notification before or as soon as spend begins.

The best approach is to enable AWS Free Tier usage alerts and set up an AWS Budgets cost budget with alerts. Free Tier usage alerts notify when service-specific usage approaches the included monthly allocations. AWS Budgets can alert on actual and forecasted charges at a threshold you choose (for example, 80% or $0) so you can act before overages accrue. Together, they provide early warning across services and costs, allowing you to stop nonessential resources, right-size, or prefer serverless before charges grow. Cost Explorer remains useful to confirm which services are consuming the Free Tier and to spot patterns, but it is primarily for analysis rather than real-time alerts.

A practical way to estimate potential overage is $C = \max(0, U - F) \times P$.

Variables used:

- **U:** your monthly usage (units)
- **F:** the monthly Free Tier allowance (same units)
- **P:** On-Demand unit price (per unit)

Why other choices fall short. Checking Cost Explorer at month-end is reactive and may be too late. Relying on Always Free does not protect EC2 Free Tier hours or other billable items. Right-sizing without alerts helps reduce risk but does not provide early notification.

Key Concepts

- AWS Free Tier categories: 12-month, Always Free, and Trials with different activation/expiration behaviors.
- Allowances are per account per calendar month and usually aggregate across Regions; excess usage is billed On-Demand.
- AWS Budgets alerts on actual/forecasted costs at thresholds (e.g., 80% or $0); Free Tier usage alerts track allowance consumption.
- Cost Explorer aids verification and trend analysis but is not a proactive notification mechanism.

Common Pitfalls

- Assuming Always Free prevents all charges while running multiple EC2 micro instances 24/7.
- Forgetting EBS and data transfer out can add costs beyond Free Tier allowances.

14.26 — Storage Gateway for hybrid storage (D3.T3.6)

Domain 3 • Task 3.6 ↑ Back to Question 14.26

A company is deploying AWS Storage Gateway for on-premises applications. To align with security best practices, which action should they prioritize?

Options & Rationales

A ✗ — Right-size the local cache and plan network bandwidth.

Rationale: These measures improve perceived performance and throughput but do not primarily address data protection or access control.

B ✗ — Enable lifecycle transitions to S3 Intelligent-Tiering.

Rationale: Lifecycle transitions optimize storage costs; they are not security controls for encryption or authorization.

C ✗ — Choose Volume Gateway and use EBS snapshots for backup.

Rationale: Volume Gateway with snapshots supports backup and disaster recovery, not encryption in transit/at rest or access control.

D ✓ — Use TLS in transit, SSE-KMS at rest, and IAM least privilege.

Rationale: TLS protects data in transit; SSE-KMS encrypts objects stored in Amazon S3; IAM least privilege limits access—together directly address data protection and access control.

Explanation

AWS Storage Gateway connects on-premises applications to durable Amazon S3 storage while keeping frequently accessed data in a local cache for low latency. The service secures data in transit with TLS and at rest in S3 using SSE-KMS, and access to S3 is governed by IAM. When asked about security best practices, the primary focus is protecting data confidentiality and enforcing least-privilege access.

Enabling TLS ensures that data moving between the gateway and AWS is encrypted in transit. Using SSE-KMS encrypts objects stored in S3 at rest with AWS Key Management Service–managed keys. Applying IAM least privilege restricts who and what can access the backed S3 data. These controls directly map to core security goals: encryption and authorization.

Other sensible actions in Storage Gateway target different objectives. Right-sizing the local cache and planning bandwidth are performance measures that influence throughput and user-perceived latency. Lifecycle transitions to S3 Intelligent-Tiering or archive classes are cost-optimization

techniques. Selecting Volume Gateway and taking EBS snapshots improves backup/disaster recovery posture. While all are valuable, they are not primarily security controls. Therefore, prioritizing TLS, SSE-KMS, and IAM least privilege best aligns with security best practices for Storage Gateway deployments.

Key Concepts

- **Encryption in transit and at rest:** TLS protects data on the wire; SSE-KMS encrypts S3 objects at rest with KMS-managed keys.
- **IAM access control:** Use least-privilege policies to limit who can access S3 data exposed via Storage Gateway.
- **Cache and bandwidth planning:** Right-sizing improves performance and user experience, not security.
- **Lifecycle transitions:** Moving objects to S3 Intelligent-Tiering reduces cost, not exposure to unauthorized access.

Common Pitfalls

- Confusing performance or cost features (cache sizing, lifecycle transitions) with security; only encryption and strict IAM policies directly address confidentiality and access control.

14.27 — Global infrastructure enables rapid reach (D1.T1.1)

Domain 1 • Task 1.1 ↑ Back to Question 14.27

A company wants to design a web application so a single physical data center failure does not cause an outage. Select TWO actions that use AWS Availability Zones to improve resilience

Options & Rationales

A ✗ — Use edge locations to run primary application servers so users experience lower latency worldwide.

Rationale: Edge locations are for content delivery and caching, not for hosting primary application servers; they don't replace Availability Zones for durability.

B ✗ — Run all components in a single Availability Zone to simplify networking and speed up replication.

Rationale: Consolidating in one zone simplifies architecture but increases risk of outage from a single zone failure and does not meet resilience goals.

C ✗ — Store backups only in the same Availability Zone as compute to reduce cross-zone data transfer costs.

Rationale: Keeping backups in the same zone creates a single point of failure; distributing backups across zones avoids loss if one zone fails.

D ✗ — Rely on a single Availability Zone but increase instance size to handle peak load during failures.

Rationale: Larger instances do not protect against a zone outage; availability requires distributing resources across multiple zones, not just scaling up.

E ✓ — Host the primary database in one Availability Zone and deploy a synchronous standby replica in a different Availability Zone.

Rationale: Keeping a standby replica in a separate Availability Zone provides failover if the primary zone becomes unavailable, improving availability for stateful services.

F ✓ — Place application servers in at least two Availability Zones and use a load balancer to distribute traffic across them.

Rationale: Running servers in multiple Availability Zones isolates against a single data center failure; a load balancer spreads requests so remaining zones continue serving users.

Explanation

Availability Zones are separate physical data centers within the same AWS region that operate independently so a problem in one does not directly affect others. Designing for resilience means placing critical resources across multiple Availability Zones so workload capacity and state can survive an outage in a single location. For stateless tiers, distributing application servers across zones and fronting them with a load balancer allows ongoing request handling when one zone fails. For stateful tiers, maintaining a replica or standby in another zone enables automated failover and protects data availability.

Why the correct choices are right and others are wrong. Distributing application servers and using a load balancer takes advantage of separate failure domains so traffic can be served from remaining zones if one fails. Deploying a synchronous standby replica in another zone protects database availability and supports automated failover. By contrast, keeping everything in one zone, storing backups only in the same zone, relying on larger instances, or misusing edge locations do not eliminate the single-point-of-failure risk that multi-AZ architectures are designed to address.

Key Concepts

- **Availability Zone separation:** Independent data centers within a region; faults in one zone are isolated from others.
- **Load balancing across zones:** Distributes incoming traffic to healthy instances in multiple zones, avoiding dependence on a single location.
- **Multi-AZ replication for databases:** Keeps a synchronized copy or standby in another zone so failover preserves service continuity.

Common Pitfalls

- Placing all resources in one zone or storing backups only in the same zone leaves the system vulnerable to a single failure. Scaling up instances inside a single zone improves capacity but not resilience. Confusing edge locations (which accelerate content delivery) with zones (which host core compute and storage) leads to incorrect architecture choices.

14.28 — Site-to-Site VPN encrypted connectivity (D3.T3.5)

Domain 3 • Task 3.5 ↑ Back to Question 14.28

A company is designing an encrypted VPN connection between its on-premises router and an AWS VPC. Which TWO statements about redundant VPN tunnels are correct?

Options & Rationales

A ✗ — Redundant tunnels replace IPsec with TLS for stronger encryption.

Rationale: Incorrect — redundant tunnels are additional IPSec tunnels; they do not change the encryption protocol to TLS.

B ✗ — Paired tunnels allow the VPN connection to span multiple AWS Regions automatically.

Rationale: Incorrect — the paired tunnels provide resilience for the connection to a specific VPC endpoint and do not create cross-Region connectivity by themselves.

C ✓ — Each VPN connection includes a pair of independent IPSec tunnels to provide resilience.

Rationale: Correct — AWS supplies two separate IPSec tunnels per Site-to-Site VPN connection so one can continue if the other fails.

D ✗ — Only one tunnel can be active at a time, so paired tunnels provide no benefit during maintenance.

Rationale: Incorrect — while failover behavior depends on routing and configuration, the purpose of paired tunnels is to maintain connectivity during failure or maintenance of one tunnel.

E ✗ — Redundant tunnels eliminate the need to configure routing on the on-premises gateway.

Rationale: Incorrect — routing must still be configured on the customer gateway so traffic switches to the healthy tunnel; redundancy does not remove routing configuration.

F ✓ — Having two tunnels improves availability by allowing automatic failover when one tunnel stops carrying traffic.

Rationale: Correct — paired tunnels enable failover so encrypted connectivity is maintained if one tunnel becomes unavailable.

Explanation

When using an AWS Site-to-Site VPN connection, each logical VPN includes two separate IPsec tunnels. These paired tunnels are independent network paths that provide resilience: if one tunnel becomes unavailable (for example, due to hardware or network failure), traffic can continue over the other tunnel so the encrypted connection remains available.

Correct choices describe the presence of two independent IPsec tunnels and the availability benefit from automatic or configured failover. Incorrect choices confuse redundancy with other changes: redundant tunnels do not remove the need for routing configuration on the customer gateway, do not change the encryption protocol from IPsec to TLS, and do not by themselves create multi-Region connectivity.

Key Concepts

- **Redundant VPN tunnels:** Two separate IPsec tunnels are provided per Site-to-Site VPN connection to improve resilience and fault tolerance.
- **Failover and routing:** Keepalive, route advertisements, or static routing determine which tunnel carries traffic; proper routing configuration on the on-premises gateway is still required to switch traffic to a healthy tunnel.

Common Pitfalls

- Assuming redundancy removes the need to configure on-premises routing or monitoring; customers still configure routing and should test failover.
- Confusing tunnel redundancy with changes to encryption protocols or with features that provide cross-Region connectivity; redundancy only provides multiple paths for the same encrypted connection.

14.29 — Managed DBs vs self-managed on EC2 (D3.T3.4)

Domain 3 • Task 3.4 ↑ Back to Question 14.29

A startup wants a managed database so they can recover from accidental data changes quickly without managing backup scripts. Which advantage of a managed database best matches this requirement?

Options & Rationales

A ✗ — Access to the underlying OS for full administrative control and custom backup tooling.

Rationale: While this enables custom backups, it requires the customer to manage scripts and maintenance, which contradicts the desire to avoid managing backups.

B ✗ — Ability to run on Spot instances to reduce costs for steady production workloads.

Rationale: Spot instances lower cost for interruptible compute but do not provide automated backup or time-based recovery features for databases.

C ✗ — Direct control of network ACLs for segregating traffic between subnets.

Rationale: Network ACLs help with traffic filtering and security, but they do not address backup automation or data recovery needs.

D ✓ — Built-in automated backups with time-based recovery to restore data to a previous moment.

Rationale: Managed services include automatic backups and allow recovery

to a prior point in time, matching the need to recover from accidental changes without custom scripts.

Explanation

Managed database offerings provide automated data protection features that reduce operational burden for customers. One core capability is scheduled and continuous backups performed by the service control plane, plus the ability to recover the database state to a previous timestamp (time-based recovery). This eliminates the need for customers to build, run, and monitor their own backup scripts.

Why the correct choice fits. The correct option describes automatic backups and recovery to a prior moment, which directly satisfies the startup's requirement to recover from accidental changes without managing backup tooling. Managed services handle the backup lifecycle, retention, and restoration process, making recovery predictable and simpler for teams with limited operational resources.

Why the other options are incorrect. The option about access to the underlying OS refers to self-managed setups where customers must create and maintain backup processes; that increases operational work rather than reducing it. The suggestion to use Spot instances focuses on cost-optimized compute that is interruptible and unrelated to backup automation. The network ACLs choice concerns traffic control and security boundaries, which do not provide backup or restore capabilities.

Key Concepts

- **Automated backups:** A service-run process that captures database snapshots and transaction history according to defined retention, reducing customer operational tasks.
- **Time-based recovery:** Restoring a database to its state at a specified timestamp, useful for undoing accidental deletes or data corruption.
- **Operational responsibility:** In managed databases, the provider performs lifecycle tasks (backups, retention, and restore mechanisms) while the customer uses the recovery features.

Common Pitfalls

- Assuming cost-focused features or network controls provide recovery capabilities is incorrect; backups and time-based restore are distinct service functions. Another mistake is confusing access to the OS with automated service features—having control increases work, not reduce it.

Glossary

- **Time-based recovery:** Restoring data to a chosen point in time to recover from unintended changes.
- **Managed database:** A provider-operated database service where routine maintenance tasks are handled by the service.

14.30 — When to require MFA (D2.T2.3)

Domain 2 • Task 2.3 ↑ Back to Question 14.30

A security team is updating an AWS sign-in policy to reduce account-takeover risk for high-privilege access across IAM and AWS IAM Identity Center. Which TWO actions are NOT recommended best practices for MFA?

Options & Rationales

A ✗ — Use hardware security keys (FIDO2) or virtual MFA devices for second-factor verification.

Rationale: Both hardware security keys and virtual MFA devices are recommended second factors.

B ✗ — Add IAM deny conditions so actions like cloudtrail:StopLogging, kms:ScheduleKeyDeletion, and ec2:TerminateInstances require aws:MultiFactorAuthPresent.

Rationale: Guarding sensitive actions with MFA conditions is a recommended defense to limit production-impacting or security-changing operations.

C ✗ — Always enable MFA for the AWS account root user.

Rationale: Enabling MFA for the root user is a must-do control to reduce the blast radius of account compromise.

D ✗ — Require MFA in a role's trust policy to use AWS STS AssumeRole for an admin role (condition: aws:MultiFactorAuthPresent).

Rationale: Conditioning role assumption on MFA is a recommended way to protect high-privilege, STS-based access.

E ✓ — Use long-lived access keys for privileged IAM users so they are not prompted for MFA.

Rationale: This is discouraged. Prefer short-lived, MFA-backed sessions over long-lived access keys, especially for privileged access.

F ✓ — Make MFA optional for the break-glass group to speed up incident access to Billing and account settings.

Rationale: Break-glass access should be tightly controlled; members must use MFA for Billing and account settings.

Explanation

Multi-factor authentication (MFA) adds a second verification factor to reduce account-takeover risk. In AWS, MFA should be enforced wherever the blast radius is high. That always includes the AWS account root user and high-privilege IAM roles and users who can change security controls, access Billing and Cost Management, or perform production-impacting actions. For federated or role-based administration, require MFA as a condition to assume high-privilege roles via AWS Security Token Service (STS). IAM condition keys such as aws:MultiFactorAuthPresent (and optionally aws:MultiFactorAuthAge) enable MFA enforcement for console sign-ins and temporary credentials from

AssumeRole or GetSessionToken. In AWS IAM Identity Center, configure MFA so users must verify a second factor at sign-in. For programmatic access, short-lived, MFA-backed sessions are preferred over long-lived access keys. Use hardware security keys (FIDO2) or virtual MFA devices as supported second factors. A tightly controlled, small break-glass group should exist for sensitive account and billing settings—and all its members must use MFA.

Under these principles, enabling MFA for the root user, requiring MFA in a role's trust policy to assume an admin role, guarding sensitive API actions with deny conditions unless aws:MultiFactorAuthPresent is true, and using hardware security keys or virtual MFA devices are all aligned to best practices. In contrast, relying on long-lived access keys for privileged users bypasses the intended protection of short-lived, MFA-backed sessions, and making MFA optional for a break-glass group weakens controls where risk is highest. Those two actions are not recommended.

Key Concepts

- **MFA enforcement points:** Apply MFA to the root user, high-privilege roles/users, sensitive actions, and Billing/account settings where compromise impact is high.
- **STS with IAM condition keys:** Use aws:MultiFactorAuthPresent (and optionally aws:MultiFactorAuthAge) to require MFA for AssumeRole or GetSessionToken sessions.
- **Short-lived vs. long-lived credentials:** Prefer short-lived, MFA-backed sessions over long-lived access keys for programmatic access.
- **Break-glass governance:** Keep the group small and require MFA to tightly control powerful account settings.

Common Pitfalls

- Assuming MFA is only for console access; STS-based role assumption should also require MFA for high-privilege roles.
- Believing incident response requires skipping MFA; break-glass access still needs MFA to prevent misuse when stakes are highest.

14.31 — AWS Backup for database backup/restore (D3.T3.4)

Domain 3 • Task 3.4 ↑ Back to Question 14.31

A company runs Amazon RDS, Amazon Aurora, and DynamoDB. They want a single place to define backup schedules, retention rules, and perform restores for these supported databases. Which is the best solution?

Options & Rationales

A ✓ — Use AWS Backup to centrally define policies and restore operations for the supported databases.

Rationale: AWS Backup provides a central service to configure backup schedules, retention, and restores across supported AWS database services,

enabling consistent policies and simplified recovery management.

B ✗ — Manually create snapshots and on-demand backups within each database service's console when needed.

Rationale: Manual backups per service work but are operationally heavy, inconsistent, and do not provide centralized policy enforcement or single-pane restore orchestration.

C ✗ — Deploy CloudFormation templates to create periodic snapshots for each database type and manage retention via template updates.

Rationale: CloudFormation can provision resources but is not intended as a centralized backup scheduler or restore orchestration tool and would require additional automation for lifecycle management.

D ✗ — Install a third-party backup agent on every database instance and manage schedules from the agent's console.

Rationale: Third-party agents may add complexity, operational overhead, and possible compatibility issues; they typically do not provide the native, centralized AWS-integrated backup orchestration.

Explanation

When an organization needs consistent backup schedules, retention policies, and the ability to recover supported AWS databases from a single place, a service that centralizes backup management is appropriate. AWS Backup is designed to coordinate backup policy creation, lifecycle (retention) settings, and restore actions for compatible AWS data stores, reducing manual steps and ensuring uniform application of protection rules across services.

Why the correct choice fits. AWS Backup centralizes policy definition and restore operations so administrators can apply one schedule and retention rule set to multiple supported database types, simplifying compliance and recoverability.

Why the other options are less suitable:

- Manually creating snapshots in each service's console requires repeated human action, increases risk of inconsistent retention, and makes auditing harder. It does not scale well.
- Using CloudFormation to provision snapshots treats infrastructure as code but is not a native backup schedule and restore orchestrator; it would need extra automation and still lacks a single restore workflow.
- A third-party agent adds integration and maintenance overhead and may not provide native AWS service-level orchestration or the same integration benefits.

Key Concepts

- **Centralized backup orchestration:** A single management point for schedules, retention, and restores across multiple data services to ensure consistency and reduce operational effort.

- **Retention policy:** Rules that determine how long backups are kept to meet recovery point objectives and compliance needs.
- **Restore orchestration:** Coordinated procedures that return data to a usable state from backups; central tools simplify and standardize this process.

Common Pitfalls

- Assuming templates or manual steps equal centralized management; they increase complexity and chance of configuration drift.
- Believing third-party agents always provide better integration; native AWS services typically offer tighter compatibility and lower operational overhead for AWS-managed databases.

Glossary

- **Backup schedule:** Timing and frequency settings for when backups occur.
- **Lifecycle/retention:** Policies that move or delete backups according to age or compliance rules.
- **Restore:** The process of recovering data from a backup to return to normal operations.

14.32 — Geo/industry compliance programs (D2.T2.2)

Domain 2 • Task 2.2 ↑ Back to Question 14.32

Which of the following statements about AWS Artifact is NOT correct?

Options & Rationales

A ✗ — It provides a central location to download AWS compliance reports and attestations.

Rationale: This is correct: AWS Artifact gives centralized access to AWS audit reports and certifications.

B ✗ — It lets customers obtain AWS security and compliance documentation for regulatory review.

Rationale: This is correct: Artifact supplies documentation (for example, SOC and ISO reports) used in regulatory assessments.

C ✗ — Access to its reports can help customers demonstrate AWS's controls during their compliance assessments.

Rationale: This is correct: Artifact reports help customers show how AWS's controls support their compliance efforts.

D ✓ — It is a tool that issues compliance certificates for a customer's cloud workloads.

Rationale: This is incorrect: AWS Artifact does not issue certifications for customer-managed workloads; it provides AWS's own reports and attestations, not customer certification.

Explanation

AWS Artifact is a self-service portal that gives customers on-demand access to AWS security and compliance documentation, such as SOC reports, ISO certifications, and other attestations. These documents describe how AWS designs and operates its own controls and are commonly used as evidence during customer audits or regulatory reviews. Customers can download these reports, share them with auditors, and reference them when mapping AWS controls to their own compliance requirements.

However, AWS Artifact does not certify customer workloads. It does not inspect an individual application or account and then issue a custom compliance certificate. Under the shared responsibility model, AWS is responsible for the security and compliance of the cloud infrastructure, while customers are responsible for how they configure and use services. Artifact provides evidence of AWS's controls, but customers must implement and document their own controls to demonstrate compliance for their workloads.

Key Concepts

- **Purpose of AWS Artifact:** Central, self-service access to AWS compliance reports and attestations used during audits and regulatory assessments.
- **Evidence for audits:** Artifact documents help customers show how AWS's controls support their own compliance programs but do not replace customer-level evidence.
- **Shared responsibility for compliance:** AWS provides compliant infrastructure and documentation; customers must configure services and processes to meet their obligations.

Common Pitfalls

- Assuming AWS Artifact can issue compliance certificates for a specific customer workload or account.
- Treating downloaded AWS reports as proof that an entire application or organization is compliant without implementing customer-side controls.

Glossary

- **AWS Artifact:** A portal that provides on-demand access to AWS security and compliance documentation, such as SOC and ISO reports.
- **Attestation:** A formal statement or report from an independent auditor describing how a provider's controls meet defined criteria.
- **Shared responsibility model:** The division of duties where AWS secures the cloud infrastructure, and customers secure and configure their applications and data in the cloud.

14.33 — Sustainability in Regions supports ESG (D1.T1.1)

Domain 1 • Task 1.1 ↑ Back to Question 14.33

A retail company wants to report how much its AWS-hosted applications

reduced greenhouse gas emissions for its annual sustainability disclosure. Which AWS approach best provides quantifiable emissions attribution for their workloads?

Options & Rationales

A ✓ — Use the AWS Customer Carbon Footprint Tool to measure and attribute emissions from the company's AWS usage.

Rationale: The Customer Carbon Footprint Tool provides estimated emissions tied to AWS usage and helps attribute reductions for reporting and governance.

B ✗ — Enable AWS CloudTrail and analyze API call logs to calculate the data center energy savings for the workloads.

Rationale: CloudTrail records API activity for auditing, but it does not provide emissions calculations or consumption-to-emissions attribution necessary for sustainability reporting.

C ✗ — Run Cost Explorer reports and map cost savings to carbon reductions for each resource to infer emissions avoided.

Rationale: Cost Explorer shows financial usage and costs; it does not translate usage into standardized emissions metrics for formal sustainability disclosure.

D ✗ — Use AWS Trusted Advisor checks to identify idle resources and assume removed resources equal direct emissions reductions.

Rationale: Trusted Advisor flags optimization opportunities but does not quantify greenhouse gas emissions; assuming idle-resource removal equals specific emissions reductions is not a validated measurement.

Explanation

The organization needs a reliable method to quantify and attribute greenhouse gas emissions associated with its AWS-hosted workloads for governance and reporting. The AWS Customer Carbon Footprint Tool estimates emissions tied to an account's cloud usage and attributes reductions over time, making it appropriate for sustainability disclosures. The other options record operational or cost information but do not provide standardized emissions figures.

Why the correct choice is best. The Customer Carbon Footprint Tool is specifically designed to estimate and attribute emissions from cloud consumption; it aggregates relevant usage and applies standardized factors so organizations can track improvements and include them in sustainability reports. This aligns with governance needs for measurable, repeatable metrics.

Why the other choices are incorrect. The option describing API activity logging records actions for audit and security purposes but lacks emissions conversion logic. The cost reporting approach provides monetary usage insight but does not supply validated greenhouse gas estimates. The recommendation to rely on optimization checks identifies wasteful resources but does not quantify the carbon impact and would produce unreliable disclosures if used alone.

Key Concepts

- **Emissions attribution:** Translating cloud resource usage into estimated greenhouse gas values using a tool designed for that purpose enables consistent reporting.
- **Audit vs. measurement:** Audit logs (API activity) are for security and operational tracing, not for converting usage into emissions.
- **Cost reporting vs. sustainability metrics:** Financial reports show spending patterns but do not equate to carbon metrics without conversion by a purpose-built estimator.

Common Pitfalls

- Confusing operational logs or cost reports with emissions measurement leads to unsupported claims in disclosures.
- Assuming that identifying idle resources directly equals a specific emissions figure without a validated estimator produces inaccurate reporting.

Glossary

- **Customer Carbon Footprint Tool:** An AWS tool that estimates greenhouse gas emissions associated with account-level cloud consumption to support reporting.
- **CloudTrail:** A service that records API calls for auditing and operational troubleshooting.
- **Cost Explorer:** A service that analyzes and visualizes AWS costs and usage.

14.34 — Boundaries of shared responsibility (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 14.34

Your security team finds sensitive customer files stored in an S3 bucket without server-side encryption enabled. Who is primarily responsible for enabling encryption for the objects stored in that bucket?

Options & Rationales

A ✗ — AWS — because AWS manages the physical storage hardware, it must encrypt customer objects automatically.

Rationale: AWS secures the underlying infrastructure, but customers are responsible for configuring encryption for their stored objects unless they choose a managed encryption option and enable it.

B ✗ — The customer's network team — they must encrypt data in transit before it reaches S3, so S3 object encryption is not needed.

Rationale: Encrypting data in transit is important and the customer controls it, but it does not replace the need to configure encryption at rest for objects in S3.

C ✗ — AWS Support — open a support case and AWS Support will enable encryption for all existing objects.

Rationale: AWS Support can advise, but it does not change a customer's data

configuration by default; enabling object encryption is a customer configuration task.

D ✓ — The customer — they must enable and manage encryption settings for objects they place in S3.

Rationale: Customers control data protection choices (such as choosing server-side encryption keys and bucket defaults) for objects they upload to S3. Amazon S3 now encrypts new objects at rest by default with SSE-S3, but customers remain responsible for ensuring that encryption configuration and key management meet their compliance requirements.

Explanation

Cloud providers operate and protect the physical systems, network, and managed service platforms. Customers control how they store and protect their data inside those services. When files are uploaded to object storage, decisions about encryption at rest and key management are set by the account owner or their administrators.

Why enabling encryption for S3 objects is the customer's responsibility:

- The infrastructure layer (servers, storage devices, and the service control plane) is operated by the cloud provider. That responsibility ensures the platform is maintained and physically secure.
- Configuration of how data is handled within a customer account — including choosing server-side encryption, supplying or using KMS keys, and applying bucket policies — is an account-level action that the customer must perform.

Discussion of the options. The correct choice describes that the customer must enable and manage encryption settings for their stored objects. This is accurate because encryption-at-rest settings and key selection reside in the customer's account control plane. The option that claims the provider will automatically encrypt customer objects misunderstands the boundary: while the provider secures the underlying systems, it does not change customer object settings by default. The suggestion that only encrypting data in transit removes the need for at-rest encryption confuses two distinct protections — both should be used when required. Finally, asking support to enable encryption assumes the provider will change data configuration automatically; support can guide but does not assume ownership of customer configuration unless explicitly contracted.

Key Concepts

- **Shared operational boundary:** The cloud provider maintains the physical infrastructure and managed service platform; the customer configures and protects their data and resource settings inside their account.
- **Encryption at rest vs. in transit:** Encryption in transit protects data during transfer; encryption at rest protects stored data. Both are complementary and must be configured appropriately by the customer.

- **Key management choices:** Customers may use provider-managed keys or bring/manage their own keys, but selecting and enabling those options is a customer task.

Common Pitfalls

- Assuming the provider will change or enforce data-level settings automatically for each customer account.
- Believing encryption in transit negates the need for encryption at rest.

Glossary

- **Encryption at rest:** Cryptographic protection applied to stored data so it is unreadable without the correct keys.
- **KMS (Key Management Service):** A managed service for creating and controlling cryptographic keys used to protect data.

14.35 — Data residency influences Region choice (D3.T3.2)

Domain 3 • Task 3.2 ↑ Back to Question 14.35

A financial services company must meet a strict data residency policy that requires all customer data, backups, and logs to remain in Germany. The team selected the EU (Frankfurt) Region and needs high availability and auditability. Which proposal would violate the residency requirement?

Options & Rationales

A ✓ — Enable Amazon S3 Cross-Region Replication from EU (Frankfurt) to EU (Paris) to improve disaster recovery.

Rationale: S3 Cross-Region Replication copies objects to another Region. Replicating to EU (Paris) moves data outside Germany, violating strict country-specific residency even if encrypted.

B ✗ — Deploy Multi-AZ Amazon RDS in EU (Frankfurt) for high availability while keeping data in-Region.

Rationale: Multi-AZ RDS replicates within a single Region across Availability Zones, preserving residency while improving resilience.

C ✗ — Send AWS CloudTrail logs to an S3 bucket in EU (Frankfurt) and keep CloudWatch Logs in the same Region.

Rationale: Keeping logs in EU (Frankfurt) ensures audit data remains in-Region, aligning with strict residency requirements.

D ✗ — Use single-Region AWS KMS keys in EU (Frankfurt) and restrict other Regions with an AWS Organizations SCP using the aws:RequestedRegion condition.

Rationale: Single-Region KMS keys keep key material in-Region, and the SCP blocks resource creation elsewhere to prevent drift.

Explanation

Data residency requirements specify the exact jurisdiction where regulated data must be stored and processed at rest. In AWS, Region choice determines where Region-scoped services (such as Amazon S3, Amazon RDS, Amazon EBS, and AWS CloudTrail logs) store customer data. AWS does not move customer content between Regions unless you explicitly configure replication. High availability can be achieved within a single Region by using Multi-AZ architectures, which preserve residency while improving resilience.

For strict country-specific residency (Germany-only), all copies of data—including backups, snapshots, and logs—must remain in the selected Region (EU (Frankfurt)). Encryption alone does not satisfy residency; encrypted replicas, logs, or snapshots stored in another Region still violate the requirement. To avoid drift, controls like AWS Organizations service control policies (SCPs) with the aws:RequestedRegion condition can deny resource creation outside approved Regions. For key management, using single-Region AWS KMS keys in the same Region avoids replicating key material across Regions.

With this in mind, enabling Amazon S3 Cross-Region Replication from EU (Frankfurt) to EU (Paris) intentionally copies data to another AWS Region. Because the requirement is Germany-only, sending data to EU (Paris) places regulated content outside the required jurisdiction and breaks residency, even if the data is encrypted. By contrast, Multi-AZ Amazon RDS improves availability within EU (Frankfurt) without crossing Regions; keeping CloudTrail and CloudWatch Logs in the same Region ensures audit data remains in-Region; and using single-Region KMS keys plus an SCP with aws:RequestedRegion both preserve residency and prevent inadvertent resource creation elsewhere.

Key Concepts

- **Data residency vs. encryption:** Residency dictates where data is stored; encryption does not permit storing copies outside the approved jurisdiction.
- **Region vs. Availability Zone:** Multi-AZ architectures stay within one Region, improving availability while maintaining residency.
- **Cross-Region features:** Services like S3 Cross-Region Replication intentionally copy data to other Regions and can violate strict residency.
- **Preventing drift:** An Organizations SCP with aws:RequestedRegion blocks resource creation in non-approved Regions.

Common Pitfalls

- Assuming encryption alone satisfies residency. Encrypted replicas or logs in another Region still violate strict country-specific requirements.
- Adding cross-Region DR by default. For strict jurisdictional policies, DR must remain in-Region unless the policy explicitly allows broader areas (for example, EU-wide).

14.36 — Protecting the root user (MFA) (D2.T2.3)

Domain 2 • Task 2.3 ↑ Back to Question 14.36

Your company uses a single AWS account. To reduce risk from compromise of the account's highest privilege identity, which single action is the best way to make accidental or unauthorized use of that identity significantly harder?

Options & Rationales

A ✗ — Create an administrative IAM user and use it for daily tasks instead of the root credentials.

Rationale: Using an admin IAM user for routine work reduces root usage but does not itself prevent unauthorized root access; protecting the root requires additional controls like MFA.

B ✗ — Store the root account password in a secure vault and share it only with senior staff.

Rationale: Vaulting credentials limits distribution but does not add a second authentication factor; if vault access or the password is leaked, the root remains vulnerable without MFA.

C ✓ — Require a hardware or virtual multi-factor authentication device for the account's root credentials.

Rationale: Adding a physical or time-based MFA for the root identity enforces possession-based verification before sensitive actions, greatly reducing risk even if the password is exposed.

D ✗ — Attach an IAM policy that denies sensitive actions unless a specific tag is present on requests.

Rationale: IAM policies cannot be attached to the root identity, so tagging-based denial does not reliably protect the root account from direct use.

Explanation

Protecting the highest-privilege account focuses on adding an additional authentication factor so possession is required in addition to a password. Multi-factor authentication (MFA) is a mechanism that requires something you know (password) and something you have (a hardware token or a time-based code from a virtual authenticator). Enabling a hardware or virtual MFA on the root identity ensures that an attacker who obtains the password cannot perform sensitive account actions without the second factor.

Why the recommended action is best. requiring hardware or virtual MFA directly raises the barrier to unauthorized use of the root identity and is specifically designed to protect high-privilege accounts. Creating an administrative IAM user is a good operational practice but does not by itself stop misuse of the root credentials. Storing the password in a vault limits exposure but does not add a second authentication factor; if the vault or password is compromised, the root remains at risk. Applying an IAM policy

that depends on tags is ineffective for the root identity because such policies cannot be attached to or govern the root in the same way they govern IAM principals.

Key Concepts

- **Multi-factor authentication (MFA):** Combines multiple verification types (knowledge and possession) so access needs both the password and a physical or time-based code.
- **Least privilege practice:** Reduce routine use of the root identity by using separate administrative identities for daily tasks, but still protect the root with stronger controls.

Common Pitfalls

- Assuming moving to an admin IAM user removes the need to secure the root; the root still exists and must be protected.
- Relying only on credential storage without adding MFA leaves a single point of failure.

14.37 — Security Hub centralizes findings (D2.T2.2)

Domain 2 • Task 2.2 ↑ Back to Question 14.37

What primary benefit does AWS Security Hub provide when it *normalizes* security alerts from multiple sources?

Options & Rationales

A ✗ — It automatically blocks malicious traffic at the network edge before reaching AWS resources.

Rationale: Security Hub does not perform network blocking; it aggregates and analyzes findings rather than acting as a network firewall.

B ✗ — It stores long-term forensic logs for legal retention and e-discovery by default.

Rationale: Security Hub focuses on findings and posture management; log retention for forensics is handled by services like CloudTrail or S3.

C ✓ — It converts varied tool alerts into a common format so teams can compare and act on them consistently.

Rationale: Normalization makes different alerts uniform, enabling consistent severity, metadata, and easier cross-tool comparisons for investigation and remediation.

D ✗ — It directly encrypts all customer data across AWS services using customer-managed keys.

Rationale: Security Hub does not perform data encryption; key management and encryption are responsibilities of services like KMS and the resource owners.

Explanation

Normalizing alerts means transforming security notifications from many tools into a single, consistent representation with comparable severity and metadata. This uniform view lets security teams quickly identify, prioritize, and correlate issues across accounts and products without interpreting each vendor's unique format. The correct choice describes this conversion and its operational benefit.

Why other choices are incorrect. The option about blocking traffic confuses detection/aggregation with enforcement; Security Hub aggregates findings but does not function as a network control or firewall. The long-term log storage option misattributes retention responsibilities — services such as CloudTrail and S3 handle logs and archival, not Security Hub's main role. The encryption option describes key management actions performed by KMS and individual services, which is outside Security Hub's purpose.

Key Concepts

- **Normalization:** Converting diverse alert formats into a shared structure so comparisons and automated processing are reliable.
- **Findings aggregation:** Collecting results from multiple sources into a consolidated view to support analysis and prioritization.

Common Pitfalls

- Mistaking aggregation tools for enforcement tools leads to expecting blocking or encryption actions from Security Hub. Its role is visibility and prioritization, not direct traffic control or default data encryption.

Glossary

- **Finding:** A recorded security issue with descriptive metadata and severity used to track and remediate problems.

14.38 — Resource-based vs identity-based policies (D2.T2.3)

Domain 2 • Task 2.3 ↑ Back to Question 14.38

A company stores reports in an Amazon S3 bucket in Account A. A data analytics application in Account B uses an IAM role to download specific objects via the AWS SDK. The team already attached an identity-based policy to that role allowing s3:GetObject on the bucket ARN, but requests still fail with AccessDenied. The application cannot be modified to assume a role in Account A, and the company does not want to create users in Account A.

What is the simplest change to enable cross-account reads?

Options & Rationales

A ✗ — Attach another identity-based policy to the IAM role in Account B allowing s3:GetObject on the bucket.

Rationale: Identity-based allows in Account B alone are insufficient. The resource owner (Account A) must also allow access; without a bucket policy (or

other allow), AccessDenied persists.

B ✗ — Create an IAM role in Account A with S3 read permissions and a trust policy for Account B; update the app to assume this role.

Rationale: Cross-account role assumption would work, but the constraint says the application cannot be modified to assume roles. It is not the simplest viable change here.

C ✓ — Add an S3 bucket policy in Account A granting s3:GetObject to the IAM role ARN from Account B on the required bucket/prefix.

Rationale: A resource-based S3 bucket policy can name the cross-account principal directly, enabling access without changing the application or creating new identities. If objects use SSE-KMS, ensure the KMS key policy also allows this principal.

D ✗ — Create an IAM user in Account A with S3 read access and provide long-term access keys to the application.

Rationale: This violates the stated constraint to avoid creating users in Account A and introduces additional credential management that is unnecessary for this scenario.

Explanation

Identity-based policies are attached to IAM users, groups, or roles and answer "what can this principal do." Resource-based policies are attached to the resource (for example, an Amazon S3 bucket policy) and answer "who can access this resource." For cross-account access, resource policies are especially useful because they can directly reference principals from other AWS accounts without requiring those principals to assume a role in the resource owner's account.

During authorization, AWS evaluates all applicable policies together. An allow in either an identity-based policy or a resource-based policy is effective only if there is no explicit deny in any policy type. For services like S3, access can be granted by either the caller's identity policy or the bucket policy; but when the resource is in a different account, the resource owner must permit the access, otherwise an identity-only allow in the caller's account still results in AccessDenied.

In this scenario, the application uses an IAM role in Account B to access an S3 bucket in Account A. Adding an identity-based allow in Account B alone does not grant cross-account access because the bucket owner must also allow it. The simplest solution is to add an S3 bucket policy in Account A that names the IAM role from Account B as the principal and grants s3:GetObject on the required keys. This meets the constraints: no app change to assume roles and no new users. If the objects are encrypted with a KMS key, the KMS key policy must also permit the same principal.

Creating a role in Account A for the application to assume would work in principle (its trust policy is a resource-based policy on the role), but the

application cannot be modified to perform role assumption. Creating a new IAM user in Account A contradicts the stated restriction and adds long-term credential management. Adding more identity-based permissions in Account B alone does not solve the resource owner's authorization requirement.

Key Concepts

- **Identity-based vs. resource-based policy:** Identity policies define allowed actions for a principal; resource policies define which principals can access a resource.
- **Cross-account access with resource policies:** Resource policies can directly list principals from other accounts, enabling access without role assumption.
- **Policy evaluation logic:** An allow in any applicable policy works only if no explicit deny exists across identity, resource, boundaries, or SCPs.
- **IAM role trust policy:** A special resource-based policy on a role that specifies who may assume it; separate from the role's permissions.

Common Pitfalls

- Changing only the caller's IAM policy and assuming that alone grants cross-account access to an S3 bucket owned by another account.
- Overlooking that the resource owner must explicitly allow access, and that explicit denies anywhere override allows.

14.39 — Migration Evaluator for business case (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 14.39

A company plans to move data-center servers to AWS and wants to identify which virtual machines can be right-sized to lower cost. Which Migration Evaluator capability directly provides measured CPU, memory, and I/O patterns to support sizing decisions?

Options & Rationales

A ✗ — Inventory analysis that collects system and dependency metadata.

Rationale: Inventory collection gathers configuration and relationship details but does not provide the time-series utilization metrics needed for right-sizing.

B ✗ — TCO estimation that models total cost across on-premises and cloud.

Rationale: TCO modeling estimates financial impact but relies on utilization data rather than providing the raw performance metrics itself.

C ✓ — Utilization profiling that captures CPU, memory and disk usage over time.

Rationale: Utilization profiling supplies actual resource consumption patterns, enabling accurate selection of instance sizes and elimination of over-provisioning.

D ✗ — Licensing assessment that evaluates software portability and license costs.

Rationale: License reviews determine cost and portability implications for software but do not measure server resource usage needed for sizing.

Explanation

When preparing migrations, reliable right-sizing depends on observed resource use rather than static inventories or cost models. The capability that captures time-series resource consumption — CPU, memory, and I/O — is specifically intended to reveal how busy a server is during typical operation. Those observed patterns let teams choose appropriately sized cloud instances, avoid paying for unused capacity, and set realistic performance expectations after migration.

Why the correct choice is right. The described capability directly delivers the performance metrics required to determine whether a VM is over-provisioned and which AWS instance family and size would be appropriate. This reduces risk of selecting instances that are too large (wasteful) or too small (performance issues).

Why the other choices are wrong. Inventory collection provides asset and dependency context but not time-based utilization numbers. TCO estimation focuses on cost comparisons and projections rather than raw usage metrics. Licensing review examines software entitlements and cost portability, which is important for migration planning but does not inform resource sizing.

Key Concepts

- **Utilization profiling:** Collects measured resource usage (CPU, memory, I/O) over time to reveal peak and average demand for each workload.
- **Right-sizing:** Matching cloud instance capacity to observed demand to reduce cost while maintaining performance.
- **TCO versus telemetry:** Cost models estimate financial impact, but telemetry provides the input metrics needed to make sizing decisions.

Common Pitfalls

- Confusing asset inventory with performance telemetry; inventory lists what exists, while profiling measures how resources are used.
- Assuming cost models alone can determine instance size without observed utilization patterns.

14.40 — X-Ray for tracing and debugging (D3.T3.8)

Domain 3 • Task 3.8 ↑ Back to Question 14.40

In AWS X-Ray, which feature provides a visual diagram that shows how services communicate and highlights latency between components for traced application requests?

Options & Rationales

A ✓ — A visual graph that displays service interactions and latency for traced requests.

Rationale: This describes the feature that maps services and shows latency between components for end-to-end traced requests.

B ✗ — A component that stores raw trace segment data for long-term archival and analysis.

Rationale: X-Ray captures segment and subsegment data, but long-term archival/storage is handled by separate services; this option misstates the mapping feature.

C ✗ — An automated detector that flags anomalies and suggests root causes for traces.

Rationale: Automatic anomaly detection and root-cause hints are provided by a different X-Ray capability, not the visual diagram of interactions.

D ✗ — A sampling controller that decides which requests are recorded to limit trace volume.

Rationale: Sampling controls trace collection rates to limit data volume, but it does not produce the interaction diagram showing services and latency.

Explanation

The described feature is the visual diagram in X-Ray that represents how application components call each other and displays latency between them for requests that were traced. This diagram helps developers and operators quickly identify slow connections, bottlenecks, or unexpected service dependencies when investigating request performance.

Why other choices are incorrect:

- Storing raw trace data long-term is not the visual diagram; archival or analytics beyond immediate tracing typically use other storage or analytics services. The mapping feature summarizes observed interactions rather than acting as a storage mechanism.
- Automated anomaly detection and root-cause suggestions are a separate X-Ray capability that analyzes traces for unusual patterns; it complements the visual diagram but is not the diagram itself.
- Sampling is the mechanism that limits which requests X-Ray records to control volume; it influences what appears in the diagram but does not produce the interaction visualization.

Key Concepts

- **Service interaction graph:** A depiction of services and edges representing calls; it summarizes relationships observed during traced requests.
- **Latency visualization:** The graph surfaces timing information so high-latency links or services can be identified visually.
- **Traced requests linkage:** Traces connect data from multiple services into a single view so the end-to-end path is clear.

Common Pitfalls

- Confusing the visualization with storage or analysis features leads to picking answers about data retention or anomaly detection instead of the graph purpose.
- Assuming sampling is the same as visualization; sampling affects visibility but does not create the diagram.

14.41 — Automation/serverless reduce ops cost (D1.T1.4)

Domain 1 • Task 1.4 ↑ Back to Question 14.41

A startup is re-architecting an image-processing workflow with the goals of paying only for actual usage, scaling automatically during spikes, and reducing manual operations. Uploads land in Amazon S3, and processing should run only when needed. Which TWO choices are anti-patterns that would undermine these goals?

Options & Rationales

A ✗ — Front the workflow with Amazon API Gateway invoking AWS Lambda so billing tracks requests and compute duration.

Rationale: API Gateway + Lambda is pay-per-request and duration, eliminating always-on instances and aligning spend to demand.

B ✓ — Use an Amazon EventBridge schedule to run a Lambda function every minute to poll S3 for new objects.

Rationale: Scheduled polling incurs invocations even with no uploads, breaking the pay-per-use, event-driven model.

C ✓ — Maintain a fixed fleet of Amazon EC2 instances sized for peak traffic and handle scaling with manual scripts.

Rationale: Pre-provisioned instances create idle cost and increase toil, opposing serverless and automation principles.

D ✗ — Use Amazon DynamoDB on-demand for metadata writes from AWS Lambda to avoid capacity planning and pay per request.

Rationale: DynamoDB on-demand scales automatically and charges per request, reducing operational overhead and idle cost.

E ✗ — Configure Amazon S3 event notifications via Amazon EventBridge to invoke AWS Lambda when new objects are uploaded.

Rationale: Event-driven invocation runs compute only on uploads, avoiding polling and minimizing idle charges.

F ✗ — Create Amazon CloudWatch alarms that start AWS Systems Manager Automation runbooks to retry failures and notify the team.

Rationale: Automated remediation reduces human effort and errors, supporting Operational Excellence with minimal overhead.

Explanation

Foundational serverless and automation patterns reduce both cost and operational effort by eliminating idle capacity and human-driven tasks. With services such as AWS Lambda, Amazon API Gateway, Amazon DynamoDB (on-demand), and AWS Fargate, pricing is consumption-based: you pay per request and for the compute time actually used, not for pre-provisioned servers. Event-driven integration (for example, Amazon S3 events routed via Amazon EventBridge) ensures processing only runs when an upload occurs, keeping idle cost near zero. Operational automation with Amazon CloudWatch and AWS Systems Manager Automation replaces manual runbooks, lowering toil and error rates.

A useful mental model for variable workloads is the cost relationship:

$$\text{Cost} = r_c \times t + r_q \times q$$

Variables used:

- r_c: compute rate ($ per GB-second for Lambda or $ per vCPU-second for Fargate)
- t: total compute time consumed (seconds)
- r_q: request rate ($ per request or invocation)
- q: number of requests in the period

To minimize spend when idle, you must reduce both compute time when there's nothing to process and the number of unnecessary invocations. Running a fixed pool of Amazon EC2 instances sized for peak demand keeps capacity allocated even when there are no uploads. That creates many hours of paid-but-idle compute and requires manual scaling scripts—contrary to the pay-per-use and automation goals. Similarly, invoking a Lambda function on a fixed schedule to poll Amazon S3 every minute generates invocations regardless of activity, increasing q and therefore cost without delivering value.

By contrast, placing Amazon API Gateway in front of AWS Lambda ties billing directly to traffic, and using Amazon DynamoDB on-demand removes capacity planning while charging per request. Configuring Amazon S3 event notifications via Amazon EventBridge triggers compute only on actual uploads, eliminating idle invocations. CloudWatch alarms that launch Systems Manager Automation runbooks provide automated retries and notifications, reducing human effort.

Therefore, the anti-patterns are maintaining a peak-sized EC2 fleet with manual scripts and scheduling Lambda to poll S3. Both undermine cost alignment and operational excellence compared to event-driven, automated serverless designs.

Key Concepts

- **Serverless pricing model:** Pay per request and duration; no cost for idle instances, which aligns spend to actual demand.

- **Event-driven architecture:** S3 events via EventBridge invoke processing only when needed, avoiding polling overhead.
- **Automation for operations:** CloudWatch with Systems Manager Automation reduces manual remediation and error risk.
- **Idle capacity avoidance:** Pre-provisioned fleets create fixed costs; variable, on-demand services convert spend to usage-based.

Common Pitfalls

- Assuming scheduled invocations are inexpensive enough to replace events; frequent schedules accumulate cost when idle.
- Keeping EC2 for "control" while overlooking idle hours and the manual work it introduces.

14.42 — 6 migration strategies: rehost → retain (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 14.42

A company needs to move dozens of virtual machines to AWS quickly with minimal changes to applications and operating systems. Which two actions align best with this migration approach?

Options & Rationales

A ✓ — Use AWS Application Migration Service to replicate servers and cut over to AWS with little alteration.

Rationale: AWS Application Migration Service is designed for bulk server replication and migrating VMs to AWS with minimal changes, matching a quick, low-change migration approach.

B ✗ — Refactor applications to use managed platform services and change application code for scalability.

Rationale: Refactoring involves redesigning applications and code changes to use cloud-managed services; this is more invasive than the minimal-change approach requested.

C ✗ — Recreate each application in a cloud-native architecture using microservices and serverless functions.

Rationale: Redesigning for cloud-native patterns is a deep modernization effort, not a minimal-change migration, so it does not match the quick lift-style approach.

D ✗ — Replace on-premises applications by subscribing to a new SaaS product and migrating users.

Rationale: Switching to a commercial SaaS offering is a product replacement strategy, not a minimal rehosting of existing servers and configurations.

E ✗ — Decommission unused systems and delete old data before migration to reduce migrated footprint.

Rationale: Retiring unused systems is a valid action in migration planning

but is a selective decommission step rather than the minimal-move strategy described.

F ✓ — Lift existing VMs into AWS as-is, keeping current OS, configurations, and application binaries unchanged.

Rationale: Moving virtual machines into the cloud without modifying their software or settings fits the minimal-change migration pattern focused on speed and reduced operational work.

Explanation

The scenario asks for the migration approach that moves existing virtual machines quickly with minimal modifications to software or configuration. The appropriate strategy focuses on relocating servers as they are into AWS, using tools that replicate server images and allow a fast cutover. Using a migration service built for server replication is consistent with this approach. Performing large code changes, adopting new SaaS products, or redesigning applications are deeper transformations and do not meet the requirement for minimal change.

Why the correct choices are right and others are wrong. Using a server replication service that handles many machines and enables cutover aligns directly with the requirement to move workloads quickly and with minimal alterations. Likewise, transferring VMs as-is (keeping existing OS and configuration) matches the low-change objective. Options that propose redesigning applications into microservices or refactoring to use managed platform services require code and architectural changes, which increase effort and do not support a fast, minimal-change move. Replacing systems with a new SaaS product changes ownership and functionality and is therefore not the described approach. Decommissioning unused systems is a preparatory or optimization action but is distinct from the core migration method requested.

Key Concepts

- **Minimal-change server relocation:** Moving virtual machines to cloud infrastructure without altering operating systems, middleware, or application binaries to reduce time and operational risk.
- **Bulk server replication service:** A service that copies running servers to AWS and orchestrates cutover, enabling rapid migration of many machines with little modification.

Common Pitfalls

- Confusing modernization with migration: Choosing to redesign or replace applications adds time and effort and contradicts the need for a quick, low-change move.
- Treating decommissioning as the same as migrating: Removing unused systems is useful for cost reduction but is not the migration method itself.

14.43 — EC2 pricing: On-Demand, Reserved, Spot (D3.T3.3)

Domain 3 • Task 3.3 ↑ Back to Question 14.43

A company runs a steady, 24x7 web/application tier whose instance attributes rarely change and also executes a nightly analytics batch that can tolerate being stopped and resumed. They want to lower EC2 compute costs while meeting reliability needs. Which TWO purchasing choices best align with these workloads?

Options & Rationales

A ✓ — Amazon EC2 Standard Reserved Instances (Regional, 1–3 years) for the steady 24x7 web tier; billing discount and instance size flexibility

Rationale: Standard RIs provide the highest RI discount when attributes are stable. Regional RIs apply an automatic billing benefit to matching usage and allow instance size flexibility within the family.

B ✗ — Bid above the On-Demand price on Spot to avoid interruptions and ensure availability

Rationale: No bidding is required today, and paying more does not prevent Spot interruptions. Spot capacity can be reclaimed with a 2-minute notice.

C ✗ — Purchase Zonal Reserved Instances for the batch jobs to guarantee capacity in a specific Availability Zone

Rationale: Zonal RIs add capacity reservation in one AZ—unnecessary for interruptible, non-steady batch. They are not the lowest-cost choice versus Spot for this use case.

D ✗ — Use Convertible Reserved Instances for batch jobs to switch instance families or OS during the term

Rationale: Convertible RIs are for long-running, steady usage when attribute changes are expected. They are not intended for intermittent, fault-tolerant batch workloads.

E ✓ — Amazon EC2 Spot Instances for nightly analytics; spare capacity at steep discount with 2-minute interruption notice

Rationale: Spot is ideal for fault-tolerant, flexible batch jobs. It uses spare capacity at up to ~90% discount but can be interrupted with a 2-minute warning.

F ✗ — Run the 24x7 web tier On-Demand to avoid commitment and scale instantly

Rationale: On-Demand fits new or unpredictable workloads but is not the most cost-effective for steady, always-on usage compared to RIs.

Explanation

Amazon EC2 offers three primary purchasing models that align to different usage patterns. On-Demand charges per running hour with no commitment—good for new, short-lived, or unpredictable usage. Reserved Instances (RIs) exchange

a 1- or 3-year commitment for a significant billing discount on matching usage. Spot Instances use spare EC2 capacity at steep discounts but can be interrupted with a 2-minute notice, so they are intended for fault-tolerant, flexible jobs.

For an always-on web/application tier with stable attributes, Standard Reserved Instances are the best fit to minimize cost. Regional RIs apply an automatic billing benefit to matching instances and offer instance size flexibility within the same family, making them well-suited when you do not require an Availability Zone capacity reservation. Performance is unchanged; RIs are a pricing benefit, not actual instances.

For a nightly analytics batch that can be stopped and resumed, Spot Instances are ideal. They deliver deep savings (up to ~90%) precisely because they can be interrupted. Batch, CI, analytics, and HPC jobs that checkpoint or retry easily are common Spot use cases. Using Spot here reduces cost while accepting the defined interruption model.

On-Demand for a 24x7 baseline is typically more expensive than RIs and does not meet the stated goal of lowering costs for steady usage. Zonal RIs add a capacity reservation in a specific Availability Zone, which is unnecessary for interruptible batch jobs and does not outperform Spot's cost for that use case. Convertible RIs provide flexibility to change instance families or OS during the term but are still designed for steady-state usage, not intermittent jobs. Bidding above On-Demand does not prevent Spot interruptions; no bidding is required today.

When comparing an RI to On-Demand, use the effective hourly cost:

$$C_{\text{eff}} = \frac{C_{\text{upfront}} + r \times h}{h}$$

Variables used:

- **C_{eff:** } Effective hourly cost [$/hour]
- **C_{upfront:** } Upfront payment [$]
- **r:** RI hourly rate [$/hour]
- **h:** Expected hours run in the term [hours]

A 24x7 workload drives h close to the total term hours, making RIs' effective rate lower than On-Demand.

Key Concepts

- **Reserved Instances (RIs):** Commitment (1–3 years) yields discounts; Standard maximizes savings when attributes are stable; Regional RIs give instance size flexibility within a family.
- **Spot Instances:** Spare capacity at steep discounts; can be interrupted with a 2-minute notice; best for fault-tolerant, flexible batch or analytics.
- **On-Demand:** No commitment, immediate scalability; best for new or unpredictable usage, not for steady 24x7 baselines.

- **Effective cost comparison:** Use C_{eff} to evaluate RI vs On-Demand over expected run hours.

Common Pitfalls

- Treating RIs as actual instances rather than a billing discount that applies to matching running instances.
- Assuming Spot interruptions can be avoided by bidding higher or that Spot suits availability-sensitive, always-on services.

14.44 — Lambda event-driven serverless scaling (D3.T3.3)

Domain 3 • Task 3.3 ↑ Back to Question 14.44

A media company uses Amazon S3 object-created events to invoke an AWS Lambda function that processes images and writes results to a downstream system. During marketing spikes, Lambda scales out quickly and the downstream system starts throttling. The team must prevent overload by limiting how many Lambda invocations run in parallel. They must keep the existing S3-to-Lambda integration and avoid adding new services. Which action is the MOST appropriate?

Options & Rationales

A ✗ — Enable provisioned concurrency on the Lambda function to keep execution environments warm and reduce cold starts.

Rationale: Provisioned concurrency reduces cold-start latency but does not restrict how many executions can run in parallel; it will not prevent downstream overload.

B ✗ — Increase the function's memory configuration to reduce execution time and lower the chance of throttling.

Rationale: More memory may shorten duration but does not cap concurrency. Faster processing can even increase request rate to the downstream system during spikes.

C ✗ — Insert an Amazon SQS queue between S3 and the function and use an event source mapping to batch and buffer events.

Rationale: SQS buffering with event source mappings can tune throughput, but it adds a new service and changes the integration—explicitly disallowed by the scenario's constraints.

D ✓ — Configure reserved concurrency on the Lambda function to cap its maximum concurrent executions and protect the downstream system.

Rationale: Reserved concurrency both guarantees capacity for the function and sets a hard upper limit on concurrent executions, directly limiting request fan-out to downstream systems.

Explanation

AWS Lambda is an event-driven, serverless compute service that automatically scales by creating concurrent function invocations as events arrive. With S3 object-created events invoking the function, traffic spikes can rapidly increase concurrency. If a downstream system cannot absorb the resulting request fan-out, you need a way to limit how many Lambda executions can run at once.

Reserved concurrency is the Lambda feature designed for this control. Assigning reserved concurrency to a function both guarantees a slice of capacity for it and, crucially for this scenario, sets a hard maximum on the function's concurrency. By capping maximum concurrency, the function's parallelism—and thus the request rate to the downstream system—is limited, protecting that system from overload. This satisfies the requirement without changing the existing S3-to-Lambda integration or introducing additional services.

Provisioned concurrency addresses a different problem: cold starts. It keeps execution environments initialized to minimize startup latency for latency-sensitive workloads. However, it does not limit how many invocations can occur at once. Using provisioned concurrency here would not mitigate downstream throttling and could even make ramp-up faster.

Increasing the function's memory may reduce per-invocation duration, but it does not provide any concurrency control. In some cases, shorter durations can increase overall throughput and worsen downstream throttling during spikes.

Introducing an Amazon SQS queue with an event source mapping is a valid pattern in general for streams and queues because event source mappings let you tune batching and parallelism, providing back-pressure. However, the scenario explicitly forbids adding new services or changing the S3-to-Lambda integration, making this option noncompliant with the constraints.

Key Concepts

- **Lambda concurrency-based scaling:** Each event can create a new concurrent invocation, enabling rapid scale-out during spikes.
- **Reserved concurrency:** Guarantees capacity and caps a function's maximum concurrency to protect downstream systems.
- **Provisioned concurrency:** Pre-initializes environments to reduce cold starts; it does not limit concurrency or request rate.
- **Back-pressure via queues:** Event source mappings for queues (e.g., Amazon SQS) control batching/parallelism, but adding a queue changes the architecture.

Common Pitfalls

- Confusing reserved concurrency (limits parallelism) with provisioned concurrency (reduces cold starts). This mix-up often leads to solutions that don't address downstream throttling.

14.45 — Post-migration optimization for cost/perf (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 14.45

A retail web application experiences large traffic spikes during promotions. The team wants capacity to increase and decrease automatically to meet demand while avoiding manual server changes. Which approach is the best fit?

Options & Rationales

A ✗ — Buy a fixed number of large reserved instances sized for peak traffic and use them throughout the year.

Rationale: Reserved instances lower cost for steady baseline usage but keep capacity fixed; they do not provide automatic scaling for unpredictable spikes, risking under-provisioning at peaks.

B ✓ — Configure Auto Scaling with scaling policies for the application fleet so instances launch and terminate based on load.

Rationale: Auto Scaling adjusts compute capacity automatically in response to demand, ensuring performance during peaks and reducing cost during low traffic by terminating unused instances.

C ✗ — Manually add or remove on-demand instances each time an expected promotion begins or ends.

Rationale: Manual changes work but require continuous human intervention and are error-prone for sudden traffic changes; they do not provide automated responsiveness.

D ✗ — Use a single very large instance to handle all peak traffic and accept idle cost between promotions.

Rationale: A single large instance avoids scaling complexity but wastes money when idle and represents a single point of failure, reducing resilience and cost efficiency.

Explanation

Auto scaling capability enables the environment to change compute capacity automatically in response to varying demand. For an application with unpredictable spikes during promotions, this approach provides elasticity—more instances are added when metrics indicate increased load and removed when traffic subsides—so performance is maintained while reducing unnecessary cost during low usage.

Why the other choices are not best. Buying fixed reserved capacity reduces hourly cost for predictable, steady workloads but does not expand or shrink capacity automatically when traffic spikes; this can cause insufficient capacity at peak times. Manually managing on-demand instances requires staff to act for each event, increasing operational overhead and risk of delayed response during sudden surges. Relying on one very large instance simplifies operations but creates a single point of failure and incurs high idle cost outside peak periods.

Key Concepts

- **Auto scaling:** Automatically adjusts compute resources based on defined metrics or schedules to match capacity with demand. This preserves performance during high load and lowers cost when demand falls.
- **Elasticity:** The ability of a system to expand and contract resources quickly and automatically to meet workload changes, improving both resilience and cost-effectiveness.
- **Cost-performance trade-off:** Static capacity (reserved or oversized single instances) can lower some costs or simplify architecture, but sacrifices responsiveness or leads to wasted spend during low-demand periods.

Common Pitfalls

- Assuming reserved capacity handles sudden increases without additional scaling measures can lead to under-provisioning at peak times.
- Believing manual scaling is sufficient for highly variable traffic overlooks the operational delay and risk of human error.

Glossary

- **Auto Scaling:** A mechanism that launches or terminates compute instances automatically based on rules or metrics.
- **On-Demand instance:** A compute resource billed hourly or per second without long-term commitment.
- **Reserved instance:** A billing discount applied to capacity reserved for a long term, appropriate for steady-state usage.

14.46 — Budget alerts for threshold breaches (D4.T4.2)

Domain 4 • Task 4.2 ↑ Back to Question 14.46

Which alerting method delivers a direct message to named people when a cost threshold is exceeded in an AWS budget?

Options & Rationales

A ✓ — Sending email alerts configured on the budget

Rationale: Budget-level email notifications send messages to specified addresses so stakeholders receive direct human-facing alerts when thresholds are crossed.

B ✗ — Triggering an automated IAM permission change

Rationale: Automated actions can modify permissions, but they do not directly send human-readable messages to individuals.

C ✗ — Publishing budget events only to an internal metrics store

Rationale: Writing events to a metrics store records data for analysis but does not directly notify people by message.

D ✗ — Relying solely on scheduled billing reports

Rationale: Periodic billing reports provide summary information later and are not immediate, direct alerts sent to named recipients when a threshold is

breached.

Explanation

To control costs, teams use budget alerts that notify people immediately when a spending rule is exceeded. A method that delivers a direct message to identified recipients is used to ensure humans are promptly informed and can take action.

Why the correct choice fits. Sending notifications by email from a budget is designed to reach specific users or mailing lists with a readable alert when a spend limit is crossed. This is a human-facing channel intended for stakeholders who need to respond quickly.

Why the other choices are incorrect. An automated permission change is an action that can enforce controls but does not itself provide a human-readable message to stakeholders. Recording events in a metrics store captures data for monitoring or dashboards but does not deliver immediate messages to named people. Scheduled billing reports are useful for periodic review but are not instant alerts triggered at the moment a threshold is crossed.

Key Concepts

- **Immediate human notification:** A method that sends messages directly to people so they can respond without inspecting dashboards.
- **Automated enforcement vs. notification:** Actions that change permissions or resources enforce policy programmatically, while notification channels inform stakeholders.
- **Event recording vs. alerting:** Storing cost events supports analysis; alerting delivers timely messages for operational response.

Common Pitfalls

- Confusing automated enforcement with alerting; enforcement may limit access but won't inform people unless paired with a notification channel.
- Assuming periodic reports substitute for real-time alerts; they are not triggered exactly when a threshold is crossed.

Glossary

- **Budget notification:** A configured message sent when a cost or usage threshold is reached.
- **Automated action:** A programmatic change applied when a budget condition is met (for example, adjusting permissions).

14.47 — Solutions Architects assist in design (D4.T4.3)

Domain 4 • Task 4.3

↑ Back to Question 14.47

A Solutions Architect is performing a structured architecture review. Which two outcomes are most likely from this review?

Options & Rationales

A ✗ — A fully implemented, production-ready infrastructure deployed into the account.

Rationale: Review activities identify improvements and guidance; they do not deploy production infrastructure themselves.

B ✓ — A report mapping the current design to best-practice principles with suggested improvements.

Rationale: Structured reviews evaluate designs against best-practice principles and produce a mapping with suggested changes.

C ✓ — A prioritized list of technical risks and recommendations to improve the design.

Rationale: Architecture reviews surface weaknesses and produce prioritized remediation guidance to reduce operational and reliability risk.

D ✗ — A signed project plan and timeline for delivering code-level changes across teams.

Rationale: Architecture reviews inform technical direction but do not serve as an executable project management plan.

E ✗ — A completed cost forecast that guarantees lower monthly charges if recommendations are applied.

Rationale: Reviews may include cost-optimization suggestions, but they cannot guarantee exact future billing outcomes.

Explanation

A structured architecture review is a consultative assessment that examines a solution's design against established best practices. The primary purpose is to identify where the current design may introduce operational, security, reliability, or performance weaknesses and to recommend prioritized improvements. Reviews produce findings and guidance rather than performing hands-on implementation or guaranteeing exact cost outcomes.

Why the correct choices fit. A prioritized list of technical risks and recommendations is a direct output of a review: the assessor documents issues, ranks them by severity or impact, and suggests remediation steps. A report that maps the current design to best-practice principles with suggested improvements is also expected: the review compares the architecture to reference practices and highlights gaps with recommended actions.

Why the incorrect choices are wrong. Producing a fully implemented, production-ready infrastructure is an execution activity outside the review scope; reviews inform what to build or change but do not perform the deployment. Creating a cost forecast that guarantees lower monthly charges is incorrect because reviews can recommend cost-saving actions but cannot promise exact billing outcomes—many variables affect final costs. Generating a

signed project plan and timeline is a project-management deliverable; while the review may influence planning, it does not replace a formal delivery plan.

Key Concepts

- **Architecture assessment:** Evaluates design against operational, security, reliability, performance, and cost best practices to identify gaps.
- **Prioritization:** Ranking findings by risk and impact helps teams focus on the most important changes first.
- **Guidance vs. execution:** Reviews provide recommendations and documentation; implementation, deployment, and guaranteed cost changes require follow-up work.

Common Pitfalls

- Assuming a review includes implementation: confusing assessment outputs with execution can lead teams to expect deployed changes without follow-up resources.
- Expecting precise billing guarantees: cost recommendations reduce risk of overspend but do not deterministically set future bills.

Glossary

- **Remediation:** Steps to correct a finding in an architecture review.
- **Best-practice principles:** Established guidance for secure, reliable, and cost-effective cloud architectures.

14.48 — Cost considerations for AI/ML analytics (D3.T3.7)

Domain 3 • Task 3.7 ↑ Back to Question 14.48

An analytics team runs AWS Glue ETL jobs in private subnets to read and write large datasets in Amazon S3. Finance reports unexpectedly high NAT Gateway charges during the ETL windows. The team wants the smallest change to eliminate these data transfer costs while keeping the subnets private (no internet access). What is the BEST solution?

Options & Rationales

A ✗ — Enable S3 Intelligent-Tiering with lifecycle policies on the data lake buckets.

Rationale: Optimizes storage class costs as access patterns change, but it does not address NAT Gateway data transfer charges incurred during S3 access from private subnets.

B ✓ — Add an Amazon S3 Gateway VPC endpoint and update private subnet route tables so ETL traffic to S3 stays private.

Rationale: Routes S3 access through a Gateway VPC endpoint, keeping traffic inside AWS and bypassing the NAT Gateway, which eliminates those data transfer charges without exposing the subnets.

C ✗ — Use AWS Glue to convert JSON to Parquet and partition by date before

querying with Amazon Athena.

Rationale: Reduces Athena cost by scanning fewer bytes, but ETL traffic to S3 would still traverse the NAT Gateway without an S3 endpoint; it does not eliminate those charges.

D ✗ — Buy a Savings Plan to cover compute during the ETL windows.

Rationale: Savings Plans are for steady EC2 or AWS Fargate usage and do not remove NAT Gateway data transfer charges; they do not solve the specific cost driver here.

Explanation

Analytics and AI/ML costs commonly include compute, storage, data scanned, and data movement. In this scenario, the unexpected charges are tied to the NAT Gateway, which is a data movement cost. When private subnets access Amazon S3 without a dedicated S3 endpoint, traffic to S3 traverses the NAT Gateway, leading to additional transfer-related charges. The most direct and minimal change to eliminate those charges is to keep S3 traffic inside the AWS network by using an Amazon S3 Gateway VPC endpoint and updating the private subnet route tables to point S3 traffic to that endpoint. This maintains a private architecture (no internet exposure) and removes the NAT path for S3 access.

Using S3 Intelligent-Tiering targets storage optimization, not network transfer. Converting JSON to Parquet and partitioning by date is an effective way to reduce Amazon Athena cost by lowering the number of bytes scanned, but it does not inherently remove NAT Gateway charges if traffic still flows through the NAT. Buying a Savings Plan can reduce steady compute costs for EC2 or AWS Fargate usage, but it does nothing to reduce NAT Gateway data transfer charges. Therefore, a Gateway VPC endpoint for S3 is the only solution that directly eliminates the identified cost driver with the smallest architectural change.

Key Concepts

- **Data movement costs:** Hidden data transfer charges can occur during ETL and analytics, including charges related to NAT Gateways when private subnets access external services.
- **S3 Gateway VPC endpoint:** Provides private connectivity from a VPC to Amazon S3; updating route tables keeps S3 traffic within AWS and avoids NAT Gateway charges.
- **Storage vs. scan vs. transfer:** S3 Intelligent-Tiering reduces storage costs; Parquet/partitioning lowers Athena scanned data; neither addresses NAT Gateway data transfer.
- **Minimal-change principle:** The best solution directly targets the primary cost driver (NAT path) with minimal architecture changes and without exposing resources to the internet.

Common Pitfalls

- Optimizing storage class or query scan size while overlooking that NAT Gateway data transfer is the dominant cost driver in this scenario. This fails to eliminate the immediate charges.

Glossary

- **Gateway VPC endpoint:** A VPC construct that routes traffic to supported AWS services (like Amazon S3) privately, without using the internet or a NAT Gateway.

14.49 — Root user only tasks (D2.T2.3)

Domain 2 • Task 2.3 ↑ Back to Question 14.49

Your company wants to permanently close a standalone AWS account that is not part of AWS Organizations. The billing console does not show the option for any administrator IAM user. Which identity must perform the account closure to complete this irreversible action?

Options & Rationales

A ✗ — An IAM user with AdministratorAccess policy attached.

Rationale: Even with full administrative IAM permissions, an IAM user cannot permanently close the account; closure requires the root credentials.

B ✗ — An IAM role assumed by the account's billing administrator.

Rationale: Assumed IAM roles, including billing administrators, cannot execute the permanent account deletion step that only the root identity can perform.

C ✗ — The AWS Organizations management account owner (if the account is in an organization).

Rationale: In AWS Organizations, the management account helps centrally manage member accounts, but this scenario explicitly involves a standalone account that is not in an organization, so only that account's root user can close it.

D ✓ — The AWS account root user (the original account owner credentials).

Rationale: Only the original account owner credentials (root user) can permanently close and delete an AWS account; IAM users or roles cannot perform this irreversible action.

Explanation

Certain account-level, irreversible actions are deliberately restricted to the original account owner credentials to protect ownership and prevent accidental loss. For a standalone AWS account that is not part of AWS Organizations, closing and deleting the account is one such high-impact action: only the root identity created with the account can complete the permanent closure process. This ensures a single, ultimate control point for account termination.

Why the correct identity is required. The root identity is the original account credential that represents ownership. AWS reserves specific, high-risk operations—like permanent account closure for standalone accounts—for that identity to reduce the chance that delegated administrators or temporary roles accidentally remove access or delete billing history.

Why the other identities are not allowed. Administrator IAM users, even with broad policies, are designed for delegated daily management but not for completing irreversible ownership changes. Assumed IAM roles and designated billing administrators provide scoped privileges for operational tasks; they cannot override the requirement for the account owner to confirm and execute permanent closure of a standalone account. In an AWS Organizations context, the management account and delegated admin accounts can centrally close member accounts created in the organization, but this does not change the requirement for root credentials when closing a standalone account.

Key Concepts

- **Root identity:** The original credentials created when the AWS account is established; used for ultimate, account-level control and specific irreversible tasks.
- **IAM identities vs. root:** IAM users and roles are for delegated administration and do not have permission to perform certain owner-only actions.
- **Irreversible actions protection:** AWS restricts high-impact changes to the root identity to prevent accidental or unauthorized account termination.

Common Pitfalls

- Assuming AdministratorAccess allows all actions: full IAM permissions do not include some root-only operations.
- Confusing organization administrative capabilities with per-account ownership: Organizations help manage member accounts but do not bypass per-account root restrictions for standalone accounts.

14.50 — IaC benefits with CloudFormation (D3.T3.1)

Domain 3 • Task 3.1 ↑ Back to Question 14.50

A team uses AWS CloudFormation and wants to keep resources aligned with their templates over time. Which actions are INCORRECT when using drift detection?

Options & Rationales

A ✗ — Automate periodic drift checks and alerts using scheduled jobs to catch divergence early.

Rationale: Regularly running drift detection is a valid practice to proactively discover drift.

B ✓ — Make routine configuration changes directly in the console and skip drift checks to move faster.

Rationale: Out-of-band edits create drift and bypass the template as the source of truth, which drift detection is intended to reveal—this is an anti-pattern.

C ✗ — Run drift detection on the stack to identify resources whose configurations differ from the template.

Rationale: This correctly uses drift detection to surface differences between actual resource state and the declared template.

D ✓ — Assume CloudFormation automatically fixes any manual console edits, so drift detection is unnecessary.

Rationale: CloudFormation does not auto-remediate out-of-band changes. Skipping drift detection violates the purpose of identifying divergence from the template.

E ✗ — Review drift results and update the template, then perform a stack update to realign resource state.

Rationale: Investigating findings and reconciling via template and stack update is a correct, template-first remediation approach.

Explanation

Drift detection in AWS CloudFormation is used to discover when live resource configurations no longer match what is declared in the stack template. Since templates are intended to be the single source of truth, teams should detect and then reconcile any differences by updating the template and applying a stack update. CloudFormation does not silently repair out-of-band changes; it requires explicit action informed by drift findings.

The correct practices include running drift detection on a stack, reviewing the results, and remediating by updating the template followed by a stack update. Automating periodic checks is also reasonable to surface divergence early. In contrast, assuming CloudFormation will automatically fix console edits ignores the fact that drift detection's role is to identify mismatches; there is no automatic correction. Similarly, making routine changes directly in the console and skipping drift checks abandons the template as the authoritative definition and intentionally creates drift.

Key Concepts

- **Drift detection:** A CloudFormation feature that reports when resource configurations differ from what the template declares.
- **Single source of truth:** The template should define desired state; changes should be made through template updates and stack operations.
- **Template-first remediation:** Align live resources by updating the template and then applying a stack update, not by ad hoc console edits.

Common Pitfalls

- Believing CloudFormation auto-corrects manual changes, leading teams to skip drift detection and miss configuration divergence.

14.51 — DMS for heterogeneous DB migrations (D3.T3.4)

Domain 3 • Task 3.4 ↑ Back to Question 14.51

A company must move a production database from an on-premises Oracle engine to an Amazon Aurora PostgreSQL instance with minimal downtime and without losing in-flight transactions. Which solution best supports continuous replication and preserves data consistency during cutover?

Options & Rationales

A ✗ — Export a nightly full dump from Oracle, import into Aurora, and schedule a short maintenance window for final sync.

Rationale: A nightly dump creates longer downtime and risks losing transactions between the last dump and cutover; it does not provide continuous synchronization.

B ✗ — Create an RDS Read Replica of the on-premises Oracle instance and promote it to primary after replication completes.

Rationale: RDS read replicas are not applicable for cross-engine migrations from Oracle to Aurora PostgreSQL and do not handle heterogeneous engine replication.

C ✓ — Use AWS Database Migration Service with ongoing change replication from the source to the target, validate data, then cut over when synchronized.

Rationale: AWS DMS supports continuous change-data capture replication between different database engines and includes data validation to minimize downtime and preserve consistency during cutover.

D ✗ — Use AWS Schema Conversion Tool to migrate data in one step directly into Aurora without separate replication.

Rationale: The Schema Conversion Tool helps convert database schemas but does not perform continuous data replication or ongoing change capture needed for low-downtime migration.

Explanation

When moving between different database engines with a requirement for minimal downtime and preserving in-flight transactions, the preferred approach uses a service that captures ongoing changes from the source and applies them to the target until cutover. A migration that continuously replicates changes keeps the target synchronized with the source so the final switch requires only a small pause for validation. Simply exporting and importing full dumps creates larger windows of data loss. Read-replica patterns are engine-specific and do not work for heterogeneous engine conversions. A schema conversion tool helps with structural differences but does not itself provide live data replication.

Key Concepts

- **Continuous change capture (CDC):** Captures transactions from the source database as they occur and applies them to the target to maintain

synchronization.

- **Heterogeneous migration:** Moving between different database engines often requires both schema conversion and ongoing data replication to avoid extended downtime.
- **Data validation during migration:** Verifying record counts and checksums or using built-in validation reduces the risk of inconsistent data after cutover.

Common Pitfalls

- Assuming an export/import approach satisfies low-downtime requirements; it can leave a gap of lost transactions.
- Believing read-replica mechanisms always work across database engines; they typically do not support cross-engine replication.
- Relying only on schema conversion without a replication mechanism will not keep live data synchronized.

14.52 — IaC promotes repeatability & op excellence (D1.T1.2)

Domain 1 • Task 1.2 ↑ Back to Question 14.52

Which statement about idempotence in infrastructure as code is NOT correct?

Options & Rationales

A ✗ — Applying the same template multiple times results in the same infrastructure state without unintended side effects.

Rationale: This describes idempotence correctly: repeated application yields the same target state and prevents unexpected changes.

B ✓ — Repeated runs of IaC always create duplicate resources, so manual cleanup is required after each deployment.

Rationale: This is incorrect and represents an anti-pattern: properly designed IaC avoids creating duplicates on repeated runs and does not require manual cleanup.

C ✗ — Idempotent deployments allow safe, repeatable updates and reduce configuration drift between environments.

Rationale: This is accurate: idempotence supports consistent updates and helps prevent differences between declared and actual resources.

D ✗ — Idempotence enables automation pipelines to redeploy templates safely as part of CI/CD workflows.

Rationale: This is correct: idempotence makes automated redeployments predictable and suitable for CI/CD practices.

Explanation

Idempotence means that running an infrastructure definition multiple times converges to a single, intended configuration rather than producing new or conflicting resources. This property is essential for automation, repeatable deployments, and predictable rollouts.

Why the NOT statement is wrong. Claiming that repeated executions always produce duplicate resources and require manual cleanup describes a flawed approach. Well-designed infrastructure-as-code tools track resource identity and update existing resources when necessary, preventing duplicates and avoiding manual intervention.

Why the other statements are correct. Saying repeated application yields the same state, that idempotence reduces drift, and that it supports CI/CD are all aligned with the purpose of idempotent deployments. These benefits enable safe automation and consistent environments.

Key Concepts

- **Idempotence:** Defining infrastructure so repeated application results in the same final state, preventing unintended extra resources.
- **Configuration drift:** The gap between declared configuration and actual deployed resources; idempotence helps keep them aligned.
- **Automation safety:** Idempotent templates allow pipelines to run deployments repeatedly without causing resource duplication or instability.

Common Pitfalls

- Assuming a deployment tool is idempotent without verifying how it manages resource identities can lead to surprises.
- Confusing one-time provisioning scripts with declarative IaC; imperative scripts may create duplicates while declarative templates are designed to be idempotent.

14.53 — Convertible RIs allow family changes (D4.T4.1)

Domain 4 • Task 4.1 ↑ Back to Question 14.53

Select TWO statements that correctly describe a capability of Convertible Reserved Instances for Amazon EC2 in a changing capacity requirement scenario

Options & Rationales

A ✗ — Family changes are automatic when you stop and restart instances of a different instance family.

Rationale: Stopping and restarting instances does not change the reservation's linked family; a reservation exchange is required to reassign value to another family.

B ✓ — You can exchange the unused value of a reservation to obtain reservations for different EC2 instance families.

Rationale: Convertible Reserved Instances support an exchange operation that converts the remaining monetary value of an existing reservation into one or more reservations for other instance families.

C ✓ — The reservation exchange preserves the benefit of lower hourly pricing compared with running instances entirely On-Demand.

Rationale: When you exchange a Convertible reservation, the discounted pricing associated with the remaining reservation value continues to apply to the new reservations, maintaining cost savings versus On-Demand rates.

D ✗ — You must cancel the original reservation and wait for a refund before purchasing different instance family reservations.

Rationale: Convertible reservations use an exchange process rather than cancellation and refund; you do not need to cancel and wait for a refund to change families.

E ✗ — Exchanges are only allowed to the same instance family but a different availability zone.

Rationale: Convertible reservations are specifically designed to allow switching between instance families, not restricted to the same family within a different AZ.

Explanation

Convertible Reserved Instances provide flexibility by allowing customers to transform the remaining monetary commitment of an active reservation into new reservations that better match changing compute needs. This works through an exchange operation that recalculates the unused reservation value and applies it toward one or more replacement reservations. The exchange mechanism keeps the discounted pricing benefit tied to the original commitment, so you retain lower rates compared with running identical capacity On-Demand.

Key Concepts

- **Convertible reservation exchange:** A controlled operation that converts the remaining reservation value into new reservations, enabling a change of EC2 instance families without canceling and repurchasing.
- **Preserved discounted pricing:** After an exchange, the discount associated with the original reservation's remaining value continues to reduce the effective hourly cost of the new reservations compared to On-Demand prices.

Why the correct statements are right. The first correct statement describes the exchange ability: instead of canceling or buying new reservations outright, you apply the remaining monetary value toward other instance families. The second correct statement describes the cost effect: because the exchange uses the existing reserved commitment, you keep discounted pricing rather than reverting to On-Demand rates.

Why the incorrect statements are wrong. Cancellation and refund are not the required path; the exchange process eliminates the need to cancel and wait for refunds. Simply stopping and restarting instances does not alter which reservations cover them. Exchanges are intended to move value between families, so the claim that exchanges are limited to the same family within a different AZ is incorrect.

Common Pitfalls

- Confusing instance lifecycle actions (stop/start) with reservation management can lead to assuming family changes happen automatically.
- Thinking cancellation and refund are necessary overlooks the built-in exchange capability.

Glossary

- **Convertible Reserved Instance:** A reservation type that lets you change the attributes of the reservation (such as instance family) by exchanging remaining value for new reservations.
- **Exchange operation:** The process that reallocates the leftover reservation value to create replacement reservations.

14.54 — Apply model during compliance audits (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 14.54

A company with dozens of Amazon S3 buckets across multiple teams discovers during a pre-audit that several buckets became publicly listable due to permissive bucket policies. The compliance lead asks for the fastest, account-wide way to prevent any existing or future buckets from becoming publicly accessible, with minimal ongoing management. What should a cloud practitioner do?

Options & Rationales

A ✗ — Encrypt all S3 buckets with SSE-KMS using a customer managed key.

Rationale: Addresses encryption at rest and key control, but does not prevent a bucket or object from being publicly accessible.

B ✓ — Enable Amazon S3 Block Public Access at the account level.

Rationale: Blocks public ACLs and bucket policies for all current and new buckets in the account. It's a single, preventive control that minimizes per-bucket work.

C ✗ — Turn on AWS Security Hub to identify and alert on public S3 buckets.

Rationale: Provides centralized detection and findings, but it does not block or prevent public access.

D ✗ — Set each bucket's ACL to private and rely on IAM to control access.

Rationale: Requires per-bucket changes and a bucket policy can still allow public access; not an account-wide preventive guardrail.

Explanation

Under the shared responsibility model, AWS secures the infrastructure, while customers configure their resources, including S3 access controls. When multiple buckets have accidentally become public due to permissive policies, the most effective and least labor-intensive fix is a preventive, account-wide control. Amazon S3 Block Public Access can be enabled at the account level to

block public ACLs and to ignore bucket policies that grant public access. This immediately protects existing buckets and applies to new buckets by default, reducing operational overhead and helping satisfy audit requirements that no bucket be publicly accessible.

Encrypting buckets with SSE-KMS improves encryption at rest and key governance, but it does not change who can access data; a publicly accessible bucket remains public even if encrypted. Turning on AWS Security Hub helps detect and aggregate findings (including public S3 buckets) but is a detective control; it does not enforce prevention. Manually setting each bucket's ACL to private is operationally heavy and incomplete because a bucket policy can still grant public access; it also does not provide an account-wide guardrail.

Key Concepts

- **Shared responsibility model:** Customers must configure S3 permissions, including preventing unintended public access.
- **S3 Block Public Access:** Account- or bucket-level setting that blocks public ACLs and prevents bucket policies from granting public access.
- **Preventive vs. detective controls:** Preventive controls (e.g., S3 Block Public Access) stop misconfigurations; detective controls (e.g., Security Hub) only report them.
- **Account-wide guardrails:** Centralized settings reduce per-resource changes and lower operational risk across teams.

Common Pitfalls

- Assuming encryption (SSE-KMS) prevents public access; it protects data at rest but does not enforce access restrictions.
- Relying solely on detection tools or ACLs, which either don't block access or can be overridden by bucket policies.

14.55 — Model across deployment models (hybrid) (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 14.55

Which statement best describes how responsibility for security and compliance is handled when an organization runs a mix of on-premises and AWS resources (a hybrid deployment)?

Options & Rationales

A ✓ — Both the cloud provider and the customer share parts of security; the customer must apply matching controls across on-premises and cloud resources.

Rationale: In hybrid deployments the provider secures the underlying cloud infrastructure while the customer configures and protects their data and access; consistent controls across environments are required.

B ✗ — The cloud provider is fully responsible for all security and compliance for any resource the customer connects to the cloud.

Rationale: This is incorrect because the provider does not handle customer

configuration, data protection, or identity management; customers retain those responsibilities.

C ✗ — The customer is solely responsible for physical security of the cloud provider's data centers when using hybrid architectures.

Rationale: This is false; physical infrastructure and data center security remain the cloud provider's responsibility, not the customer's.

D ✗ — Security controls only need to be applied in the cloud; on-premises systems are outside the shared model.

Rationale: Incorrect because hybrid models require aligned controls both on-premises and in the cloud to maintain consistent security and compliance posture.

Explanation

When organizations operate both on-premises systems and cloud services together, responsibility for protecting resources is split between the cloud provider and the customer. The provider handles the security of the underlying infrastructure, physical facilities, and foundational services. The customer remains responsible for how they configure services, protect their data, and manage identities and access. In a hybrid scenario, this means the organization must apply compatible security and compliance controls across both environments so protections remain effective regardless of where a workload runs.

Why the correct choice fits and others do not. The correct statement identifies the split of duties and emphasizes that the customer must implement matching controls across both environments. The incorrect options either place full responsibility on the provider, mistakenly assign physical data-center duties to the customer, or suggest ignoring on-premises controls—each of which contradicts the shared-responsibility principle for hybrid architectures.

Key Concepts

- **Shared responsibility model:** Describes which security tasks the provider manages (infrastructure, physical security) and which the customer manages (configuration, data protection, identity).
- **Hybrid alignment:** Ensures controls such as access management, encryption, and monitoring are implemented consistently on-premises and in the cloud to avoid gaps.

Common Pitfalls

- Assuming the provider covers customer-specific configuration or data protection leads to exposed resources.
- Applying security only to cloud resources and ignoring on-premises systems creates inconsistent protection across the hybrid estate.

Glossary

- **Configuration responsibility:** Customer actions such as setting access permissions, network controls, and encryption settings.
- **Provider responsibility:** Cloud vendor duties like securing the physical data centers, host infrastructure, and managed platform services.

14.56 — EC2 families: compute-optimized vs memory (D3.T3.3)

Domain 3 • Task 3.3 ↑ Back to Question 14.56

A company runs a web service that performs heavy numerical simulations using many CPU threads and keeps only small temporary state in memory. They must minimize cost while delivering high CPU throughput. Which EC2 instance family is the best choice?

Options & Rationales

A ✗ — Choose a memory-optimized instance family (large memory per vCPU).

Rationale: Memory-optimized instances are tuned for workloads requiring lots of RAM per CPU, which is unnecessary for small in-memory state and increases cost.

B ✗ — Choose a general-purpose instance family (balanced vCPU and memory).

Rationale: General-purpose instances offer a balanced ratio and may be over-provisioned for CPU-bound simulations, making them less cost-efficient than compute-optimized options.

C ✓ — Choose a compute-optimized instance family (high vCPU-to-memory ratio).

Rationale: Compute-optimized families provide more vCPU capacity per GB of RAM, matching CPU-bound workloads and lowering cost for compute-heavy tasks.

D ✗ — Choose a storage-optimized instance family (high local I/O throughput).

Rationale: Storage-optimized instances target heavy disk I/O workloads; they do not provide the best CPU-to-memory ratio for compute-bound simulations.

Explanation

When selecting an EC2 family, match the workload's dominant resource need to the instance's vCPU-to-memory balance. For a workload that performs intensive CPU calculations while maintaining only small transient state, the key capacity metric is the number of virtual CPUs relative to RAM. Compute-focused instance types are configured with a higher vCPU-per-GB-of-memory ratio, delivering more processing power for each dollar of memory provisioned. This makes them appropriate and cost-efficient for CPU-bound tasks.

Why the correct choice fits. The recommended option identifies an instance family designed to maximize CPU resources compared to memory. For heavy

numerical or parallel compute jobs, that higher vCPU density provides better throughput and often lower cost than choosing instances with extra RAM that will remain unused.

Why the other choices are unsuitable. Choosing instances optimized for large memory capacity provides excess RAM that the workload does not need and increases cost. General-purpose instances give a balanced CPU and memory mix, which can be acceptable but is not the most efficient for strictly CPU-bound workloads. Storage-optimized instances focus on local I/O performance and do not prioritize the vCPU-to-memory ratio needed for compute-heavy processing.

Key Concepts

- **vCPU-to-memory ratio:** The comparative measure of processing cores to RAM; select an instance family whose ratio aligns with the workload's dominant resource.
- **Compute-bound workload:** An application where processing cycles (CPU) are the limiting factor rather than memory or disk I/O.

Common Pitfalls

- Selecting an instance based on a single metric like price without matching the dominant resource leads to wasted capacity.
- Confusing high I/O or high-memory requirements with CPU needs; each optimization target is a different instance family.

14.57 — ALB vs NLB vs GWLB options (D3.T3.3)

Domain 3 • Task 3.3 ↑ Back to Question 14.57

Select TWO options that describe appropriate uses or characteristics of a Gateway Load Balancer when integrating third-party virtual network appliances into an AWS VPC

Options & Rationales

A ✓ — Provide transparent forwarding that lets the appliance receive and handle traffic without replacing client IP addresses.

Rationale: Gateway Load Balancer forwards traffic in a way that preserves visibility for the appliance, enabling stateful inspection and logging based on original connection details.

B ✗ — Act as a global content distribution service to cache and accelerate web assets for users worldwide.

Rationale: Global caching and acceleration are functions of a content distribution service rather than a Gateway Load Balancer.

C ✗ — Optimize high-volume TCP passthrough with extremely low latency for static content delivery at the network edge.

Rationale: High-throughput, low-latency TCP passthrough for edge delivery is

a use case for a network-focused load balancer or edge service, not the primary purpose of a Gateway Load Balancer.

D ✗ — Replace the need for virtual appliances by providing built-in intrusion detection and custom traffic inspection rules.

Rationale: Gateway Load Balancer forwards traffic to third-party appliances; it does not itself provide built-in, customizable inspection features that replace those appliances.

E ✓ — Insert a third-party virtual firewall inline so all traffic flows through the appliance before reaching backend instances.

Rationale: Gateway Load Balancer is designed to direct packets to virtual appliances (for example, firewalls) so the appliance inspects and processes traffic in-line.

F ✗ — Terminate HTTPS connections and perform application-layer routing based on URL paths.

Rationale: Application-layer termination and content-based routing are functions of an application-oriented load balancer, not a Gateway Load Balancer.

Explanation

A Gateway Load Balancer is intended to insert and forward traffic to third-party virtual network appliances so those appliances can inspect, filter, or modify packets as if they were inline network devices. It does not perform application-layer content routing, global caching, or replace the specialized inspection capabilities of a virtual appliance.

Key reasoning steps:

- Use a Gateway Load Balancer when you need to route traffic through vendor appliances (such as firewalls, intrusion prevention systems, or other network functions) that operate at or near the packet level. This enables centralized deployment of appliance-based inspection without reconfiguring each instance.
- Gateway Load Balancer preserves the flow context so appliances can make decisions based on original connection information, which is important for stateful processing and accurate logging.
- It is not designed to terminate TLS and perform URL-based routing (an application-layer concern), nor is it a global caching or edge acceleration service.

Why the correct answers are right and others are wrong. The correct options describe inserting a third-party device inline and forwarding traffic while preserving client information—both fundamental to deploying virtual network appliances using a Gateway Load Balancer. The incorrect options describe application-layer termination, edge content distribution, or built-in inspection capabilities, which are not provided by a Gateway Load Balancer and belong to

other services or appliances.

Key Concepts

- **Gateway forwarding:** Forwards traffic to virtual appliances so the appliance can process packets inline with minimal interruption to the flow.
- **Appliance transparency:** Maintains original connection visibility so stateful devices can inspect and log traffic using client and server addresses.

Common Pitfalls

- Confusing Gateway Load Balancer with an application-level balancer and expecting URL/path routing or TLS termination leads to incorrect architecture choices.
- Assuming the Gateway Load Balancer provides built-in deep inspection or caching can prevent proper use of third-party appliance features.

Glossary

- **Virtual appliance:** A software-based network function (for example, firewall or IDS) that runs in a virtual machine or container and inspects or modifies network traffic.
- **Inline deployment:** Placing an appliance in the traffic path so all relevant packets pass through it for processing.

14.58 — Fargate vs EC2 for containers (D3.T3.3)

Domain 3 • Task 3.3 ↑ Back to Question 14.58

A company plans to run containerized applications on AWS. Select TWO statements that correctly guide when to choose AWS Fargate versus Amazon EC2 for the container compute layer.

Options & Rationales

A ✓ — Use Amazon EC2 when you require full control of the underlying instances, including custom AMIs or specialized kernel configuration.

Rationale: EC2 provides instance-level access, enabling custom images and low-level tuning that are not available with Fargate.

B ✗ — Choose AWS Fargate when you need direct SSH access to the container host for troubleshooting.

Rationale: Fargate does not provide host-level access; you cannot SSH into the underlying infrastructure.

C ✗ — Use AWS Fargate only for stateless containers; stateful containers must run on EC2.

Rationale: Fargate can run tasks that persist data using services like Amazon EFS; it is not limited to stateless use only.

D ✗ — Choose Amazon EC2 when you want a serverless, per-task compute model without managing capacity.

Rationale: A serverless, per-task model without capacity management describes Fargate, not EC2.

E ✓ — Use AWS Fargate when you want AWS to handle the container host lifecycle (provisioning, patching, and scaling) to reduce operational effort.

Rationale: Fargate is a serverless, task-level compute option where AWS manages the underlying hosts, removing the need to operate instances.

Explanation

Selecting compute for containers on AWS boils down to the level of infrastructure control you need versus the desire to minimize operations. AWS Fargate is a serverless, task-focused compute option. You define CPU and memory at the task level, and AWS operates the container hosts behind the scenes—covering capacity provisioning, OS patching, and scaling. This reduces undifferentiated heavy lifting. By contrast, Amazon EC2 exposes the underlying instances, allowing you to choose AMIs, modify kernel parameters, and install host-level software—useful when low-level customization or specialized images are required.

The correct guidance is to pick Fargate when you want AWS to manage the container host lifecycle and reduce operational burden, and to pick EC2 when you need full instance control such as custom AMIs or kernel tuning. The statement suggesting Fargate for SSH-based troubleshooting is incorrect because host access is not provided with Fargate. The claim that EC2 offers a serverless, per-task model is wrong; that describes Fargate. Finally, asserting Fargate is only for stateless containers is inaccurate; it can run workloads that use persistent storage via complementary services.

Key Concepts

- **Managed vs. instance control:** Fargate manages hosts; EC2 exposes instances for customization.
- **Operational overhead:** Fargate reduces provisioning, patching, and scaling tasks.
- **Customization needs:** EC2 enables custom AMIs and kernel-level tuning.

Common Pitfalls

- Assuming Fargate allows SSH to hosts or is limited to stateless apps; it abstracts hosts and supports persistent storage via integrated services.

14.59 — Cost Explorer for historical analytics (D4.T4.2)

Domain 4 • Task 4.2 ↑ Back to Question 14.59

A company uses AWS Organizations with consolidated billing and several linked accounts. Last month it purchased Savings Plans with upfront fees to reduce Amazon EC2 costs across all accounts. In AWS Cost Explorer, Finance is reviewing a monthly view using Unblended cost and still sees EC2 spend appear unchanged. They want the report to show the true effective cost of the

commitments across the period and linked accounts. What should they do in Cost Explorer?

Options & Rationales

A ✗ — Use AWS Cost Explorer with daily granularity and group by API operation to reveal Savings Plans coverage.

Rationale: Granularity and API operation grouping do not change how commitment fees are recognized; they won't reveal the amortized benefit.

B ✓ — Use AWS Cost Explorer and switch the metric to Amortized cost to spread Savings Plans/RI upfront fees over time across linked accounts.

Rationale: Amortized cost allocates upfront fees of Savings Plans/RIs across the covered period, reflecting the true effective cost across linked accounts.

C ✗ — Activate the Project tag as a cost allocation tag and rerun the same monthly view for last quarter.

Rationale: Cost allocation tags are not retroactive and do not affect commitment amortization; they won't fix the missing benefit in historical views.

D ✗ — In AWS Cost Explorer, group by Purchase option while keeping Unblended cost to separate On-Demand from Savings Plans.

Rationale: Grouping by Purchase option shows mix, but Unblended cost omits the amortized effect of upfront commitments; benefits still won't be fully reflected.

Explanation

AWS Cost Explorer is used to visualize historical AWS spend and usage. Selecting the correct cost metric is essential when analyzing commitment-based discounts. Unblended cost reflects the list price at the time of usage and will not spread any upfront payments for Reserved Instances (RIs) or Savings Plans across the billing period. Amortized cost, by contrast, distributes the upfront fees of RIs and Savings Plans evenly over the relevant period to show the true effective cost. Under consolidated billing in AWS Organizations, amortized cost will reflect these benefits across linked accounts, giving Finance an accurate view of how commitments reduce total EC2 cost over time.

In this scenario, Finance is reviewing a monthly Unblended cost view and cannot see the expected savings because the upfront Savings Plans payment is not being amortized. Changing the metric in Cost Explorer to Amortized cost addresses this by spreading the upfront fee over the commitment term, making the effective cost reduction visible across the period and accounts. Grouping and filtering choices (such as Purchase option or API operation) are useful for analysis, but they do not change how the costs are recognized. Similarly, activating tags helps with attribution going forward but does not retroactively tag historical spend nor affect amortization.

Why the other choices are incorrect:

- Switching to daily granularity and grouping by API operation changes the view, not the cost recognition method; it will not display amortized benefits.
- Activating cost allocation tags does not retroactively apply to past charges and has no bearing on how upfront commitments are spread.
- Grouping by Purchase option helps reveal the mix of On-Demand versus Savings Plans, but with Unblended cost it still won't reflect the amortized effect of upfront fees.

Key Concepts

- **Unblended cost:** Shows list prices at the time of usage; does not account for spreading upfront RI/Savings Plans fees.
- **Amortized cost:** Spreads upfront RI/Savings Plans fees across the period, revealing true effective cost across linked accounts.
- **Metric vs. grouping:** The metric determines cost recognition; grouping (Service, Purchase option, Region) organizes data but doesn't change the metric.
- **Linked accounts (Organizations):** Commitment benefits are visible across linked accounts when using the appropriate metric in Cost Explorer.

Common Pitfalls

- Expecting Unblended cost to show commitment savings from upfront payments.
- Assuming cost allocation tags retroactively apply to historical spend or change commitment accounting.

14.60 — AWS Artifact provides reports on-demand (D2.T2.2)

Domain 2 • Task 2.2 ↑ Back to Question 14.60

Which statement best describes the primary purpose of AWS Artifact?

Options & Rationales

A ✓ — It gives customers on-demand access to AWS audit and compliance documents they can use for their own audits.

Rationale: AWS Artifact provides a centralized portal where customers can retrieve AWS audit reports and certifications for their compliance needs.

B ✗ — It schedules and runs new third-party audits of a customer's AWS account on demand.

Rationale: AWS Artifact does not perform or schedule third-party audits of customer accounts; it provides access to AWS's existing audit reports and certifications.

C ✗ — It stores a customer's own internal compliance reports and manages their retention policies.

Rationale: AWS Artifact contains AWS's compliance documentation rather than acting as a repository for customer-created compliance reports.

D ✗ — It is a support channel for opening compliance-related service tickets with AWS Support.

Rationale: AWS Artifact is not a support ticketing service; it is a documentation portal for retrieving AWS compliance evidence.

Explanation

AWS Artifact is a centralized documentation portal that enables customers to retrieve authoritative audit reports and certifications produced for AWS services. These documents—such as independent audit attestations and international certification records—help customers demonstrate how AWS controls support their compliance and audit activities. Artifact supplies existing evidence on demand; it does not perform audits, host customer-created compliance files, or function as a support ticketing system.

Key Concepts

- **Compliance documentation portal:** A single location where service provider audit reports and certifications are made available to customers for review and reuse.
- **Independent audit evidence:** Third-party assessments and certificates issued to the cloud provider that customers can reference to support their own compliance programs.
- **Scope of service:** The portal provides provider-generated documents rather than performing new audits or managing a customer's internal records.

Common Pitfalls

- Confusing availability of provider audit reports with running audits against a customer's environment leads to choosing a statement that implies Artifact triggers new audits.
- Assuming Artifact is a general document storage service for customer files or a support channel can misidentify its purpose.

14.61 — AWS Config rules for compliance (D2.T2.2)

Domain 2 • Task 2.2 ↑ Back to Question 14.61

Which statement about a service that records resource configurations and checks them against desired states for continuous compliance is NOT correct?

Options & Rationales

A ✓ — It replaces API call auditing tools and provides the same detailed user activity logs as an API audit trail.

Rationale: This is incorrect: the service focuses on resource configuration states and compliance evaluations, not on detailed API call or user activity logging; it does not substitute for an API auditing tool.

B ✗ — It records configuration histories and evaluates changes to identify drift from desired states.

Rationale: This is a correct description: the service captures configuration snapshots and change history to detect when resources diverge from expected configurations.

C ✗ — It continuously evaluates resources using rules and can initiate automatic remediation when evaluations fail.

Rationale: This is accurate: the service runs rules (managed or custom) to assess compliance and can trigger defined remediation actions for noncompliant findings.

D ✗ — It offers prebuilt rule sets maintained by the provider and the ability to author custom rule logic for specialized checks.

Rationale: This is correct: the service provides provider-maintained managed checks plus support for custom rules to evaluate specific compliance requirements.

Explanation

The described service continuously records the configuration of supported resources and maintains a change history. It compares recorded states to declared desired configurations using rule evaluations to detect configuration drift and report noncompliance. Evaluations can use provider-maintained rule templates or user-created logic, and can be paired with automated or manual remediation actions to return resources to compliance.

Why the incorrect statement fails. The incorrect claim confuses configuration auditing with API-level auditing. The service specializes in resource state and compliance checks rather than producing detailed API call logs that show which principal performed which API action; an API audit trail remains the appropriate tool for tracking user and service activity.

Key Concepts

- **Configuration recording:** Captures snapshots and ongoing changes of resource settings so historical states are available for comparison.
- **Rule-based evaluation:** Runs checks against desired state definitions; rules can be managed templates or custom logic authored by customers.
- **Remediation linkage:** Noncompliant findings can trigger corrective actions, either automatically or by operator initiation.

Common Pitfalls

- Assuming configuration auditing also provides full user activity logs leads to gaps in forensic or access investigations.
- Believing managed rules cover every organizational policy without using custom rules can leave specific requirements unchecked.

14.62 — Trade-offs among Well-Architected pillars (D1.T1.2)

Domain 1 • Task 1.2

↑ Back to Question 14.62

A company runs a web application on Amazon EC2 with Auto Scaling configured to add instances during traffic spikes and remove them when idle. Which statement about this design is WRONG?

Options & Rationales

A ✗ — Auto Scaling helps meet increased demand while avoiding payment for idle capacity.

Rationale: Auto Scaling adds capacity when needed and removes it when not, which aligns performance with usage and reduces unnecessary cost.

B ✓ — Auto Scaling guarantees zero downtime for any infrastructure failure without additional architectural changes.

Rationale: Auto Scaling does not by itself provide complete fault tolerance; design choices like spreading instances across Availability Zones and using load balancers are needed for high availability.

C ✗ — Auto Scaling can improve performance only when combined with appropriate scaling policies and monitoring.

Rationale: Effective scaling depends on well-chosen metrics, thresholds, and monitoring so instances are added or removed at the right time.

D ✗ — Auto Scaling supports cost optimization by reducing the number of running instances when load decreases.

Rationale: By removing excess instances during low demand, Auto Scaling helps lower compute costs while maintaining required capacity.

Explanation

Auto Scaling is a mechanism that adjusts compute capacity to match demand, helping balance performance and cost. It launches or terminates EC2 instances based on metrics and rules you define. However, Auto Scaling alone does not automatically solve all availability or fault-tolerance requirements: if your instances are all in a single Availability Zone or you lack health checks and a load balancer, an infrastructure failure can still cause downtime.

- The statement that Auto Scaling meets demand while avoiding payment for idle capacity is accurate because dynamic scaling aligns running instances with usage.
- The claim that Auto Scaling requires appropriate scaling policies and monitoring is accurate; without sensible metrics and thresholds, scaling can be too slow or cause flapping.
- The incorrect statement asserts a guarantee of zero downtime without further design; this is false because high availability typically also requires spreading instances across multiple Availability Zones, health checks, and a load balancer to route traffic away from failed instances.
- The statement that Auto Scaling reduces instance count during low load is accurate and relates directly to cost optimization.

Key Concepts

- **Auto Scaling:** Automatically adjusts the number of EC2 instances based on defined metrics and policies to meet demand and manage cost.
- **Scaling policies and monitoring:** Rules and metrics (for example, CPU or request rate) determine when to add or remove instances; good observability prevents slow or excessive scaling.
- **High availability vs. scaling:** Scaling changes capacity but does not by itself provide resilience to AZ- or instance-level failures; architectural patterns (multi-AZ, load balancers, health checks) are required.

Common Pitfalls

- Assuming Auto Scaling equals fault tolerance leads to under-designed availability; designers must also distribute instances and use health checks.
- Relying on a single poorly tuned metric can cause inappropriate scaling actions, either too frequent changes or delayed response.

14.63 — AWS security & compliance add value (D1.T1.1)

Domain 1 • Task 1.1 ↑ Back to Question 14.63

A company must show who changed critical IAM settings during a recent incident and provide evidence for a regulatory audit. Which two outcomes are best supported by enabling AWS CloudTrail?

Options & Rationales

A ✗ — Directly reduces monthly compute charges by optimizing instance usage.

Rationale: CloudTrail provides logging and visibility but does not optimize or change compute consumption to lower billing.

B ✗ — Encrypts application data stored in S3 buckets to ensure data-at-rest confidentiality.

Rationale: Encryption of stored objects is handled by services like Amazon S3 and KMS; CloudTrail focuses on event recording, not data encryption.

C ✓ — Creates tamper-evident, time-ordered records of API activity to support investigations.

Rationale: CloudTrail captures API calls with timestamps and event details, producing auditable logs that investigators can use to reconstruct actions.

D ✗ — Automatically blocks malicious API requests in real time to prevent further unauthorized changes.

Rationale: CloudTrail is a logging and audit service; it records events but does not actively block API calls—other services are needed for prevention.

E ✓ — Provides verifiable event logs that auditors can review to confirm compliance activities and shorten audit cycles.

Rationale: CloudTrail records provide documentary evidence of administrative actions and access, which auditors use to validate controls and speed reviews.

Explanation

AWS CloudTrail is a managed service that records account and service API activity as events. It captures who performed actions, when they occurred, and details about the request and response. These chronological event records are essential when reconstructing incidents and producing documentary evidence for audits. CloudTrail does not itself prevent or remediate activity, nor does it alter data storage encryption or compute costs.

Key Concepts

- **Audit logging:** Retains a sequential record of API calls and account activity so actions can be traced during investigations and reviews.
- **Forensic reconstruction:** Time-stamped event details (caller identity, source IP, request parameters) enable investigators to determine what changed and by whom.
- **Compliance evidence:** Persistent, verifiable logs serve as proof that controls were applied and help auditors validate security and governance posture.

Common Pitfalls

- Confusing logging with prevention: recording events does not stop an attack; active controls or detection services are required for blocking or automated response.
- Expecting cost reduction: CloudTrail increases visibility but does not directly optimize resource usage or lower compute bills.

Glossary

- **CloudTrail:** AWS service that records API activity and account events for auditing and operational troubleshooting.
- **Event record:** A log entry capturing who called which API, when, and with what parameters.

14.64 — Customer-only security tasks (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 14.64

A small business plans to store customer documents in Amazon S3. Which security task is primarily the customer's responsibility to ensure sensitive files are handled correctly?

Options & Rationales

A ✗ — Maintaining the physical servers and data center power systems that host S3.

Rationale: Physical infrastructure maintenance is managed by the cloud provider, not the customer.

B ✓ — Assign sensitivity levels and handling rules to each dataset (data classification).

Rationale: Determining which files are sensitive and how they must be

protected is a customer responsibility; this guides encryption, access, and retention decisions.

C ✗ — Automatically rotating AWS-managed encryption keys used by S3 without customer involvement.

Rationale: Management of provider-owned, AWS-managed keys is handled by the service; customer chooses if they want customer-managed keys instead.

D ✗ — Operating global edge locations and networking for Amazon's CDN services.

Rationale: Operation of edge locations and the CDN network is the provider's responsibility, not the customer's.

Explanation

When using cloud storage services, responsibilities split between the cloud provider and the customer. The provider operates and secures the underlying infrastructure, including physical facilities, servers, and the global network that runs managed services. The customer controls how their content is classified, accessed, and configured within those services.

The correct task is assigning sensitivity levels and handling rules to datasets. This activity—deciding which files contain confidential or regulated information and defining how they must be protected (encryption, access restrictions, retention)—is inherently a business decision tied to compliance, privacy, and risk appetite. It informs choices like whether to use customer-managed encryption keys, restrict access with least-privilege identities, or apply stricter lifecycle rules.

Other tasks in the options are provider responsibilities. Keeping physical servers and power systems operational is managed by the cloud provider. Using AWS-managed encryption keys and the service's automatic rotation is handled by the provider when those keys are chosen. Operating edge locations and CDN networking is also a provider function.

Key Concepts

- **Shared responsibility model:** The provider secures the infrastructure; the customer secures their data, identities, and configurations.
- **Data classification:** The process of labeling datasets by sensitivity and defining handling requirements that drive protection controls and compliance.
- **Customer-driven configuration:** Decisions such as which keys to manage, which users have access, and how data is lifecycle-managed depend on the customer's policies.

Common Pitfalls

- A common mistake is assuming the provider will determine which of a customer's files are sensitive. Another is confusing provider-managed operational tasks (infrastructure, edge operations, provider-managed keys) with customer responsibilities (policy, classification, access control).

14.65 — AWS Artifact for compliance reports (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 14.65

A healthcare startup is building a patient portal on Amazon EC2, Amazon RDS, and Amazon S3. They will handle protected health information (PHI) and are preparing for an external audit. The team wants to use AWS Artifact to do two things: (1) present auditor-verified evidence of AWS-owned controls for the underlying AWS infrastructure and managed services they use, and (2) accept any required account-level agreement to build under HIPAA. Which actions should they take using AWS Artifact?

Options & Rationales

A ✓ — Download the SOC 2 Type II report and program summaries from AWS Artifact Reports to evidence AWS-owned controls.

Rationale: AWS Artifact Reports provides SOC reports and program summaries documenting design and operating effectiveness of AWS controls across data centers, network, and managed services.

B ✓ — Review and accept the HIPAA Business Associate Addendum in AWS Artifact Agreements for the AWS account handling PHI.

Rationale: AWS Artifact Agreements is where you accept account-level agreements such as the HIPAA BAA—required when handling PHI.

C ✗ — Enable AWS CloudTrail and AWS Config, then present those logs and configuration histories as customer-owned control evidence.

Rationale: This is good practice for customer-owned controls, but it is not an action performed in AWS Artifact. The question asks for actions using AWS Artifact.

D ✗ — Use AWS Artifact to automatically validate and monitor the startup's EC2, RDS, and S3 configurations for HIPAA compliance.

Rationale: AWS Artifact supplies compliance evidence for AWS-owned controls; it does not configure, monitor, or validate customer workloads.

E ✗ — Use AWS Artifact to generate least-privilege IAM policies and enable KMS encryption for the application.

Rationale: IAM policies and KMS encryption are customer-owned controls. AWS Artifact cannot create or configure them; it only provides AWS compliance evidence and agreements.

F ✗ — Publish the downloaded SOC reports on the company's public website so all stakeholders can access them.

Rationale: Redistributing restricted reports is a pitfall. Share AWS Artifact documents only with parties who need them per the report terms.

Explanation

The shared responsibility model divides duties between AWS and the customer. AWS secures the infrastructure that runs AWS Cloud services (security of

the cloud), while customers configure and operate their workloads securely (security in the cloud). For audits, AWS provides authoritative evidence of its own controls through AWS Artifact. AWS Artifact has two key facets at this level: Reports (e.g., SOC 1/2/3, ISO 27001/27017/27018, PCI DSS Attestation of Compliance, and program summaries) and Agreements (e.g., HIPAA Business Associate Addendum, GDPR Data Processing Addendum).

Your overall compliance control set combines AWS's controls with your own workload controls:

$$C_{total} = C_{AWS} + C_{customer}$$

Variables used:

- C_{total}: total number of controls relied upon
- C_{AWS}: AWS-owned controls evidenced via AWS Artifact
- $C_{customer}$: customer-owned controls (configuration, processes)

To meet the two stated needs using AWS Artifact, the team should accept the HIPAA Business Associate Addendum in AWS Artifact Agreements for the account handling PHI, and download auditor-validated evidence—such as the SOC 2 Type II report and program summaries—from AWS Artifact Reports to demonstrate AWS-owned controls across data centers, network, and managed services. They should align any downloaded reports with their in-scope AWS Regions and services and map responsibilities using AWS-provided responsibility summaries. Customer-side evidence (for example, IAM least privilege, CloudTrail retention, or KMS encryption settings) remains separate and complements, but does not replace, AWS Artifact materials.

Proposals that claim AWS Artifact can automatically validate or monitor EC2, RDS, or S3 configurations are incorrect because AWS Artifact neither configures nor validates workloads. Publishing SOC reports publicly is also inappropriate; restricted documents should be shared only with parties who need them. Creating IAM policies or enabling KMS encryption is essential but falls under customer-owned controls, not actions performed within AWS Artifact.

Key Concepts

- **Shared responsibility model:** AWS secures the cloud; customers secure workloads and configurations in the cloud.
- **AWS Artifact Reports:** Download SOC/ISO/PCI reports and program summaries evidencing AWS-owned controls for infrastructure and managed services.
- **AWS Artifact Agreements:** Review and accept account-level agreements such as the HIPAA BAA and GDPR DPA.
- **Evidence pairing:** Align AWS reports to in-scope Regions/services and combine with CloudTrail, AWS Config, and IAM evidence for customer-owned controls.

Common Pitfalls

- Assuming possession of an AWS report equals application compliance; AWS Artifact only covers AWS-owned controls.
- Redistributing restricted reports beyond those who need them, violating report-sharing terms.

Exam 15

Question 15.1

A web application occasionally experiences slow responses. Which observability data type best helps you follow a single request across services to find which component added most latency?

○ **A.** Metrics — provide aggregated numerical measures like average latency over time.

○ **B.** Logs — contain immutable event records for auditing and debugging.

○ **C.** Traces — show an end-to-end request path with timing between components.

○ **D.** Alerts — notify operators when thresholds are crossed so they can restart services.

Explanation→

Question 15.2

A company uses AWS Organizations to link several member accounts under one paying account. Which statement best describes how usage-based volume pricing discounts are determined across the organization?

○ **A.** Each linked account is evaluated separately for volume discounts; aggregation does not apply.

○ **B.** Only the management account's usage qualifies for volume discounts; member accounts are excluded.

○ **C.** Usage across linked accounts is combined so the organization can reach higher volume pricing tiers.

○ **D.** Volume discounts apply only if every linked account has identical service usage patterns.

Explanation→

Question 15.3

Which statement best defines Amazon EC2 Dedicated Hosts for compliance and licensing scenarios?

○ **A.** An entire physical server allocated to one account that exposes sockets, cores, and a host ID, enabling host affinity and BYOL; pricing is per host.

○ **B.** Instances that run on single-tenant hardware managed by AWS placement, billed per instance, with no visibility into sockets/cores or guaranteed host affinity.

○ **C.** A VPC tenancy setting that makes all new instances dedicated and provides control over specific physical servers for licensing and audits.

○ **D.** A billing model where you pay only for instance capacity while still mapping workloads to specific sockets/cores for per-socket licensing.

Explanation→

Question 15.4

A team stores large, raw analytics files in Amazon S3 and needs to run ad hoc SQL queries without provisioning or managing query servers. Which AWS service best meets this requirement?

○ **A.** Amazon Redshift — a provisioned data warehouse optimized for complex, high-performance analytics with cluster management.

○ **B.** Amazon EMR — a managed big-data platform for running distributed processing frameworks such as Spark or Hadoop on provisioned clusters.

○ **C.** Amazon Athena — serverless SQL queries run directly over data in Amazon S3 without managing query infrastructure.

○ **D.** Amazon RDS — managed relational databases for transactional workloads that require long-running database instances and schema management.

Explanation→

Question 15.5

A company wants to improve sustainability by minimizing idle compute. The workload runs infrequently and is event-driven. Which actions are INCORRECT for this goal? Select TWO to AVOID.

☐ **A.** Keep large Amazon EC2 instances running 24/7 for occasional tasks to prevent cold starts.

☐ **B.** Migrate the processing to AWS Lambda and trigger it only on relevant events.

☐ **C.** Use AWS Fargate so containers run only when tasks are invoked.

☐ **D.** Manually pre-provision an ECS cluster on EC2 and keep services running at all times for a weekly job.

☐ **E.** Use AWS Step Functions to coordinate event-driven serverless tasks without running persistent servers.

Explanation→

Question 15.6

Your team plans to connect three VPCs (VPC-A, VPC-B, VPC-C) so workloads can communicate privately. Which TWO statements describe correct behavior or a required action when using VPC peering?

☐ **A.** VPC peering automatically allows routing through chained peers so route tables do not need updates.

☐ **B.** A peered VPC will not forward packets to another VPC on your behalf; it cannot act as a transit hop.

☐ **C.** Using a peering connection removes the need to configure route table entries for private traffic.

☐ **D.** You must create a separate peering connection for each VPC pair so traffic can flow directly between them.

☐ **E.** Once VPC-A peers with VPC-B, VPC-A can use that link to reach on-premises networks connected to VPC-B by VPN.

☐ **F.** VPC peering allows connections between VPCs that share the same IP ranges without changes.

Explanation→

Question 15.7

A finance team expects to see project-level costs but reports all resources appear under a single account total. Which likely cause explains why costs cannot be split by project in the cost reports?

○ **A.** The account is missing server-side encryption, which prevents the billing tool from reading resource usage.

○ **B.** Network traffic is routed through an internet gateway, causing costs to be aggregated at the VPC level rather than per resource.

○ **C.** All resources are in the same Availability Zone, so cost reports cannot separate them by project.

○ **D.** Many resources were not labeled with the project identifier tag, so the billing tool cannot group costs by project.

Explanation→

Question 15.8

Which statements describe how AWS re:Post helps resolve technical questions?

☐ **A.** It offers a guaranteed 15-minute response-time SLA for all questions.

☐ **B.** AWS participation and moderation help maintain accuracy and quality.

☐ **C.** It replaces the need for AWS Support plans for production incidents.

☐ **D.** It is a private forum limited to users within a single AWS account.

☐ **E.** Community voting and peer review elevate reliable answers faster.

Explanation→

Question 15.9

Which capability best describes AWS Transit Gateway's primary role in connecting multiple VPCs and an on-premises datacenter?

○ **A.** Perform inline traffic inspection and act as a stateful firewall for all traffic between VPCs.

○ **B.** Automatically convert VPC peering into a managed VPN with per-flow encryption between every VPC pair.

○ **C.** Replace Internet Gateways by providing public internet access for instances in all connected VPCs.

○ **D.** Provide a single, central routing hub that aggregates VPCs and on-premises networks so they can exchange routes through one attachment point.

Explanation→

Question 15.10

Which scenario best describes when to choose AWS Outposts for a workload?

○ **A.** A public-facing website with unpredictable global traffic that should be served from many edge locations worldwide.

○ **B.** A temporary batch analysis job that can tolerate interruptions and runs only when spare capacity is available.

○ **C.** An application needs cloud APIs and managed services running inside a company data center because users require extremely low latency to local devices.

○ **D.** A lightweight static marketing site with minimal backend needs and tight cost constraints where hosting simplicity is key.

Explanation→

Question 15.11

Which of the following best describes the primary purpose of AWS Security Token Service (STS)?

○ **A.** Stores and manages long-lived access keys for IAM users to authenticate API requests indefinitely.

○ **B.** Acts as a directory service for user accounts and stores passwords for AWS-managed identities.

○ **C.** Encrypts data at rest and handles customer-managed cryptographic keys for S3 and EBS volumes.

○ **D.** Provides temporary AWS credentials that let an authenticated principal assume a role and access AWS resources.

Explanation→

Question 15.12

Which statement best defines an AWS Availability Zone (AZ)?

○ **A.** A separate geographic area that contains multiple AZs and serves as a data residency boundary.

○ **B.** An extension of a Region that places select services near metro areas for single-digit millisecond latency and is opt-in per account.

○ **C.** One or more discrete data centers within a single Region, with independent power, cooling, and networking, connected by high-throughput, low-latency links.

○ **D.** A fault-isolation boundary that spans multiple Regions to provide disaster recovery across geographic areas.

Explanation→

Question 15.13

A startup plans a small web application in us-east-1 with two Amazon EC2 instances running 24×7 behind an Application Load Balancer, a NAT Gateway for outbound traffic, EBS gp3 volumes, and moderate data transfer out to the internet. Leadership wants a credible monthly cost forecast to review with stakeholders and a way to track actual spend against the forecast after launch. Which approach is the MOST appropriate?

○ **A.** Estimate only EC2 instance hours in AWS Pricing Calculator and assume Free Tier/discounts auto-apply; omit data transfer, ALB, NAT; skip budgets until after launch.

○ **B.** Price the workload in a different Region to get lower prices, ignore Multi-AZ/backup options to simplify, and let taxes/credits reconcile differences later.

○ **C.** Create AWS Budgets for the target monthly amount without an estimate; budgets will automatically calculate per-service usage and forecast monthly

cost.

○ **D.** Use AWS Pricing Calculator in us-east-1 to model EC2 (2×24×7, Graviton), EBS gp3, ALB, NAT, and data transfer; compare On-Demand vs Savings Plans; share export; then create aligned AWS Budgets.

Explanation→

Question 15.14

A company runs Amazon EC2 instances in private subnets across two Availability Zones in the same AWS Region. The instances frequently access Amazon S3 buckets in the same Region. All outbound traffic currently routes through a single NAT Gateway in one AZ. The company wants to minimize data transfer charges for this S3 access and avoid NAT Gateway processing fees. What should the company do?

○ **A.** Deploy a NAT Gateway in each Availability Zone and route each private subnet to its local NAT Gateway.

○ **B.** Use Amazon CloudFront in front of the S3 buckets so EC2 requests are served from edge locations, reducing Regional egress.

○ **C.** Assign Elastic IP addresses to the EC2 instances so they can access S3 directly without NAT.

○ **D.** Create a VPC gateway endpoint for Amazon S3 and update private subnet route tables to use it.

Explanation→

Question 15.15

A regulated insurer stores sensitive claim documents in Amazon S3 and processes them with AWS Lambda. An audit requires two specific controls: (1) all objects must be encrypted at rest with keys the company manages, and (2) only encrypted connections may access the bucket. Under the shared responsibility model, which actions are the company responsible for implementing to meet these requirements?

☐ **A.** Rely on Amazon S3 defaults to enforce TLS automatically without a bucket policy or code changes.

☐ **B.** Depend on AWS to patch both the AWS Lambda runtime and your function's third-party libraries.

☐ **C.** Enable S3 Block Public Access at the account level to prevent public reads.

☐ **D.** Turn on AWS CloudTrail and Amazon GuardDuty across accounts and centralize logs to Amazon S3.

☐ **E.** Configure Amazon S3 server-side encryption with AWS KMS (SSE-KMS) using a customer managed key and a restrictive key policy.

☐ **F.** Add an Amazon S3 bucket policy that denies requests not using TLS (require HTTPS/TLS-only access).

Explanation→

Question 15.16

Which VPC traffic-control feature automatically permits response packets for a connection that was allowed inbound, because it tracks connection state?

○ **A.** Network ACL (subnet-level, stateless)

○ **B.** VPC route table (routing decision)

○ **C.** Internet gateway (connectivity to the internet)

○ **D.** Security group (instance-level, stateful)

Explanation→

Question 15.17

Which cost is typically a direct, recurring expense for operating an on-premises data center but is usually covered by the cloud provider when you move workloads to AWS?

○ **A.** Subscription fees for managed database services

○ **B.** Electricity for running servers and cooling systems

○ **C.** Operator time to write application code

○ **D.** Network bandwidth charges for internet egress

Explanation→

Question 15.18

A company needs a disaster recovery setup in a second AWS Region that runs a smaller but ready copy of their production system to take over quickly during an outage. Which recovery topology best matches this requirement?

○ **A.** A minimal core environment pre-created and scaled up only after failover

○ **B.** Full duplicate infrastructure actively serving live traffic across Regions

○ **C.** Periodic backups copied to another Region and restored when needed

○ **D.** A reduced-capacity duplicate continuously running in the alternate Region for fast failover

Explanation→

Question 15.19

An online retailer runs a web app on a few Amazon EC2 instances in one Availability Zone and an Amazon RDS MySQL DB instance in a single AZ. A recent AZ disruption caused hours of downtime. The company wants to achieve at least 99.9% availability within one AWS Region, keep the customer experience consistent during an AZ failure, avoid changing the database engine, and minimize operational effort. Which solution is the BEST fit?

○ **A.** Use an Application Load Balancer with an EC2 Auto Scaling group across three AZs; keep the database single-AZ to reduce cost; store static assets in Amazon S3.

○ **B.** Keep EC2 instances in a single AZ for simplicity; enable Amazon RDS Multi-AZ; serve static assets from Amazon S3.

○ **C.** Place an Application Load Balancer with cross-zone load balancing in front of an EC2 Auto Scaling group spanning three AZs; enable Amazon RDS Multi-AZ; store static assets in Amazon S3.

○ **D.** Replace the relational database with Amazon DynamoDB (regional, multi-AZ); keep EC2 in one AZ; use Amazon S3 for static content.

Explanation→

Question 15.20

A startup runs model training jobs and real-time inference. They want a simple cost metric to compare and optimize different configurations. Select TWO statements that best align with using compute-hours as the primary cost measure for ML workloads.

☐ **A.** Always use the largest GPU instances to finish faster; higher per-hour rates always reduce total compute-hours cost.

☐ **B.** Favor managed analytics services that bill per query because they always produce lower compute-hour totals for model training.

☐ **C.** Move older model artifacts to cheaper object storage classes to lower compute-hour charges.

☐ **D.** Reduce billed compute-hours by shortening training time and choosing appropriately sized instances or enabling auto-scaling for inference.

☐ **E.** Ignore data egress because bandwidth and region transfers are unrelated to compute-hour calculations.

☐ **F.** Record compute-hours for each training and inference run to estimate and compare costs across instance types and durations.

Explanation→

Question 15.21

A company operates dozens of AWS accounts in AWS Organizations. The security team needs centralized, timely visibility into account-specific AWS-initiated security advisories (for example, RDS CA certificate rotations or TLS deprecations) and wants to automatically notify stakeholders and launch a standard remediation runbook. Which combination of actions best meets these requirements with minimal manual effort?

☐ **A.** Rely on the public Service health view to monitor security notifications affecting your resources.

☐ **B.** Use Amazon GuardDuty to receive AWS-initiated advisories such as RDS CA certificate rotations.

☐ **C.** Enable AWS Health Organizational View with a delegated administrator to aggregate account-specific events across all member accounts.

☐ **D.** Create an Amazon EventBridge rule for AWS Health events to publish to Amazon SNS and start an AWS Systems Manager Automation runbook.

☐ **E.** Do nothing; AWS will automatically remediate certificate rotations and TLS deprecations without customer action.

Explanation→

Question 15.22

A team deploys an AWS Lambda function to process customer uploads. Which responsibility clearly remains with the customer when using this serverless service?

○ **A.** Applying operating system patches to the virtual machines that run the Lambda functions.

○ **B.** Automatically creating and managing encryption keys for the customer's data without any configuration.

○ **C.** Scaling the function's code to meet an increase in concurrent requests without any customer input.

○ **D.** Ensuring the function's application code and the permissions it uses follow least-privilege principles.

Explanation→

Question 15.23

A company runs web applications on AWS managed services and wants to know which security task stays with their team rather than AWS. Which task is

the customer's responsibility?

○ **A.** Managing user accounts, permissions, and multi-factor authentication for the application's users.

○ **B.** Physically securing the data center buildings where the cloud servers reside.

○ **C.** Patching and maintaining the underlying hypervisor used to host virtual machines.

○ **D.** Designing and operating the global fiber network that connects AWS Regions.

Explanation→

Question 15.24

An online photo service runs a Lambda-based image API with approximately 10,000,000 invocations per month. Traffic is very steady. The average execution time is 200 ms with a 1 GB memory setting. A three-month change freeze prevents any code, memory, runtime architecture, or networking changes. Latency is acceptable; the only goal is to reduce Lambda compute charges with minimal effort.

Which is the BEST approach?

○ **A.** Enable Provisioned Concurrency to pre-initialize the function and lower duration-based costs.

○ **B.** Migrate the function to arm64 (AWS Graviton) for a lower per-ms compute rate.

○ **C.** Lower Amazon CloudWatch Logs retention to 7 days to cut Lambda costs.

○ **D.** Purchase a Compute Savings Plan sized to the steady baseline; it automatically reduces eligible Lambda compute charges.

Explanation→

Question 15.25

A retailer is building a PCI cardholder data environment on AWS using Amazon S3. They download the AWS PCI DSS Attestation of Compliance (AoC) from AWS Artifact. Their plan is to use Amazon S3 in 5 AWS Regions. The AoC lists Amazon S3 as in scope for 3 of those Regions. How many planned Region deployments are not covered by the AWS AoC and therefore require additional due diligence on provider controls?

○ **A.** 0 Regions

○ **B.** 3 Regions

○ **C.** 5 Regions

○ **D.** 2 Regions

Explanation→

Question 15.26

A development team needs an automated process on EC2 instances to read objects from a private S3 bucket. The access should be time-limited, auditable, and avoid embedding long-lived credentials on the instances. Which option is the BEST choice to grant the EC2 instances the needed permissions?

○ **A.** Attach an IAM role to the EC2 instances using an instance profile that grants read access to the bucket.

○ **B.** Create an IAM user with programmatic access and place its access key and secret on each EC2 instance.

○ **C.** Attach a resource-based policy to the S3 bucket allowing the EC2 instance ID to access objects.

○ **D.** Configure the application on the instance to call AWS STS with hard-coded IAM user credentials to obtain temporary tokens.

Explanation→

Question 15.27

An e-commerce company's login API is being hit by thousands of credential-stuffing attempts from a single public IP that reverse-resolves to an Amazon EC2 address. The company does not control that AWS account and wants AWS to investigate and stop the activity at its source. What is the BEST action to take with AWS?

○ **A.** Open a highest-severity AWS Support technical case asking AWS to disable the attacking instance.

○ **B.** Submit an abuse report to AWS Trust & Safety with the EC2 source IP, concise log excerpts, and precise UTC timestamps so AWS can notify the owning account.

○ **C.** Enable AWS Shield Advanced and add an AWS WAF rule to block the IP so AWS is alerted to take down the source.

○ **D.** Use WHOIS to identify and email the customer using that EC2 IP because AWS cannot contact other customers due to privacy.

Explanation→

Question 15.28

A web application experiences large, unpredictable spikes in user requests during the day and minimal traffic at night. The operations team wants to reduce idle compute costs while keeping performance consistent. Which is the best solution?

○ **A.** Purchase Reserved Instances sized for peak traffic to get a lower hourly rate and keep the same number of instances running.

○ **B.** Manually add and remove EC2 instances each day based on expected traffic forecasts to match capacity to demand.

○ **C.** Use Amazon EC2 Auto Scaling with a target-tracking policy that adjusts instances automatically based on a CPU or request metric.

○ **D.** Place an Elastic Load Balancer in front of fixed-size instances to distribute traffic evenly, without changing instance count.

Explanation→

Question 15.29

Which AWS service gives organizations a centralized portal to download formal audit and compliance documents (for example, security attestations and certificates) on demand to support their own regulatory reviews?

○ **A.** AWS Config

○ **B.** AWS CloudTrail

○ **C.** AWS Organizations

○ **D.** AWS Artifact

Explanation→

Question 15.30

Select TWO actions that demonstrate right-sizing of compute resources after a migration to reduce cost while maintaining performance

☐ **A.** Purchase a long-term Savings Plan to lower hourly compute costs for stable baseline usage.

☐ **B.** Use a content delivery network to cache static assets closer to users and decrease origin load.

☐ **C.** Move infrequently accessed files to a lower-cost object storage tier to reduce storage spend.

☐ **D.** Enable Auto Scaling to add and remove instances automatically in response to short-term traffic spikes.

☐ **E.** Consolidate multiple lightly used workloads onto fewer instances and adjust instance types to match combined demand.

☐ **F.** Reduce instance families or sizes for underutilized virtual machines after observing low CPU and memory metrics.

Explanation→

Question 15.31

An organization is hardening their AWS account root identity to reduce risk of account takeover. Select TWO actions that directly add a second authentication factor to the root account (Select TWO).

☐ **A.** Turn on CloudTrail to log all management API calls for the account.

☐ **B.** Register a hardware one-time-password device for the root account.

☐ **C.** Store root credentials in an encrypted secret vault and share them with the ops team.

☐ **D.** Create an administrative IAM user and use that identity for daily tasks.

☐ **E.** Apply a resource-based policy to prevent root from accessing specific services.

☐ **F.** Enable a virtual time-based MFA authenticator app for the root account.

Explanation→

Question 15.32

A startup hosts a WordPress marketing site on a single Amazon Lightsail instance. They want higher availability and HTTPS with minimal management and predictable monthly cost. They do not need Application Load Balancer features and prefer to stay on Lightsail for now.

Review the exhibit and choose the best next step.

Table 5: Exhibit: Current state and goals

Item	Current	Goal/Constraint
App	WordPress on one Lightsail instance	Stay on Lightsail; minimal changes
Networking	Static public IP; Lightsail DNS	Public HTTPS without managing certs on instance

Item	Current	Goal/Constraint
Availability	Single AZ	Survive an AZ failure; add a second instance
Cost	Low, predictable	Keep predictable monthly pricing
Advanced features	None	No path-based routing; no PrivateLink needed

○ **A.** Add a Lightsail load balancer with TLS termination and place two Lightsail instances in different Availability Zones behind it.

○ **B.** Export the instance to Amazon EC2, place it behind an Application Load Balancer, and use an Auto Scaling group across AZs.

○ **C.** Peer Lightsail to a VPC and migrate the database to Amazon RDS while keeping the single Lightsail instance with its static IP.

○ **D.** Install an SSL/TLS certificate on the Lightsail instance and open port 443 in the instance-level firewall.

Explanation→

Question 15.33

A company will allow a third-party SaaS vendor to write backups to an Amazon S3 bucket in the company's AWS account. The company will use a cross-account IAM role for delegation. Which TWO configuration choices are NOT recommended?

☐ **A.** Use short role session durations appropriate for automation, and require MFA when humans assume the role.

☐ **B.** Configure the role's trust policy with Principal set to "*" so any account can assume it.

☐ **C.** Monitor role assumptions with AWS CloudTrail and create CloudWatch alarms on unusual activity.

☐ **D.** Share long-term access keys with the vendor instead of using AWS STS AssumeRole temporary credentials.

☐ **E.** Attach a permissions policy that allows only s3:PutObject to the designated backup bucket.

☐ **F.** Add a Condition in the trust policy requiring sts:ExternalId with a unique value shared with the vendor.

Explanation→

Question 15.34

Which AWS service is commonly relied on as the primary tamper-evident source of API call records when an auditor needs evidence of who performed management-plane actions?

○ **A.** Amazon CloudWatch Logs — stores application and system logs for real-time metrics and alarms.

○ **B.** AWS Config — tracks resource configuration changes and provides a history of resource states.

○ **C.** AWS CloudTrail — records account API activity and can provide tamper-evident event logs for auditing.

○ **D.** Amazon GuardDuty — provides threat detection and alerts based on telemetry and anomalies.

Explanation→

Question 15.35

A company migrating to AWS needs to standardize resource tagging, allocate costs by business unit, enforce change control, and set guardrails using service control policies. According to the AWS Cloud Adoption Framework (AWS CAF), which perspective should lead this work?

○ **A.** Platform perspective

○ **B.** Governance perspective

○ **C.** Security perspective

○ **D.** Operations perspective

Explanation→

Question 15.36

Which statement best describes the IAM condition key aws:MultiFactorAuthPresent?

○ **A.** It evaluates to true only when the request uses MFA-authenticated credentials, allowing you to require MFA for console and AWS STS sessions.

○ **B.** It sets a maximum time since the last MFA verification to limit session duration.

○ **C.** It enables MFA for all users at AWS IAM Identity Center sign-in by itself.

○ **D.** It automatically replaces all long-lived IAM user access keys with short-lived credentials.

Explanation→

Question 15.37

Select TWO benefits that are provided when an organization uses AWS Marketplace consolidated billing for third-party software purchases

☐ **A.** Purchases through Marketplace always include automatic software updates managed by AWS.

☐ **B.** Marketplace charges are combined with other AWS service costs to simplify internal chargeback and cost allocation.

☐ **C.** AWS Marketplace issues separate invoices per vendor to help reconcile each supplier individually.

☐ **D.** All Marketplace software charges appear on a single AWS invoice for centralized payment.

☐ **E.** Marketplace automatically deploys vendor images into a private network using PrivateLink.

☐ **F.** Marketplace converts all license types to a single universal license managed by AWS.

Explanation→

Question 15.38

A company needs interactive BI dashboards that run complex SQL joins across billions of rows with high user concurrency during business hours. Most historical data is stored as partitioned Parquet files in Amazon S3, while current-day transactions reside in Amazon Aurora. The team wants sub-second to seconds-level response times, to absorb short-lived peaks without overprovisioning, and to avoid cluster management while paying only for processing used. Which approach best meets these goals?

○ **A.** Amazon Athena to run serverless SQL directly on S3; pay-per-query for low operations overhead and ad hoc analytics.

○ **B.** Provisioned Amazon Redshift RA3 cluster using Concurrency Scaling and Redshift Spectrum to balance performance and data lake access.

○ **C.** Amazon RDS for PostgreSQL with read replicas to offload reporting from production and handle analytics at scale.

○ **D.** Amazon Redshift Serverless with Redshift Spectrum to query S3 and federated queries to join Aurora; automatic scaling and no cluster management for consistent, low-latency analytics.

Explanation→

Question 15.39

A company pointed its website traffic to an Amazon CloudFront distribution to reduce latency and lighten load on the origin. After deployment, the origin server still receives nearly every user request and global users report unchanged latency. Which is the most likely reason CloudFront is not reducing origin traffic as expected?

○ **A.** The CloudFront distribution is deployed only to a single Availability Zone, so most users reach the origin instead of an edge location.

○ **B.** The distribution was created without an origin configured, so CloudFront cannot cache content and always contacts the origin.

○ **C.** Edge locations are geographically distant from users, so requests are routed to the origin instead of being served from cache.

○ **D.** The origin's responses include headers that instruct caches not to store content, so CloudFront forwards each request to the origin.

Explanation→

Question 15.40

A media company must migrate 600 TB of archived video from its on-premises data center to Amazon S3 in the us-east-1 Region within 10 days. The network team can allocate at most 200 Mb/s sustained bandwidth for the migration. The data consists of millions of small files. The security team requires encryption using AWS KMS keys and a tamper-resistant, trackable process. Which is the BEST solution to meet the deadline and requirements?

○ **A.** Order multiple AWS Snowball Edge Storage Optimized devices for offline transfer to Amazon S3; perform parallel local copies and let AWS import the data. Data is encrypted with AWS KMS and devices have end-to-end tracking.

○ **B.** Use AWS DataSync over the existing 200 Mb/s internet link to transfer directly to Amazon S3 during off-peak hours.

○ **C.** Provision a new 1 Gbps AWS Direct Connect and use AWS DataSync to migrate the data to Amazon S3 within the 10-day window.

○ **D.** Request multiple AWS Snowcone devices and run AWS DataSync on them to seed data, then trickle incremental changes over the limited link.

Explanation→

Question 15.41

A small e-commerce site expects occasional large traffic spikes that could be simple volumetric DDoS events. Which AWS offering automatically provides

baseline detection and mitigation for these common network floods without extra enrollment?

○ **A.** AWS Shield Advanced — enhanced mitigation with near-real-time reporting and cost protection.

○ **B.** AWS Shield Standard — built-in, always-active protection against typical volumetric DDoS floods.

○ **C.** AWS WAF — a managed web application firewall that automatically blocks volumetric network floods.

○ **D.** Security groups — stateful host-level filters that inherently stop large-scale DDoS traffic.

Explanation→

Question 15.42

A web application experiences variable traffic spikes daily. The team wants to maintain fast response times during peaks while avoiding high costs when traffic is low. Which approach best balances performance and cost?

○ **A.** Provision a fixed large fleet of instances sized for peak load to ensure headroom.

○ **B.** Use only Spot Instances without scaling to minimize cost regardless of availability.

○ **C.** Deploy a single very large instance and rely on its capacity to handle all traffic.

○ **D.** Configure Auto Scaling to add and remove instances based on real-time load.

Explanation→

Question 15.43

A company runs Amazon EC2 instances in private subnets within a custom Amazon VPC. The instances must download objects from Amazon S3 and call a third-party SaaS API without sending any traffic over the public internet. The company wants all communication to stay within the VPC using private connectivity. Which approach best meets these requirements?

○ **A.** Deploy NAT Gateways in each Availability Zone and route private subnets through them to reach S3 and the SaaS API over the internet.

○ **B.** Attach an Internet Gateway and assign public IPs to the instances so they can access S3 and the SaaS API directly.

○ **C.** Use a Gateway VPC endpoint for Amazon S3 and an AWS PrivateLink

interface endpoint to the SaaS; update routes and security groups for private access.

○ **D.** Use a Gateway VPC endpoint for S3 and a NAT Gateway for the SaaS API; keep instances in private subnets.

Explanation→

Question 15.44

A company wants a simple way for customer service staff to answer calls and chats from any location without installing client software. Which AWS service feature best provides an in-browser agent console for handling customer interactions?

○ **A.** Amazon Chime SDK — a real-time communication toolkit for building custom audio/video applications.

○ **B.** AWS Systems Manager Session Manager — a browser-based shell for managing servers remotely.

○ **C.** Amazon Connect Contact Flows — configurable rules that direct incoming interactions to endpoints.

○ **D.** Amazon Connect Contact Control Panel — a web-based agent console for accepting and managing calls, chats, and tasks.

Explanation→

Question 15.45

Which Amazon SES feature uses a cryptographic signature added to outgoing messages so recipients can confirm the message was sent by the claimed domain and has not been tampered with?

○ **A.** Sender Policy Framework (SPF) DNS record

○ **B.** DomainKeys Identified Mail (DKIM) signing

○ **C.** Dedicated sending IP address

○ **D.** Account suppression list

Explanation→

Question 15.46

A security analyst notices several management API calls from an IAM role at unusual hours. Which Amazon GuardDuty capability most directly helps detect this as abnormal by comparing activity to each resource's normal usage patterns?

○ **A.** Matching API calls against a curated blocklist of known malicious IP addresses.

○ **B.** Relying on static signature rules like traditional antivirus to identify threats.

○ **C.** Scanning Amazon S3 buckets for public objects to find exposed data.

○ **D.** Using learned normal activity profiles (behavioral baselining) to flag deviations.

Explanation→

Question 15.47

A regulated enterprise is moving to a multi-account strategy using AWS Organizations. The security team must ensure that across all current and future accounts:

- AWS CloudTrail cannot be stopped or deleted
- Amazon S3 Block Public Access cannot be disabled at the account or bucket level

They want a centralized, preventive control with minimal ongoing operations. Which is the BEST solution?

○ **A.** Attach IAM permission boundaries to developer roles to block CloudTrail and S3 public access changes across accounts.

○ **B.** Deploy AWS Config rules and conformance packs to auto-remediate when CloudTrail or S3 Block Public Access is changed.

○ **C.** Create Amazon CloudWatch alarms for CloudTrail and S3 configuration changes to notify security for manual remediation.

○ **D.** Use AWS Organizations SCPs to deny stopping/deleting CloudTrail and changing S3 Block Public Access, attached at the org root or OUs for all accounts.

Explanation→

Question 15.48

A company wants to minimize long-term credentials and apply least privilege.

- An application running on Amazon EC2 must read objects from a private Amazon S3 bucket.
- External contractors authenticate with the company's identity provider and need short-term AWS Management Console access to selected services for a 3-month project.

Which TWO actions meet these goals using AWS best practices?

☐ **A.** Create an IAM user for the EC2 application with access keys; store the keys on the instance and rotate them every 90 days.

☐ **B.** Attach an IAM role to the EC2 instance using an instance profile that allows s3:GetObject; the application uses temporary credentials issued by AWS STS.

☐ **C.** Set up workforce federation so contractors authenticate with the external IdP and assume IAM roles via AWS STS; attach permission policies to those roles.

☐ **D.** Create individual IAM users for contractors with console passwords and add them to an IAM group that has the required permissions.

☐ **E.** Provide contractors the IAM role ARN and have them sign in directly as the role using the AWS sign-in URL; no separate user is required.

☐ **F.** Create an IAM group for the EC2 instance and attach policies so the instance can authenticate as the group to access S3.

Explanation→

Question 15.49

Which AWS service caches objects at edge locations to reduce origin latency for read-heavy global content when compute runs in a single Region?

○ **A.** Amazon Route 53 latency-based routing

○ **B.** Amazon CloudFront

○ **C.** AWS Global Accelerator

○ **D.** Amazon S3 Cross-Region Replication

Explanation→

Question 15.50

Which type of security control is primarily responsible for recording user and API activity so administrators can spot unusual behavior and perform audits (for example, using AWS CloudTrail)?

○ **A.** Preventive control — blocks access and enforces strict configuration to stop attacks.

○ **B.** Corrective control — removes threats and returns systems to a trusted state.

○ **C.** Detective control — captures and logs activity to identify anomalous behavior.

○ **D.** Compensating control — replaces technical controls with non-technical processes temporarily.

Explanation→

Question 15.51

A startup reviewed its three-tier web application in the AWS Well-Architected Tool. Leadership set a measurable reliability target of RTO = 15 minutes for the database tier and wants clear evidence of posture improvement as recommendations are implemented. Which actions should the team prioritize to directly connect these goals to architecture decisions and show progress over time?

☐ **A.** Adopt Savings Plans for steady EC2 usage to make costs predictable and demonstrate reliability improvements.

☐ **B.** Enable S3 Block Public Access and SSE-KMS to meet the database RTO requirement.

☐ **C.** Configure AWS Backup to take manual weekly snapshots of the database.

☐ **D.** Add Amazon CloudWatch CPU alarms on the database to meet the 15-minute RTO.

☐ **E.** Place the web tier in an Auto Scaling group behind an Application Load Balancer with health checks to achieve the 15-minute database RTO.

☐ **F.** Enable Amazon RDS Multi-AZ for the database to improve reliability toward the 15-minute RTO target.

☐ **G.** Create a Well-Architected Tool milestone after implementing changes to snapshot the architecture and compare risk reduction over time.

Explanation→

Question 15.52

An application uses Amazon RDS and must remain available with minimal downtime during an Availability Zone disruption. The team is planning their approach. Select TWO actions they should AVOID.

☐ **A.** Running periodic failover drills to validate application behavior with the Multi-AZ standby

☐ **B.** Enabling Multi-AZ so a synchronized standby exists in another Availability Zone with automatic failover

☐ **C.** Monitoring RDS events and metrics to confirm when a Multi-AZ failover occurs

☐ **D.** Planning for automatic failover to the standby during an Availability Zone disruption

☐ **E.** Placing the primary and its standby in the same Availability Zone to reduce inter-AZ costs

☐ **F.** Relying on asynchronous read replicas instead of Multi-AZ for availability and failover

Explanation→

Question 15.53

A company must decide who is responsible for operating system patching for two deployment options: a virtual server on Amazon EC2 and a managed database on Amazon RDS. Which statement best describes the division of responsibility for OS patching?

○ **A.** The customer is responsible for OS patching on both Amazon EC2 and Amazon RDS instances.

○ **B.** AWS patches the OS for EC2 instances, but the customer patches the OS for Amazon RDS.

○ **C.** AWS applies operating system patching for Amazon RDS, while the customer is responsible for patching the OS on Amazon EC2 instances.

○ **D.** Neither AWS nor the customer patches the OS; patching is automatic and not a shared responsibility.

Explanation→

Question 15.54

In a migration business case built with AWS Migration Evaluator, which formula calculates the payback period P in years given the one-time migration cost and annual savings?

○ **A.** $P = C_{\text{migrate}} / S$

○ **B.** $P = S / C_{\text{migrate}}$

○ **C.** $P = C_{\text{on-prem, annual}} - C_{\text{AWS, annual}}$

○ **D.** $P = C_{\text{AWS, annual}} / C_{\text{on-prem, annual}}$

Explanation→

Question 15.55

Which two statements describe how resource tags help implement internal showback or chargeback reporting?

☐ **A.** Tags let teams attach owner and environment labels to resources so costs can be reported back to the responsible group.

☐ **B.** Tagging enforces encryption of stored data, ensuring billed storage is always encrypted before allocation.

☐ **C.** Tags replace the need for cost reports by deleting untagged resources automatically to avoid unallocated costs.

☐ **D.** Tags are only useful for access control and cannot be used to group cost data for internal reporting.

☐ **E.** Applying tags directly to billing invoices is required for chargeback to work because invoices accept only tagged line items.

☐ **F.** Consistent tagging enables automated grouping of resources so finance can allocate spending to business units for internal billing.

Explanation→

Question 15.56

Which statement correctly describes how the AWS Management Console, the AWS CLI, and AWS SDKs relate when interacting with AWS services?

○ **A.** The Management Console uses a different authentication system than the CLI and SDKs, so permissions must be configured separately.

○ **B.** The AWS CLI is only for manual interactive use and cannot be used for scripted automation or batch operations.

○ **C.** They all send requests to the same AWS service APIs and rely on IAM for authentication and authorization.

○ **D.** SDKs bypass IAM and authenticate directly to services using language-level credentials embedded in application code.

Explanation→

Question 15.57

Which capability best describes AWS Firewall Manager's primary purpose within an organization?

○ **A.** It provides per-instance host-based intrusion detection that inspects file system activity on EC2 instances.

○ **B.** It serves as a managed DNS service to route user traffic to the nearest edge location.

○ **C.** It centrally deploys and enforces uniform network and application protection policies across multiple accounts.

○ **D.** It is a centralized key management system for creating and rotating encryption keys across services.

Explanation→

Question 15.58

A company begins migrating workloads to AWS and wants to make sure teams can operate and support the new cloud environment. Which approach best focuses on developing the needed staff capabilities during adoption?

○ **A.** Create targeted, role-based training and assess skills gaps before each migration phase.

○ **B.** Require a single-day, company-wide cloud overview for all employees and consider the work done.

○ **C.** Immediately reassign existing staff away from cloud projects and hire consultants to run operations indefinitely.

○ **D.** Delay any training until after migration completes to avoid distracting delivery teams.

Explanation→

Question 15.59

Which outcomes are provided by organizing approved IT services into a portfolio in AWS Service Catalog?

☐ **A.** Defines which users or groups can access a curated set of preapproved products

☐ **B.** Replaces IAM by storing user credentials required to launch resources

☐ **C.** Provides a built-in cost-reporting dashboard with detailed billing for each product

☐ **D.** Groups multiple product templates so administrators can manage approvals and visibility centrally

☐ **E.** Automatically converts on-premises servers into managed AWS instances

Explanation→

Question 15.60

Which of the following best defines data classification as a customer responsibility under the shared responsibility model?

○ **A.** Managing AWS infrastructure hardware lifecycle, including physical server maintenance in a region.

○ **B.** Issuing service-level encryption for all AWS managed service backups automatically.

○ **C.** Assigning sensitivity levels to datasets and specifying handling requirements for each category.

○ **D.** Configuring AWS physical network cabling between Availability Zones.

Explanation→

Question 15.61

A team is deploying a web app on AWS Elastic Beanstalk and wants capacity to adjust automatically. Which actions are INCORRECT for using Elastic Beanstalk's built-in scaling?

☐ **A.** Configure scaling rules to add or remove instances based on average CPU utilization.

☐ **B.** Use the environment's scaling settings to set minimum and maximum instance counts.

☐ **C.** Edit the Auto Scaling group directly in the EC2 console to override environment settings long-term.

☐ **D.** Disable the platform's scaling and manually start/stop EC2 instances over SSH to control capacity.

☐ **E.** Set a reasonable maximum instance limit to cap scale-out capacity.

Explanation→

Question 15.62

A media company stores large video files in Amazon S3. The security policy mandates encryption at rest, but the team wants the lowest operational overhead and does not need to control key policies or grants. Which option best meets the requirement?

○ **A.** Amazon S3 server-side encryption (SSE-KMS) using a customer managed KMS key

○ **B.** Client-side encryption using customer managed keys in AWS KMS before uploading to S3

○ **C.** Use TLS for Amazon S3 service endpoints only

○ **D.** Amazon S3 server-side encryption (SSE-S3) with AWS-managed keys

Explanation→

Question 15.63

A company runs identical web applications in two AWS Regions behind Application Load Balancers. The company wants users to be directed to the Region with the lowest network latency, and Route 53 must automatically stop

returning answers for any Region whose endpoints become unhealthy. Which Route 53 features should the company use?

☐ **A.** Weighted routing policy with equal weights across Regions.

☐ **B.** Geolocation routing policy based on user country or continent.

☐ **C.** Route 53 health checks associated with DNS records to remove unhealthy endpoints from responses.

☐ **D.** Latency-based routing policy in Amazon Route 53 to direct users to the lowest-latency Region.

☐ **E.** Multi-value answer routing to return multiple healthy IPs to clients.

☐ **F.** Failover routing policy with a primary Region and a secondary Region.

Explanation→

Question 15.64

A company uses a template-based deployment system to create identical environments. Which outcome best illustrates the property where applying the same template multiple times leaves the environment unchanged after the first successful run?

○ **A.** Storing all templates in a versioned repository for history and rollback.

○ **B.** Deterministic deployments that yield the same final state when run repeatedly.

○ **C.** Breaking templates into smaller files so teams can reuse parts across projects.

○ **D.** Running automated checks to validate templates before creating resources.

Explanation→

Question 15.65

A startup runs a MySQL database on a single Amazon EC2 instance. Traffic is unpredictable with sudden spikes. The team wants to reduce operational burden (patching, backups), improve availability, and pay mostly for the database capacity actually consumed. The solution must handle bursts automatically, provide Multi-AZ durability with near-instant failover, enable point-in-time recovery and encryption, and require minimal application changes. Which is the BEST solution?

○ **A.** Amazon RDS for MySQL with Multi-AZ, read replicas, and Performance Insights; improves availability and visibility but uses fixed instances and manual scaling.

○ **B.** Amazon DynamoDB with on-demand capacity and global tables; scales automatically but requires moving from relational MySQL to NoSQL data modeling.

○ **C.** Self-managed MySQL on Amazon EC2; build your own replication, backups, patching, monitoring agents, and recovery runbooks.

○ **D.** Amazon Aurora Serverless v2 (MySQL-compatible) with Multi-AZ, automated backups, KMS encryption; scales capacity in fine-grained units to handle spikes and reduce idle cost.

Explanation→

Exam 15 — Answer Key

Question	Correct answer(s)
15.1	C
15.2	C
15.3	A
15.4	C
15.5	A, D
15.6	B, D
15.7	D
15.8	B, E
15.9	D
15.10	C
15.11	D
15.12	C
15.13	D
15.14	D
15.15	E, F
15.16	D
15.17	B
15.18	D
15.19	C
15.20	D, F
15.21	C, D
15.22	D
15.23	A
15.24	D
15.25	D
15.26	A
15.27	B

Question	Correct answer(s)
15.28	C
15.29	D
15.30	E, F
15.31	B, F
15.32	A
15.33	B, D
15.34	C
15.35	B
15.36	A
15.37	B, D
15.38	D
15.39	D
15.40	A
15.41	B
15.42	D
15.43	C
15.44	D
15.45	B
15.46	D
15.47	D
15.48	B, C
15.49	B
15.50	C
15.51	F, G
15.52	E, F
15.53	C
15.54	A
15.55	A, F
15.56	C
15.57	C

Question	Correct answer(s)
15.58	A
15.59	A, D
15.60	C
15.61	C, D
15.62	D
15.63	C, D
15.64	B
15.65	D

Exam 15 Explanations

15.1 — Observability for operational excellence (D1.T1.2)

Domain 1 • Task 1.2 ↑ Back to Question 15.1

A web application occasionally experiences slow responses. Which observability data type best helps you follow a single request across services to find which component added most latency?

Options & Rationales

A ✗ — Metrics — provide aggregated numerical measures like average latency over time.

Rationale: Metrics give summarized measurements (for example, averages or percentiles) and are useful for detecting trends, but they do not show the detailed component-by-component path of an individual request.

B ✗ — Logs — contain immutable event records for auditing and debugging.

Rationale: Logs record events and messages useful for diagnosing issues, but they typically require stitching together multiple entries to reconstruct a request flow and lack the structured timing across services that traces give.

C ✓ — Traces — show an end-to-end request path with timing between components.

Rationale: Traces provide a per-request view that records each service hop and timing, making it possible to locate where latency accumulates along the call path.

D ✗ — Alerts — notify operators when thresholds are crossed so they can restart services.

Rationale: Alerts inform you that a problem exists by monitoring metrics or logs, but they do not themselves provide the detailed per-request path or timing information needed to pinpoint the component causing latency.

Explanation

Observability helps teams detect and diagnose operational issues by collecting metrics, logs, and traces. Traces capture the sequence of calls a single request makes across components and record timing for each hop; this per-request perspective is ideal for finding where latency is introduced. Metrics are time-series numbers (for example, average or p95 latency) that surface trends and trigger alerts but do not show the internal sequence of service interactions. Logs are immutable entries describing events and errors, useful for detailed context and auditing, yet reconstructing a full request path from logs is often manual and time-consuming. Alerts are notifications based on metrics or log

conditions and prompt investigation but do not contain the diagnostic path information themselves.

Key Concepts

- **Traces:** Provide an end-to-end view of a single request, including the timing and order of interactions between components, which helps identify latency hotspots.
- **Metrics:** Time-series numerical summaries useful for trend detection and threshold-based alerting, not per-request sequencing.
- **Logs:** Detailed event records useful for context and debugging, but require correlation to reconstruct request journeys.

Common Pitfalls

- Confusing aggregated latency metrics with the detailed per-request timings that traces provide can lead to inefficient troubleshooting.
- Expecting alerts or logs alone to reveal which component added latency often results in longer mean time to recovery because they lack the structured end-to-end timing that traces supply.

15.2 — Consolidated billing benefits (D4.T4.2)

Domain 4 • Task 4.2 ↑ Back to Question 15.2

A company uses AWS Organizations to link several member accounts under one paying account. Which statement best describes how usage-based volume pricing discounts are determined across the organization?

Options & Rationales

A ✗ — Each linked account is evaluated separately for volume discounts; aggregation does not apply.

Rationale: Incorrect because linked accounts' usage is combined rather than treated fully independently for volume pricing.

B ✗ — Only the management account's usage qualifies for volume discounts; member accounts are excluded.

Rationale: Incorrect because discounts consider aggregated usage across the organization, not only the management account.

C ✓ — Usage across linked accounts is combined so the organization can reach higher volume pricing tiers.

Rationale: Aggregated usage from linked accounts counts toward volume thresholds, enabling lower unit pricing for the organization.

D ✗ — Volume discounts apply only if every linked account has identical service usage patterns.

Rationale: Incorrect because differing usage patterns do not prevent aggregated usage from qualifying for pricing tiers.

Explanation

When accounts are grouped within an organization, their resource consumption can be combined to determine eligibility for usage-based price breaks. This means total consumption across linked accounts is measured together to reach thresholds that unlock lower unit rates.

Why the correct statement fits. The correct statement says that the accounts' usage is combined to reach higher volume pricing tiers. That describes the organization-level aggregation where combined consumption counts toward pricing thresholds so the whole organization benefits from lower per-unit prices once a threshold is met.

Why the other statements are wrong. Saying each account is evaluated separately ignores aggregation and would prevent many organizations from reaching discounts that are intended to be shared. Suggesting only the management (paying) account counts is incorrect because member account consumption contributes to the aggregate. Requiring identical usage patterns across accounts is irrelevant; differing patterns do not stop totals being summed for pricing.

Key Concepts

- **Volume aggregation:** Combining consumption across linked accounts to reach pricing thresholds that may reduce unit costs.
- **Management (payer) account:** The account that receives and pays the combined invoice for the organization, but not the sole source of usage for pricing calculations.
- **Member accounts:** Individual linked accounts whose usage contributes to the organization's aggregated totals.

Common Pitfalls

- Assuming discounts apply per account rather than across the organization will underestimate potential savings.
- Confusing who pays the invoice with which usage counts toward pricing thresholds.

15.3 — Dedicated Hosts vs Dedicated Instances (D4.T4.1)

Domain 4 • Task 4.1 ↑ Back to Question 15.3

Which statement best defines Amazon EC2 Dedicated Hosts for compliance and licensing scenarios?

Options & Rationales

A ✓ — An entire physical server allocated to one account that exposes sockets, cores, and a host ID, enabling host affinity and BYOL; pricing is per host.

Rationale: Dedicated Hosts provide host-level visibility and control to map workloads to specific physical resources and meet per-socket/core licensing. You pay for the whole host.

B ✗ — Instances that run on single-tenant hardware managed by AWS placement, billed per instance, with no visibility into sockets/cores or guaranteed host affinity.

Rationale: This describes Dedicated Instances, which provide isolation but not host-level attributes or control of placement.

C ✗ — A VPC tenancy setting that makes all new instances dedicated and provides control over specific physical servers for licensing and audits.

Rationale: Setting VPC tenancy to "dedicated" defaults launches to Dedicated Instances only; it does not grant host-level control or identification.

D ✗ — A billing model where you pay only for instance capacity while still mapping workloads to specific sockets/cores for per-socket licensing.

Rationale: Per-socket/core mapping requires Dedicated Hosts with per-host pricing; per-instance billing lacks host-level visibility and control.

Explanation

Amazon EC2 offers two single-tenant options for isolation-driven compliance: Dedicated Instances and Dedicated Hosts. Both avoid multi-tenant compute, but they differ in control, visibility, and billing.

Dedicated Instances run on hardware that is physically dedicated to one account. AWS controls placement across hosts. You are billed per instance and do not see or control underlying sockets/cores. There is no guaranteed instance-to-host affinity. This fits compliance-only isolation when you do not need to tie workloads to specific physical servers.

Dedicated Hosts allocate an entire physical server to your account and expose host-level attributes such as sockets, cores, and a host ID. This enables host affinity and is essential for bring-your-own-license (BYOL) models that count per-socket, per-core, or per-VM, and for audits requiring identification of physical servers. Pricing is per host, so you pay for the full server and must manage utilization carefully.

A related pitfall is setting a VPC's tenancy to "dedicated," which causes new instances in that VPC to launch as Dedicated Instances by default. This does not provide host-level control and can lead to unexpected ongoing costs if overlooked.

Therefore, the description that highlights a physical server allocated to your account with sockets/cores visibility, host ID, host affinity, BYOL support, and per-host pricing correctly defines Dedicated Hosts. The description of single-tenant instances with per-instance billing and no host visibility is Dedicated Instances. The statement about a VPC tenancy setting incorrectly implies host control. The idea that you can pay per instance while still mapping to sockets/cores contradicts how host-level visibility works.

Key Concepts

- **Dedicated Instances:** Single-tenant hardware with AWS-managed placement; per-instance billing; no socket/core visibility or host affinity.
- **Dedicated Hosts:** Entire physical server to one account; exposes sockets/cores and host ID; supports host affinity and BYOL; per-host pricing.
- **VPC "dedicated" tenancy:** Makes new instances Dedicated Instances by default; does not grant host-level control and can cause surprise costs.
- **Fit-for-purpose choice:** Use Dedicated Instances for isolation-only; use Dedicated Hosts when audits or BYOL require mapping to specific physical servers.

Common Pitfalls

Confusing Dedicated Instances with Dedicated Hosts or assuming VPC "dedicated" tenancy provides host control. Ignoring that Dedicated Hosts bill per host, which can lead to paying for idle capacity.

15.4 — Athena for interactive S3 querying (D3.T3.7)

Domain 3 • Task 3.7 ↑ Back to Question 15.4

A team stores large, raw analytics files in Amazon S3 and needs to run ad hoc SQL queries without provisioning or managing query servers. Which AWS service best meets this requirement?

Options & Rationales

A ✗ — Amazon Redshift — a provisioned data warehouse optimized for complex, high-performance analytics with cluster management.

Rationale: Redshift requires provisioning and managing a cluster (or using managed serverless modes for different use cases) and is designed as a data warehouse rather than ad hoc SQL over S3 objects without infrastructure.

B ✗ — Amazon EMR — a managed big-data platform for running distributed processing frameworks such as Spark or Hadoop on provisioned clusters.

Rationale: EMR is intended for distributed data processing and typically involves cluster provisioning and management, which contradicts the requirement to avoid managing query servers.

C ✓ — Amazon Athena — serverless SQL queries run directly over data in Amazon S3 without managing query infrastructure.

Rationale: Athena is a serverless interactive SQL query service that lets you query data in S3 directly without provisioning or managing query servers.

D ✗ — Amazon RDS — managed relational databases for transactional workloads that require long-running database instances and schema management.

Rationale: RDS provides managed relational databases but relies on continuously running database instances and predefined schemas, making it unsuitable for ad hoc, serverless SQL directly against S3 objects.

Explanation

The requirement is ad hoc SQL querying over data stored in Amazon S3 without provisioning or managing query servers. Athena is designed to let users run interactive SQL statements directly against objects in S3 while the service handles query execution and scaling. This avoids the need to create, size, or maintain query infrastructure.

Why the correct choice fits. The correct service provides serverless SQL query capability over S3 objects and scales automatically, enabling ad hoc analytics without infrastructure management.

Why the other options are not suitable:

- The data warehouse service requires provisioning and operating a cluster (or choosing a different architecture) and is focused on structured, high-performance warehousing rather than immediate ad hoc queries on S3 objects with no server management.
- The managed big-data processing platform is intended for distributed processing frameworks and typically involves cluster management; it is more operationally intensive than a serverless query option.
- The managed relational database service supports transactional workloads with persistent database instances and schema-first design; it is not intended for directly querying raw objects in S3 without loading or managing database servers.

Key Concepts

- **Serverless query service:** A service that executes queries without requiring customers to provision or manage underlying servers, handling scaling and availability for query execution.
- **Schema-on-read:** An approach where the data structure is interpreted at query time rather than requiring a predefined schema before data is stored.

Common Pitfalls

- Assuming a managed relational database or data warehouse is automatically serverless; many such services typically require provisioning and management unless a specific serverless mode is chosen.
- Confusing distributed processing platforms with interactive SQL query services; they serve different operational and use-case needs.

15.5 — Sustainability design: minimize idle resources (D1.T1.2)

Domain 1 • Task 1.2 ↑ Back to Question 15.5

A company wants to improve sustainability by minimizing idle compute. The workload runs infrequently and is event-driven. Which actions are INCORRECT for this goal? Select TWO to AVOID.

Options & Rationales

A ✓ — Keep large Amazon EC2 instances running 24/7 for occasional tasks to prevent cold starts.

Rationale: Maintaining always-on servers for sporadic work leaves capacity idle, which contradicts using serverless options to avoid provisioned idle resources.

B ✗ — Migrate the processing to AWS Lambda and trigger it only on relevant events.

Rationale: Lambda runs code on demand with no standing servers, aligning with avoiding idle capacity.

C ✗ — Use AWS Fargate so containers run only when tasks are invoked.

Rationale: Fargate is a serverless container compute model that eliminates managing and idling host servers.

D ✓ — Manually pre-provision an ECS cluster on EC2 and keep services running at all times for a weekly job.

Rationale: Pre-provisioned EC2 hosts for infrequent jobs create idle servers, violating the goal to remove provisioned idle capacity with serverless.

E ✗ — Use AWS Step Functions to coordinate event-driven serverless tasks without running persistent servers.

Rationale: Step Functions is a serverless orchestrator and does not require always-on servers, helping avoid idle compute.

Explanation

Sustainability in cloud design emphasizes reducing wasted compute to lower energy use and carbon impact. One effective approach is adopting serverless compute. With serverless, the provider runs your code or containers only when needed, so you do not keep virtual machines or hosts running during idle periods. This directly addresses idle capacity, which is a primary source of inefficiency.

The action that keeps large Amazon EC2 instances running continuously for occasional tasks is wasteful because it leaves provisioned servers idle most of the time. Likewise, manually pre-provisioning an Amazon ECS cluster on EC2 and keeping services running for a weekly job maintains persistent hosts with little utilization between runs. Both choices violate the goal of minimizing idle compute because they rely on always-on capacity.

By contrast, moving processing to AWS Lambda executes code only in response to events, avoiding standing servers. Running containers on AWS Fargate similarly removes the need to manage or idle EC2 hosts; tasks consume resources only while running. Using AWS Step Functions to coordinate event-driven tasks also avoids persistent infrastructure and complements a serverless approach.

Key Concepts

- **Serverless compute:** Runs code or containers on demand without provisioning or managing servers, reducing idle capacity.
- **Idle resources:** Provisioned instances or hosts that sit unused consume energy without delivering value.
- **Event-driven execution:** Triggers workload only when events occur, aligning usage with demand and lowering waste.

Common Pitfalls

- Keeping instances online "just in case" to avoid cold starts, which leads to persistent idle resources and undermines sustainability goals.

15.6 — VPC peering use cases & limits (D3.T3.5)

Domain 3 • Task 3.5

↑ Back to Question 15.6

Your team plans to connect three VPCs (VPC-A, VPC-B, VPC-C) so workloads can communicate privately. Which TWO statements describe correct behavior or a required action when using VPC peering?

Options & Rationales

A ✗ — VPC peering automatically allows routing through chained peers so route tables do not need updates.

Rationale: Peering is not transitive and does not auto-propagate routes; route tables still need explicit entries for each direct peering connection.

B ✓ — A peered VPC will not forward packets to another VPC on your behalf; it cannot act as a transit hop.

Rationale: Peering does not support forwarding traffic between peers; packets are only delivered between the two endpoints of a peering link.

C ✗ — Using a peering connection removes the need to configure route table entries for private traffic.

Rationale: Peering requires adding appropriate route table entries so subnets know to send traffic via the peering connection.

D ✓ — You must create a separate peering connection for each VPC pair so traffic can flow directly between them.

Rationale: VPC peering is point-to-point; each pair requires its own peering connection and route entries for direct private IP traffic.

E ✗ — Once VPC-A peers with VPC-B, VPC-A can use that link to reach on-premises networks connected to VPC-B by VPN.

Rationale: Peering does not provide edge-to-edge forwarding; it cannot carry traffic to on-premises networks attached to the other VPC.

F ✗ — VPC peering allows connections between VPCs that share the same IP ranges without changes.

Rationale: Peering cannot be established if the VPC CIDR ranges overlap; non-overlapping address ranges are required.

Explanation

VPC peering creates a private, direct IP path between two VPCs so resources can exchange traffic without traversing the public internet. A core limitation is that a peering connection links only the two VPC endpoints involved. Therefore, each pair of VPCs that must communicate needs its own peering link and explicit routing entries; a peering link will not forward traffic from one VPC to a third VPC or to external networks connected to the peer.

Why the correct choices are right. The statement about creating a separate peering connection for each pair is correct because peering is point-to-point and not a multi-hop fabric. Establishing a peering connection alone is not sufficient; you must also add route table entries so subnets send matching destination IPs over that peering link. The statement that a peered VPC will not forward packets to another VPC is correct because peering does not permit transit routing — it does not act as a router between other VPCs or external links.

Why the incorrect choices are wrong. Routing through a peered VPC to reach on-premises via that VPC's VPN is not supported because edge-to-edge forwarding is disallowed. Claims that peering auto-propagates routes or removes the need to update route tables are incorrect: routes must be configured explicitly for each peering pair. Allowing peering between overlapping IP ranges is incorrect since overlapping CIDRs prevent establishing a peering connection.

Key Concepts

- **VPC peering point-to-point:** A peering link connects exactly two VPCs and only carries traffic between them.
- **Non-transitive routing:** Peering does not support multi-hop forwarding; packets are not routed through one peer to reach another.
- **Route table control:** Route entries must be created in each VPC to direct traffic to the peering connection.

Common Pitfalls

- Assuming one peering connection can act as a hub for multiple VPCs leads to missing connectivity; each pair needs its own link.
- Forgetting to add route table entries after creating a peering connection results in no traffic flow despite an active connection.

Glossary

- **Peering link:** A direct, private network connection between two VPCs used for exchanging private IP traffic.
- **Transit routing:** Forwarding traffic through an intermediate network or device to reach a third network; not supported by VPC peering.

15.7 — Tags for cost allocation/chargeback (D4.T4.2)

Domain 4 • Task 4.2 ↑ Back to Question 15.7

A finance team expects to see project-level costs but reports all resources appear under a single account total. Which likely cause explains why costs cannot be split by project in the cost reports?

Options & Rationales

A ✗ — The account is missing server-side encryption, which prevents the billing tool from reading resource usage.

Rationale: Encryption affects data protection, not the presence of metadata used to group costs; it does not stop cost allocation by tags.

B ✗ — Network traffic is routed through an internet gateway, causing costs to be aggregated at the VPC level rather than per resource.

Rationale: Routing choices change network flow but do not remove resource metadata needed for cost grouping; tagging is required for allocation.

C ✗ — All resources are in the same Availability Zone, so cost reports cannot separate them by project.

Rationale: Availability Zone placement does not prevent cost breakdown by project; tags provide the necessary grouping information.

D ✓ — Many resources were not labeled with the project identifier tag, so the billing tool cannot group costs by project.

Rationale: If resources lack the project tag, the billing system has no metadata to assign costs to projects, preventing grouped reporting.

Explanation

Cost reports rely on metadata assigned to individual resources to attribute charges to business owners or projects. When resources lack that identifying label, the reporting tool cannot map consumption to a project, so costs remain in undifferentiated account totals. The correct resolution is to ensure resources are consistently labeled with the project identifier so the billing system can group and report costs accordingly.

Why the correct choice is right and others are wrong. The correct choice identifies missing resource labels as the root cause; without those labels the billing tool cannot assign costs to projects. The other options describe infrastructure or security attributes that do not provide the grouping metadata needed for cost allocation. Encryption and networking settings affect security and traffic, not whether resources carry the identifiers required for chargeback or showback reporting.

Key Concepts

- **Resource tag:** A metadata label attached to a cloud resource that identifies ownership, project, or purpose; used for organizing and reporting.

- **Cost grouping:** The process of aggregating usage and charges by metadata so stakeholders can see spending per project or owner.

Common Pitfalls

- Assuming infrastructure placement or encryption controls cost visibility; these do not replace missing metadata.
- Expecting automatic grouping without verifying that the required tags are present and applied consistently across resources.

Glossary

- **Tag key/value:** The name and value pair used as a label on resources to categorize and filter them for reporting purposes.

15.8 — re:Post for community problem solving (D4.T4.3)

Domain 4 • Task 4.3 ↑ Back to Question 15.8

Which statements describe how AWS re:Post helps resolve technical questions?

Options & Rationales

A ✗ — It offers a guaranteed 15-minute response-time SLA for all questions.

Rationale: re:Post is community-driven and does not provide response SLAs; support SLAs are tied to AWS Support plans.

B ✓ — AWS participation and moderation help maintain accuracy and quality.

Rationale: AWS oversight curates content and reduces misinformation, improving trust in accepted solutions.

C ✗ — It replaces the need for AWS Support plans for production incidents.

Rationale: Community Q&A does not substitute for official AWS Support; critical incidents still require support plans.

D ✗ — It is a private forum limited to users within a single AWS account.

Rationale: re:Post is a public community knowledge platform, not an account-restricted forum.

E ✓ — Community voting and peer review elevate reliable answers faster.

Rationale: Peer validation highlights the most useful responses, helping users find correct solutions quickly.

Explanation

AWS re:Post is a community-driven question-and-answer platform designed to speed problem resolution by surfacing trustworthy, reusable solutions. The key mechanism is crowdsourced validation—participants upvote and review responses so the most accurate and helpful answers rise to the top. In addition, AWS involvement and moderation help keep content accurate and aligned with current guidance, which reduces misinformation and shortens the time it takes to find a correct fix.

The statement about community voting and peer review is correct because it explains how re:Post highlights reliable content rapidly. The statement about AWS moderation is also correct because AWS curation increases quality and trust in the answers users rely on.

Claims of a guaranteed 15-minute response SLA are incorrect; re:Post is not an SLA-backed support channel. Saying it is private to a single account is wrong; it is a public knowledge community. Finally, asserting that it replaces AWS Support plans is incorrect; community Q&A does not substitute for official support during production incidents.

Key Concepts

- **Community validation**: Upvotes and peer review promote accurate answers, reducing time to find solutions.
- **AWS moderation**: AWS oversight improves content quality and trustworthiness.

Common Pitfalls

- Confusing community Q&A with formal support SLAs or assuming it replaces paid AWS Support for production issues.

15.9 — Transit Gateway connects VPCs & on-prem (D3.T3.5)

Domain 3 • Task 3.5 ↑ Back to Question 15.9

Which capability best describes AWS Transit Gateway's primary role in connecting multiple VPCs and an on-premises datacenter?

Options & Rationales

A ✗ — Perform inline traffic inspection and act as a stateful firewall for all traffic between VPCs.

Rationale: Transit Gateway is not a packet inspection or stateful firewall service; specialized network or security appliances perform inspection and firewalling.

B ✗ — Automatically convert VPC peering into a managed VPN with per-flow encryption between every VPC pair.

Rationale: Transit Gateway provides centralized routing and attachments, but it does not convert peering into separate VPNs or automatically create per-flow encrypted tunnels between every VPC pair.

C ✗ — Replace Internet Gateways by providing public internet access for instances in all connected VPCs.

Rationale: Transit Gateway manages private routing between networks; Internet Gateways (or NAT/egress solutions) are still required for public internet access.

D ✓ — Provide a single, central routing hub that aggregates VPCs and on-premises networks so they can exchange routes through one attachment point.

Rationale: Transit Gateway centralizes connectivity by linking VPC and on-prem attachments into a single routable fabric and controlling route propagation with route tables.

Explanation

AWS Transit Gateway is a managed networking service that acts as a central routing hub for cloud and on-premises networks. It lets you attach multiple VPCs and customer networks (for example, VPNs or Direct Connect) to a single, shared gateway so routes can be exchanged and controlled from a small number of route tables. This reduces the complexity of managing many individual point-to-point connections and simplifies routing decisions.

Why the correct choice fits. The described capability explains that Transit Gateway aggregates attachments and provides centralized route control. That captures the core purpose: simplify and scale connectivity by offering one logical attachment point per network and using route propagation and route tables to determine how traffic flows between attachments.

Why the other descriptions are incorrect. The idea that Transit Gateway performs inline traffic inspection or functions as a stateful firewall confuses routing with security appliances; separate services or appliances provide packet inspection and stateful filtering. The suggestion that it automatically converts VPC peering into managed per-flow VPNs is inaccurate because Transit Gateway provides routable attachments, not per-flow encrypted tunnels between every VPC pair. Finally, Transit Gateway does not replace Internet Gateways for providing public internet egress; public access still requires the appropriate internet/NAT components in each VPC.

Key Concepts

- **Transit Gateway aggregation:** A single managed gateway can attach many VPCs and on-prem connections to reduce point-to-point complexity.
- **Route tables and propagation:** Route tables on the gateway control which attachments can reach each other and how routes are shared.
- **Attachment types:** Attachments are logical connections (VPC, VPN, Direct Connect gateway) used to integrate networks.

Common Pitfalls

- Confusing routing services with security functions leads to expecting inspection or firewalling from Transit Gateway. Also, assuming it provides internet egress or creates per-flow VPNs mixes separate networking responsibilities.

Glossary

- **Attachment:** A logical connection between a network (VPC or on-prem) and the Transit Gateway.
- **Route table (Transit Gateway):** A central routing construct that controls how attached networks exchange traffic.

15.10 — Outposts for on-prem hybrid consistency (D3.T3.3)

Domain 3 • Task 3.3 ↑ Back to Question 15.10

Which scenario best describes when to choose AWS Outposts for a workload?

Options & Rationales

A ✗ — A public-facing website with unpredictable global traffic that should be served from many edge locations worldwide.

Rationale: Global edge delivery is better solved with a CDN rather than bringing AWS infrastructure into a private data center.

B ✗ — A temporary batch analysis job that can tolerate interruptions and runs only when spare capacity is available.

Rationale: Interruption-tolerant batch work is a fit for spot capacity in the cloud, not fixed on-premises rack deployments.

C ✓ — An application needs cloud APIs and managed services running inside a company data center because users require extremely low latency to local devices.

Rationale: Outposts extends AWS-managed compute and services into a customer's site to provide low-latency access while keeping the same AWS APIs and management model.

D ✗ — A lightweight static marketing site with minimal backend needs and tight cost constraints where hosting simplicity is key.

Rationale: Simple, low-cost hosting is typically handled by managed cloud storage or simple hosting services rather than deploying on-premises infrastructure.

Explanation

AWS Outposts is a fully managed option that places AWS compute and related services physically at a customer site to support workloads needing very close proximity to local systems. The key idea is bringing the cloud operational model, APIs, and managed services into an on-premises environment so applications can interact with local devices or systems with minimal network delay.

Why the correct choice fits. The chosen scenario requires extremely low latency to local devices and benefits from using the same AWS APIs and managed services inside the company's data center. This matches the purpose of a managed on-site solution that preserves cloud management while reducing round-trip latency to distant Regions.

Why the other choices are not appropriate. The option describing global edge delivery targets distributing content worldwide and is best served by a content delivery network and edge locations, not placing AWS racks on-premises. The interruption-tolerant batch job is suited for cost-saving cloud options like spot instances that accept interruptions; it does not require dedicated on-site

infrastructure. The simple static site emphasizes low cost and easy hosting; that is better hosted with managed storage or simple hosting services rather than deploying physical racks locally.

Key Concepts

- **Purpose of on-premises managed AWS hardware:** Provides AWS-managed compute and services inside a customer facility so workloads needing close proximity to local systems see lower latency while keeping familiar cloud APIs and management.
- **Low-latency local processing:** Workloads that must exchange frequent, time-sensitive data with local devices benefit from being located physically near those devices to reduce network round-trip time.
- **Appropriate alternatives:** CDN/edge services for global content distribution; spot instances for interruption-tolerant batch; managed cloud hosting for simple static sites.

Common Pitfalls

- Assuming any workload benefits from on-site AWS hardware; only those that require both AWS APIs and very low local latency typically justify Outposts. Confusing edge CDN use-cases or interruption-tolerant, low-cost batch jobs with Outposts leads to wrong choices.

Glossary

- **CDN:** Content Delivery Network, used to cache and deliver content from edge locations close to users.
- **Spot instances:** Cloud compute capacity offered at a discount with the possibility of interruption.
- **Managed on-site AWS:** A service model that brings AWS-managed infrastructure into a customer's physical location for low-latency and hybrid scenarios.

15.11 — Federation with external IdP (D2.T2.3)

Domain 2 • Task 2.3 ↑ Back to Question 15.11

Which of the following best describes the primary purpose of AWS Security Token Service (STS)?

Options & Rationales

A ✗ — Stores and manages long-lived access keys for IAM users to authenticate API requests indefinitely.

Rationale: STS does not create or manage permanent access keys; long-lived IAM access keys are separate and not the responsibility of STS.

B ✗ — Acts as a directory service for user accounts and stores passwords for AWS-managed identities.

Rationale: STS is not a user directory or password store; directory services are

provided by other services like AWS Directory Service.

C ✗ — Encrypts data at rest and handles customer-managed cryptographic keys for S3 and EBS volumes.

Rationale: Key management and encryption are functions of AWS KMS, not STS; STS focuses on temporary credentials and role assumption.

D ✓ — Provides temporary AWS credentials that let an authenticated principal assume a role and access AWS resources.

Rationale: AWS STS issues temporary credentials tied to an assumed role so authenticated identities can access AWS resources for a limited time.

Explanation

AWS Security Token Service (STS) is a service that provides temporary credentials for AWS access. When an identity—such as a federated user or an IAM principal—needs to act with a specific set of permissions, STS issues time-limited credentials that represent an assumed role. These credentials include temporary access key material and a session token that allow API calls for the duration of the session, reducing the risk associated with long-lived static credentials.

Why the correct description fits. The correct choice states that STS provides temporary credentials tied to role assumption and limited-duration access. This captures STS's primary purpose: enabling least-privilege, short-duration access patterns used for federation, cross-account access, and service-to-service delegation without creating permanent keys.

Why the other descriptions are incorrect. The statement about storing and managing long-lived access keys confuses STS with IAM user credentials; STS deliberately issues transient credentials rather than permanent ones. Describing STS as a directory or password store is inaccurate because directory and identity stores are handled by separate services. Claiming STS manages encryption keys is wrong; cryptographic key lifecycle and encryption at rest are responsibilities of AWS KMS and service-specific encryption features.

Key Concepts

- **Temporary credentials:** Short-duration access tokens that reduce exposure from compromised keys by expiring automatically.
- **Role assumption:** A mechanism where an identity obtains permissions by taking on an IAM role, represented by STS credentials for a session.

Common Pitfalls

- Assuming STS creates permanent user keys or stores passwords leads to misuse; STS is about issuing session-based credentials, not managing long-lived secrets or encryption keys.

Glossary

- **Role assumption:** The act of acquiring a role's permissions for a limited session.
- **Session token:** The additional piece of data paired with temporary access keys that authenticates an STS-issued session.

15.12 — Regions vs Availability Zones vs Local Zones (D3.T3.2)

Domain 3 • Task 3.2 ↑ Back to Question 15.12

Which statement best defines an AWS Availability Zone (AZ)?

Options & Rationales

A ✗ — A separate geographic area that contains multiple AZs and serves as a data residency boundary.

Rationale: This describes an AWS Region, not an AZ. Regions are isolated geographic areas that include multiple AZs.

B ✗ — An extension of a Region that places select services near metro areas for single-digit millisecond latency and is opt-in per account.

Rationale: This describes an AWS Local Zone, which extends a parent Region for ultra-low latency and offers a limited service set.

C ✓ — One or more discrete data centers within a single Region, with independent power, cooling, and networking, connected by high-throughput, low-latency links.

Rationale: This is the definition of an AZ. AZs are physically separate facilities inside a Region and act as fault-isolation boundaries.

D ✗ — A fault-isolation boundary that spans multiple Regions to provide disaster recovery across geographic areas.

Rationale: AZs do not span Regions. Multi-Region architectures support disaster recovery; AZs provide high availability within a single Region.

Explanation

AWS infrastructure is organized hierarchically. A Region is a separate geographic area that contains multiple Availability Zones (AZs) and provides data residency boundaries. An AZ is one or more discrete data centers inside a Region. These facilities are engineered for physical separation with independent power, cooling, and networking, and are interconnected by high-throughput, low-latency links. AZs are fault-isolation boundaries; designing across at least two AZs improves availability and resilience within a single Region.

Common high-availability patterns use multiple AZs: deploying Amazon EC2 instances in separate subnets (one per AZ) behind an Elastic Load Balancer, using Amazon RDS Multi-AZ for automatic failover, and relying on regional

services like Amazon S3 and Amazon EFS that replicate across multiple AZs by design. Under the shared responsibility model, AWS operates the infrastructure while customers select multi-AZ architectures to meet reliability goals.

Local Zones extend a Region by placing selected services closer to major metro areas to achieve single-digit millisecond latency. They are anchored to a parent Region, offer a limited service set (such as EC2 and EBS), and are opt-in per account. A Local Zone is not a substitute for multi-AZ designs in the parent Region.

The statement describing one or more discrete, physically separate data centers within a Region connected by high-throughput, low-latency links is the definition of an AZ. The statement about a separate geographic area containing multiple AZs defines a Region. The statement about metro-proximate, opt-in infrastructure with a limited service set defines a Local Zone. The statement claiming a fault-isolation boundary that spans multiple Regions is incorrect—AZs exist only within a single Region; multi-Region designs are a disaster recovery strategy.

Key Concepts

- **Region:** Separate geographic area with multiple AZs and data residency boundaries; isolation supports disaster recovery across Regions.
- **Availability Zone:** Physically separate data centers within a Region; fault-isolation boundaries connected by low-latency links for high availability.
- **Local Zone:** Region extension near metro areas for ultra-low latency; limited services; not a replacement for multi-AZ resilience.

Common Pitfalls

- Treating a Local Zone as an AZ for high availability or assuming AZs span Regions. Multi-Region is for disaster recovery, not a substitute for multi-AZ within a Region.

15.13 — Pricing Calculator for monthly estimates (D4.T4.2)

Domain 4 • Task 4.2 ↑ Back to Question 15.13

A startup plans a small web application in us-east-1 with two Amazon EC2 instances running 24×7 behind an Application Load Balancer, a NAT Gateway for outbound traffic, EBS gp3 volumes, and moderate data transfer out to the internet. Leadership wants a credible monthly cost forecast to review with stakeholders and a way to track actual spend against the forecast after launch. Which approach is the MOST appropriate?

Options & Rationales

A ✗ — Estimate only EC2 instance hours in AWS Pricing Calculator and assume Free Tier/discounts auto-apply; omit data transfer, ALB, NAT; skip budgets until after launch.

Rationale: Omitting supporting services and data transfer is a documented

pitfall, and Free Tier/discounts do not auto-apply in the estimate. Budgets help track against the model from day one.

B ✗ — Price the workload in a different Region to get lower prices, ignore Multi-AZ/backup options to simplify, and let taxes/credits reconcile differences later.

Rationale: Prices vary by Region and taxes/credits are not reflected in the calculator. Excluding Multi-AZ/backup underestimates storage and I/O that should be modeled.

C ✗ — Create AWS Budgets for the target monthly amount without an estimate; budgets will automatically calculate per-service usage and forecast monthly cost.

Rationale: Budgets track actual spend versus a target; they do not build per-service estimates. A calculator estimate should precede budgets for meaningful tracking.

D ✓ — Use AWS Pricing Calculator in us-east-1 to model EC2 (2×24×7, Graviton), EBS gp3, ALB, NAT, and data transfer; compare On-Demand vs Savings Plans; share export; then create aligned AWS Budgets.

Rationale: This models all relevant per-service line items using rate × usage in the correct Region, considers purchasing options, communicates assumptions, and uses AWS Budgets to track actual vs modeled spend.

Explanation

At planning time, the AWS Pricing Calculator is the right tool to translate usage assumptions into a per-service cost model in a specific Region. You enter capacity and usage metrics—such as EC2 instance hours, EBS gp3 GB-months and any provisioned performance, expected data transfer out, plus supporting components like an Application Load Balancer and a NAT Gateway. The calculator applies published unit prices and aggregates them into a monthly total. You can also model purchasing choices (for example, On-Demand versus Savings Plans for steady workloads) and enable Multi-AZ or backup features so their storage and I/O are included. After sharing/exporting the estimate to validate assumptions with stakeholders, create AWS Budgets that align to the modeled amount so you can track actual spend against the forecast after launch.

The underlying calculation is:

$$\text{Monthly cost} = \sum(\text{rate} \times \text{usage})$$

Variables used:

- **rate:** Published unit price per usage unit [$/unit]
- **usage:** Estimated consumption for the month [units/month]
- **Monthly cost:** Total modeled charge [$]

The solution that builds a comprehensive estimate in us-east-1 for EC2 (two instances running 24×7, preferably Graviton for cost efficiency if chosen), EBS

gp3, ALB, NAT Gateway, and data transfer—while comparing On-Demand to Savings Plans—meets the forecasting requirement and avoids common omissions. Sharing the estimate aligns stakeholders on assumptions, and AWS Budgets then track actual spend versus the modeled amount.

Other choices fail key requirements: estimating only EC2 and assuming Free Tier or discounts will auto-apply ignores known pitfalls; pricing in a different Region and omitting Multi-AZ/backup features distorts the model and taxes/credits are not reflected; relying on AWS Budgets alone does not produce an estimate because budgets track rather than calculate costs.

Key Concepts

- **AWS Pricing Calculator:** Converts service usage inputs (instance hours, GB-months, requests, data transfer) into per-service line items for a chosen Region, then sums monthly cost.
- **Rate × Usage Model:** Each cost component equals unit price multiplied by estimated consumption; aggregating these totals yields the forecast.
- **Purchasing Options:** Compare On-Demand and Savings Plans for steady workloads to understand cost trade-offs in the estimate.
- **AWS Budgets Alignment:** Use budgets to monitor actual spend versus the modeled total and stay accountable to the forecast.

Common Pitfalls

- Forgetting data transfer, load balancers, NAT Gateway, snapshots, or CloudWatch logs, or assuming Free Tier/discounts auto-apply, leads to underestimates. Pricing in the wrong Region also skews results.

15.14 — Data transfer pricing within/across Regions (D4.T4.1)

Domain 4 • Task 4.1 ↑ Back to Question 15.14

A company runs Amazon EC2 instances in private subnets across two Availability Zones in the same AWS Region. The instances frequently access Amazon S3 buckets in the same Region. All outbound traffic currently routes through a single NAT Gateway in one AZ. The company wants to minimize data transfer charges for this S3 access and avoid NAT Gateway processing fees. What should the company do?

Options & Rationales

A ✗ — Deploy a NAT Gateway in each Availability Zone and route each private subnet to its local NAT Gateway.

Rationale: This reduces cross-AZ data transfer, but S3 traffic still flows through NAT Gateways, incurring NAT data processing fees.

B ✗ — Use Amazon CloudFront in front of the S3 buckets so EC2 requests are served from edge locations, reducing Regional egress.

Rationale: CloudFront helps internet-facing egress. EC2-to-S3 access from private subnets would still traverse internet paths via NAT and incur NAT

processing costs.

C ✗ — Assign Elastic IP addresses to the EC2 instances so they can access S3 directly without NAT.

Rationale: Using public or Elastic IPs can trigger data transfer charges even within the same AZ and does not minimize cost for intra-Region S3 access.

D ✓ — Create a VPC gateway endpoint for Amazon S3 and update private subnet route tables to use it.

Rationale: An S3 VPC gateway endpoint keeps EC2↔S3 traffic on the AWS network and bypasses the NAT Gateway, avoiding NAT processing and cross-AZ NAT path costs.

Explanation

Data transfer pricing depends on the traffic path. Inbound from the internet is generally free, while outbound to the internet is billed per GB. Within a Region, traffic using private IPs in the same Availability Zone (AZ) is typically free, but traffic that crosses AZs is billed per GB in each direction. NAT Gateways add per-GB data processing charges and can cause extra inter-AZ costs if subnets in other AZs route to a centralized NAT. For Amazon S3 specifically, using a VPC gateway endpoint keeps traffic on the AWS network and avoids NAT processing fees. In many cases, S3-to-EC2 transfers within the same Region are often free of data transfer charges, but unnecessary NAT traversal still adds cost.

Creating a VPC gateway endpoint for Amazon S3 and updating private subnet routes sends EC2↔S3 traffic over the AWS network without using the NAT Gateway, eliminating NAT processing charges and inter-AZ NAT paths. Other options either continue to use NAT (and thus keep NAT processing costs) or move traffic onto public/Elastic IP paths that can trigger data transfer charges.

Why the other choices are incorrect:

- Deploying a NAT Gateway in each AZ reduces inter-AZ data transfer to the NAT, but all S3 traffic still incurs NAT data processing fees, failing the goal to avoid NAT costs.
- Putting CloudFront in front of S3 helps internet-facing workloads by reducing Regional egress, but EC2 instances in private subnets would still reach CloudFront via NAT (internet path), preserving NAT processing costs.
- Assigning Elastic IPs moves traffic to public addressing. Using public or Elastic IPs can trigger data transfer charges even within the same AZ and does not minimize intra-Region S3 access costs.

Key Concepts

- **VPC Gateway Endpoint (S3/DynamoDB):** Keeps traffic on the AWS network and avoids NAT Gateway processing fees for those services.
- **Inter-AZ Data Transfer:** Billed per GB in each direction; centralized NAT can add these costs for cross-AZ flows.

- **Public/Elastic IP Use:** Even within the same AZ, public paths can trigger data transfer charges.
- **CloudFront Scope:** Optimizes internet-facing egress and latency, not private EC2-to-S3 paths inside a VPC.

Common Pitfalls

- Centralizing a single NAT Gateway to save hourly cost but increasing total spend due to inter-AZ data transfer and NAT processing fees.
- Assuming CloudFront reduces charges for internal EC2-to-S3 access from private subnets.

15.15 — Shared controls between AWS and customer (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 15.15

A regulated insurer stores sensitive claim documents in Amazon S3 and processes them with AWS Lambda. An audit requires two specific controls: (1) all objects must be encrypted at rest with keys the company manages, and (2) only encrypted connections may access the bucket. Under the shared responsibility model, which actions are the company responsible for implementing to meet these requirements?

Options & Rationales

A ✗ — Rely on Amazon S3 defaults to enforce TLS automatically without a bucket policy or code changes.

Rationale: AWS offers TLS, but customers must enforce TLS-only access; relying on defaults does not prove or guarantee compliance.

B ✗ — Depend on AWS to patch both the AWS Lambda runtime and your function's third-party libraries.

Rationale: AWS patches the Lambda runtime, but customers must patch application dependencies. Patching also does not address the encryption controls requested.

C ✗ — Enable S3 Block Public Access at the account level to prevent public reads.

Rationale: A strong configuration control for exposure risk, but it does not implement encryption at rest or enforce TLS for this requirement.

D ✗ — Turn on AWS CloudTrail and Amazon GuardDuty across accounts and centralize logs to Amazon S3.

Rationale: Useful logging and threat detection controls, but they do not satisfy encryption at rest or TLS enforcement requirements.

E ✓ — Configure Amazon S3 server-side encryption with AWS KMS (SSE-KMS) using a customer managed key and a restrictive key policy.

Rationale: Customers choose SSE-KMS, create and manage customer managed keys, and write key policies to control access—meeting the "keys they manage"

requirement.

F ✓ — Add an Amazon S3 bucket policy that denies requests not using TLS (require HTTPS/TLS-only access).

Rationale: AWS provides TLS support, but customers must enforce TLS-only access via bucket policy to satisfy encrypted-in-transit requirements.

Explanation

The shared responsibility model divides work by scope: AWS secures the global infrastructure and provides managed services and security features, while customers configure those features to meet workload requirements and demonstrate compliance. Encryption and transport security are classic shared controls—AWS offers capabilities (for example, AWS Key Management Service and TLS), but customers must make secure choices and enforce policies for their data paths.

To satisfy encryption at rest with keys under customer control, the customer should configure Amazon S3 to use server-side encryption with AWS KMS (SSE-KMS) and choose a customer managed key. Customers create and manage these keys, write restrictive key policies and grants, and thereby control who can use the key. This directly fulfills the audit requirement that keys are managed by the company.

For encryption in transit, AWS supports TLS endpoints for S3, but proving and enforcing that every request is encrypted is the customer's responsibility. An S3 bucket policy that denies any request not using TLS (that is, require TLS-only access) is the standard way to enforce and evidence this control.

Other actions are valuable but do not meet the stated controls. Enabling S3 Block Public Access reduces accidental exposure but neither encrypts data at rest nor enforces TLS. Turning on AWS CloudTrail and Amazon GuardDuty strengthens logging, monitoring, and threat detection, yet does not implement encryption or transport enforcement. Assuming S3 enforces TLS by default without a bucket policy leaves a compliance gap because the customer has not enforced TLS-only access. Relying on AWS to patch both the Lambda runtime and application libraries misunderstands patch responsibilities—AWS patches the runtime and platform, while customers must patch their application dependencies—plus patching does not address encryption.

Key Concepts

- **Shared controls:** AWS provides security features; customers must configure and enforce them for their workloads (e.g., encryption, logging, identity).
- **SSE-KMS with customer managed keys:** Customers select SSE-KMS, create and manage keys, and write key policies to control use, meeting "keys we control."
- **TLS enforcement:** AWS supports TLS, but customers enforce TLS-only access to S3 via a bucket policy that denies non-TLS requests.

- **Control families distinction:** Logging/monitoring (CloudTrail, GuardDuty) are separate from encryption and do not satisfy at-rest or in-transit encryption requirements.

Common Pitfalls

- Assuming AWS automatically enforces TLS for all S3 access without a bucket policy. Customers must explicitly require TLS-only access.
- Confusing platform patching with application patching, or believing logging controls alone fulfill encryption requirements.

15.16 — Security groups vs NACLs (D3.T3.5)

Domain 3 • Task 3.5 ↑ Back to Question 15.16

Which VPC traffic-control feature automatically permits response packets for a connection that was allowed inbound, because it tracks connection state?

Options & Rationales

A ✗ — Network ACL (subnet-level, stateless)

Rationale: Network ACLs are stateless and require explicit rules for both directions; they do not automatically allow return traffic.

B ✗ — VPC route table (routing decision)

Rationale: Route tables direct packets between subnets and gateways but do not inspect or allow/deny traffic based on connection state.

C ✗ — Internet gateway (connectivity to the internet)

Rationale: An internet gateway provides internet access for a VPC but does not perform stateful firewalling or track connection state.

D ✓ — Security group (instance-level, stateful)

Rationale: Security groups are stateful, so return traffic for an allowed inbound connection is automatically permitted without separate inbound/outbound rules.

Explanation

In AWS VPCs, traffic controls differ by where they apply and whether they track connection state. A stateful control remembers the details of an established connection and automatically permits matching return packets; a stateless control treats each packet independently and requires explicit rules for both directions.

The security group is an instance-level, stateful firewall. When an inbound packet is allowed by a security group rule, the security group records that connection and automatically allows the corresponding outbound response packets without needing a separate outbound rule for the same flow. This behavior simplifies rule configuration for typical client-server interactions.

A network ACL is a subnet-level, stateless filter. It evaluates each packet on its own and requires separate rules to allow the initial and return packets. Route

tables and internet gateways serve routing and connectivity roles rather than acting as firewalls that inspect connection state.

Key Concepts

- **Stateful vs. stateless:** Stateful systems track connection state and permit return traffic automatically; stateless systems do not track state and need explicit bidirectional rules.
- **Security group purpose:** Instance-level virtual firewall that simplifies bidirectional traffic handling by tracking connections.
- **Network ACL purpose:** Subnet-level packet filter that applies rules per packet without automatic return-traffic handling.

Common Pitfalls

- Confusing routing components (route tables, internet gateway) with firewall behavior leads to thinking they control return traffic; they do not inspect or permit traffic based on connection state.
- Assuming a stateless filter will automatically allow responses can cause failed connections if return rules are missing.

Glossary

- **Security group**: Instance-level, stateful traffic control that tracks connections.
- **Network ACL:** Subnet-level, stateless packet filter requiring explicit rules for both directions.

15.17 — On-prem cost elements: facility/power/refresh (D1.T1.4)

Domain 1 • Task 1.4

↑ Back to Question 15.17

Which cost is typically a direct, recurring expense for operating an on-premises data center but is usually covered by the cloud provider when you move workloads to AWS?

Options & Rationales

A ✗ — Subscription fees for managed database services

Rationale: Managed database subscriptions are a cloud service cost rather than an on-premises-only expense.

B ✓ — Electricity for running servers and cooling systems

Rationale: Power consumption and cooling are ongoing facility expenses that on-premises owners pay; cloud providers include these in service pricing.

C ✗ — Operator time to write application code

Rationale: Development labor exists for both on-premises and cloud deployments and is not unique to on-premises facilities.

D ✗ — Network bandwidth charges for internet egress

Rationale: Cloud providers commonly charge for data transfer; this is not an on-premises-only recurring facility expense.

Explanation

Power for servers, racks, and the supporting cooling infrastructure is a recurring operational expense that organizations with their own data centers must pay directly. When workloads run in a cloud provider's facilities, the provider absorbs the facility-level utilities and recovers those costs through its service pricing, so customers no longer pay separate electricity or cooling bills for those hosted resources.

Why the correct content fits. Electricity and cooling are tied to the physical operation of on-site hardware and facilities, making them a distinctive on-premises cost category. The cloud model shifts responsibility for utilities to the provider, converting those fixed operational expenses into part of the service fee.

Why the other choices are incorrect. Subscription fees for managed databases represent cloud service costs rather than an on-premises-only charge; development labor to write code is a shared organizational expense regardless of deployment location; network egress fees are commonly billed by cloud providers and therefore are not unique on-premises facility expenses.

Key Concepts

- **Facility utility costs:** On-premises data centers incur direct bills for electricity and cooling linked to running hardware and environmental controls.
- **Cloud provider responsibility:** Moving to cloud relocates facility-level utilities to the provider, who bundles them into service pricing rather than billing customers separately for power.

Common Pitfalls

- Confusing cloud service charges with on-premises facility costs; some expenses (like managed service subscriptions or egress charges) are billed by the cloud but are not unique to owning a data center.
- Assuming all operating expenses disappear in the cloud; operational functions remain, but facility utilities are typically handled by the provider.

Glossary

- **Egress charges:** Fees for sending data out of a cloud provider's network to the public internet or other regions.
- **Managed service subscription:** A paid cloud offering where the provider operates and maintains a service on behalf of the customer.

15.18 — DR with cross-Region replication (D3.T3.2)

Domain 3 • Task 3.2 ↑ Back to Question 15.18

A company needs a disaster recovery setup in a second AWS Region that runs a smaller but ready copy of their production system to take over quickly during

an outage. Which recovery topology best matches this requirement?

Options & Rationales

A ✗ — A minimal core environment pre-created and scaled up only after failover

Rationale: This describes keeping only essential components ready and launching full capacity after an incident, which prioritizes lower running cost but slower recovery.

B ✗ — Full duplicate infrastructure actively serving live traffic across Regions

Rationale: This is a fully distributed active setup that maximizes availability and performance but costs more because all Regions serve traffic.

C ✗ — Periodic backups copied to another Region and restored when needed

Rationale: This relies on restoring data from remote backups; it is cost-efficient but typically has longer recovery time compared with a running replica.

D ✓ — A reduced-capacity duplicate continuously running in the alternate Region for fast failover

Rationale: A running, smaller-scale replica in another Region provides quicker recovery than restoring from backups while saving cost versus full active-active.

Explanation

Designing cross-Region recovery involves a range of topologies that trade ongoing cost, recovery speed, and operational complexity. One option runs a smaller, live copy of the system in a secondary Region at reduced capacity so it can take over quickly when the primary fails. This approach keeps components warm and running, reducing recovery time compared with restoring from backups while costing less than running full capacity in two Regions.

Other strategies include maintaining only core services that are scaled up after an incident; that lowers ongoing expense but increases the time needed to reach full capacity. A fully active multi-Region deployment serves traffic from every Region simultaneously, offering the fastest failover and highest availability but also the highest continuous cost and complexity. Using periodic cross-Region backups stores recoverable data offsite and is the most cost-conscious option but results in the longest restore windows.

Why the correct choice fits. Keeping a reduced-capacity duplicate continuously operating in a secondary Region balances faster recovery and lower ongoing expense. It meets the requirement for a smaller but ready copy that can assume full duties quickly, unlike approaches that either wait to scale after an event or keep both Regions fully active.

Key Concepts

- **Warm, running duplicate:** A smaller-capacity replica kept active in another Region to shorten recovery time while reducing ongoing cost compared with fully active deployments.

- **Pilot/minimal pre-provisioned core:** Essential systems exist but the rest are started only after failover; good for cost savings but slower recovery.
- **Active-active multi-Region:** All sites serve traffic continuously to maximize resilience and performance at higher cost and complexity.

Common Pitfalls

- Confusing a running scaled-down duplicate with a minimal standby: the former is actively processing or ready to process, while the latter may require significant provisioning after failover.
- Assuming backups alone achieve quick recovery: backups protect data but typically increase recovery time because systems must be rebuilt and data restored.

Glossary

- **Failover:** Switching operations from a failed system to a standby system.
- **Recovery time objective (RTO):** Target time to restore service after an incident.
- **Recovery point objective (RPO):** Maximum acceptable amount of data loss measured in time.

15.19 — Multi-AZ availability & fault tolerance benefits (D1.T1.1)

Domain 1 • Task 1.1 ↑ Back to Question 15.19

An online retailer runs a web app on a few Amazon EC2 instances in one Availability Zone and an Amazon RDS MySQL DB instance in a single AZ. A recent AZ disruption caused hours of downtime. The company wants to achieve at least 99.9% availability within one AWS Region, keep the customer experience consistent during an AZ failure, avoid changing the database engine, and minimize operational effort. Which solution is the BEST fit?

Options & Rationales

A ✗ — Use an Application Load Balancer with an EC2 Auto Scaling group across three AZs; keep the database single-AZ to reduce cost; store static assets in Amazon S3.

Rationale: The database remains a single point of failure. An AZ outage impacting the DB would still take the application down, missing the availability goal.

B ✗ — Keep EC2 instances in a single AZ for simplicity; enable Amazon RDS Multi-AZ; serve static assets from Amazon S3.

Rationale: The compute tier is still a single point of failure. If that AZ fails, there are no healthy application instances to serve traffic despite a resilient database.

C ✓ — Place an Application Load Balancer with cross-zone load balancing in

front of an EC2 Auto Scaling group spanning three AZs; enable Amazon RDS Multi-AZ; store static assets in Amazon S3.

Rationale: Removes single points of failure across tiers. RDS Multi-AZ provides synchronous replication and automated failover; ALB+Auto Scaling keep capacity across AZs; S3 is Regional.

D ✗ — Replace the relational database with Amazon DynamoDB (regional, multi-AZ); keep EC2 in one AZ; use Amazon S3 for static content.

Rationale: Violates the requirement to avoid changing the database engine and leaves compute in one AZ. DynamoDB is not a drop-in replacement for a relational workload.

Explanation

Availability Zones (AZs) are separate data centers within a Region, connected by low-latency links. Designing across AZs provides high availability and fault tolerance so the system can continue operating even if an AZ fails. Many AWS managed services are Multi-AZ by design: Amazon S3 and Amazon DynamoDB are Regional and store data redundantly across multiple AZs. For relational databases, Amazon RDS Multi-AZ uses synchronous replication and automated failover. In the compute tier, Elastic Load Balancing with cross-zone load balancing and Amazon EC2 Auto Scaling spread instances across AZs, removing single points of failure. These patterns reduce mean time to recover (MTTR), which improves overall availability.

Availability is improved by reducing MTTR relative to mean time between failures (MTBF):

$$A = \frac{\text{MTBF}}{\text{MTBF} + \text{MTTR}}$$

Variables used:

- **A:** Availability [unitless fraction]
- **MTBF:** Mean Time Between Failures [hours]
- **MTTR:** Mean Time To Recover [hours]
- **T:** Total time observed [hours]

Expected downtime over period T is Downtime $= (1 - A) \times T$. For example, at $A = 0.9999$ over $T = 8,760$ hours, downtime is about 0.876 hours (≈52 minutes). Targeting 99.9% to 99.99% availability requires eliminating single-AZ dependencies and minimizing manual failover.

The solution that places an Application Load Balancer with cross-zone load balancing in front of an EC2 Auto Scaling group spanning three AZs, enables Amazon RDS Multi-AZ, and stores static assets in Amazon S3 is best. It delivers Multi-AZ resilience across compute and database tiers with automated failover and keeps static content available from a Regional service—meeting the 99.9% objective within one Region while minimizing operational effort.

Other proposals fall short. Keeping the database single-AZ leaves a clear single point of failure; an AZ outage affecting the DB will bring down the app. Keeping compute in a single AZ fails the requirement because there would be no healthy instances to serve traffic if that AZ fails, even if the database is Multi-AZ. Moving to Amazon DynamoDB changes the database engine (disallowed) and still leaves compute single-AZ.

Key Concepts

- **Multi-AZ architecture:** Spreading components across AZs delivers high availability and fault tolerance within a Region.
- **Cross-zone load balancing with Auto Scaling:** Distributes traffic and capacity across AZs, replacing unhealthy instances automatically.
- **RDS Multi-AZ:** Synchronous replication and automated failover reduce MTTR and remove single-AZ database dependencies.
- **Availability math:** $A = \frac{\text{MTBF}}{\text{MTBF+MTTR}}$ and Downtime $= (1 - A) \times T$ quantify the impact of reducing MTTR.

Common Pitfalls

- Assuming hardening only one tier (compute or database) is sufficient. A single-AZ dependency in any tier can negate availability gains for the whole application.

Glossary

- **Availability Zone (AZ):** An isolated data center within an AWS Region.
- **MTTR:** Time required to restore service after a failure; lowering MTTR raises availability.

15.20 — Cost considerations for AI/ML analytics (D3.T3.7)

Domain 3 • Task 3.7 ↑ Back to Question 15.20

A startup runs model training jobs and real-time inference. They want a simple cost metric to compare and optimize different configurations. Select TWO statements that best align with using compute-hours as the primary cost measure for ML workloads.

Options & Rationales

A ✗ — Always use the largest GPU instances to finish faster; higher per-hour rates always reduce total compute-hours cost.

Rationale: Larger GPUs can reduce wall-clock time but cost much more per hour; they may or may not reduce overall compute-hours cost depending on efficiency and utilization.

B ✗ — Favor managed analytics services that bill per query because they always produce lower compute-hour totals for model training.

Rationale: Per-query billing applies to analytics queries and may not reflect training compute; it does not guarantee lower compute-hours for training jobs

and may not be comparable.

C ✗ — Move older model artifacts to cheaper object storage classes to lower compute-hour charges.

Rationale: Archiving artifacts reduces storage costs but does not change compute-hours consumed during training or inference, so it doesn't address the compute-hours metric.

D ✓ — Reduce billed compute-hours by shortening training time and choosing appropriately sized instances or enabling auto-scaling for inference.

Rationale: Shorter runs and right-sized instances directly lower the compute-hours consumed; auto-scaling for inference prevents over-provisioning and reduces total billed compute time.

E ✗ — Ignore data egress because bandwidth and region transfers are unrelated to compute-hour calculations.

Rationale: While egress charges are separate from compute-hours, they can affect total workload cost; ignoring them may understate end-to-end expenses even if compute-hours are optimized.

F ✓ — Record compute-hours for each training and inference run to estimate and compare costs across instance types and durations.

Rationale: Computing total instance-hours (compute-hours) for runs yields a consistent basis to compare different instance families and job lengths, enabling cost comparisons independent of storage or per-request billing.

Explanation

For ML workloads, a practical way to compare and optimize costs is to use a single, consistent unit: compute-hours. Compute-hours measure how many hours compute resources are billed (for training or inference) and allow apples-to-apples comparisons across instance types and job durations. Focusing on compute-hours helps teams prioritize actions that directly reduce billed compute time, such as shortening jobs, choosing right-sized instances, or enabling scaling for variable load.

Why tracking compute-hours matters. it isolates the compute component of cost from storage, data transfer, or per-query pricing so decisions about instance choices and job scheduling are evidence-based. Actions that reduce run duration or prevent idle instances cut compute-hours and therefore the portion of your bill tied to compute consumption.

Discussion of options:

- The suggestion to record compute-hours for each run aligns with this approach because it provides a consistent, comparable metric across experiments and instance families. That makes it defensible as a primary optimization signal.
- Reducing run time and selecting appropriately sized instances (including use of scaling for inference) directly lowers billed compute-hours, so these

operational changes follow from using compute-hours as the metric.
- Moving artifacts to cheaper storage lowers storage costs but does not change how many compute-hours training or inference consume, so it does not address compute-hours optimization.
- Choosing services because they bill per query is an unrelated billing model; per-query pricing is not a universal proxy for compute-hours, especially for long training jobs.
- Ignoring data egress overlooks other bill components; compute-hours optimization is helpful but not a complete cost picture.
- Assuming larger GPUs always reduce total compute-hours cost is unsafe: higher hourly rates can offset shorter runtimes, so total billed compute-hours or cost must be evaluated case-by-case.

Key Concepts

- **Compute-hours:** A unit measuring billed hours of compute resources used for training or inference; useful for comparing instance usage and run durations.
- **Right-sizing:** Choosing an instance type and size that matches workload needs to avoid excess billed hours from idle or overpowered machines.
- **Auto-scaling for inference:** Adjusting capacity to match demand reduces idle time and total billed compute-hours.

Common Pitfalls

- Confusing compute-hours with total cost: storage, data transfer, and managed-service billing also contribute to bills.
- Assuming faster hardware always lowers cost: higher hourly rates can negate runtime reductions, so measure total billed hours and cost.

15.21 — Health Dashboard security notifications (D2.T2.4)

Domain 2 • Task 2.4 ↑ Back to Question 15.21

A company operates dozens of AWS accounts in AWS Organizations. The security team needs centralized, timely visibility into account-specific AWS-initiated security advisories (for example, RDS CA certificate rotations or TLS deprecations) and wants to automatically notify stakeholders and launch a standard remediation runbook. Which combination of actions best meets these requirements with minimal manual effort?

Options & Rationales

A ✗ — Rely on the public Service health view to monitor security notifications affecting your resources.

Rationale: The public view shows service-wide issues. Many security advisories are account-specific and do not appear there.

B ✗ — Use Amazon GuardDuty to receive AWS-initiated advisories such as RDS CA certificate rotations.

Rationale: GuardDuty detects threats and compromise indicators. AWS-initiated

advisories and required actions are delivered through AWS Health, not GuardDuty.

C ✓ — Enable AWS Health Organizational View with a delegated administrator to aggregate account-specific events across all member accounts.

Rationale: Organizational View centralizes Health events from all accounts in an organization so operations teams can coordinate response from a single place.

D ✓ — Create an Amazon EventBridge rule for AWS Health events to publish to Amazon SNS and start an AWS Systems Manager Automation runbook.

Rationale: EventBridge routes Health notifications for alerting and can trigger Systems Manager Automation documents to standardize remediation steps.

E ✗ — Do nothing; AWS will automatically remediate certificate rotations and TLS deprecations without customer action.

Rationale: AWS Health includes deadlines and recommended actions; many advisories require customer remediation steps to avoid impact.

Explanation

AWS Health Dashboard surfaces security-related events that may affect a customer's resources. It has two distinct views: the public Service health view for AWS service issues by Region, and the Your account view for account-specific advisories such as RDS CA certificate rotations, TLS version deprecations, and required configuration changes. Each event includes scope, an impact timeline, and recommended actions so customers can assess urgency and plan remediation.

In multi-account environments, AWS Organizations adds central visibility via AWS Health Organizational View. A delegated administrator can aggregate account-level Health events across all member accounts, enabling a central operations or security team to see which accounts and resources are affected and coordinate responses.

For automation, the AWS Health API integrates with Amazon EventBridge. Rules can match Health event patterns and route them to targets such as Amazon SNS for stakeholder notifications and AWS Systems Manager Automation to launch standardized runbooks. This event-driven approach reduces manual effort and helps ensure timely, consistent remediation.

The option to enable Organizational View meets the requirement for centralized visibility across many accounts. The option to create an EventBridge rule that publishes to SNS and invokes a Systems Manager Automation document satisfies the need to notify stakeholders and automatically kick off remediation steps.

Relying only on the public Service health view is insufficient because many security notifications are account-specific and never published publicly. Using Amazon GuardDuty would not meet the requirement because GuardDuty

focuses on threat detection for compromised resources, not AWS-initiated advisories and required customer actions. Assuming AWS will automatically remediate certificate rotations or deprecations is risky; Health events commonly include deadlines and recommended actions customers must take to avoid impact.

Key Concepts

- **AWS Health views:** Public Service health shows service-wide events; Your account view lists account-specific advisories with scope, timeline, and actions.
- **Organizational View:** Aggregates Health events across all accounts in AWS Organizations via a delegated administrator for centralized operations.
- **Event-driven automation:** EventBridge routes AWS Health events to targets like SNS and Systems Manager Automation to notify and run standard runbooks.
- **Shared responsibility:** AWS informs customers about advisories; customers must act on required changes to protect their workloads.

Common Pitfalls

- Monitoring only the public Service health page and missing account-specific security advisories.
- Confusing GuardDuty threat findings with AWS Health advisories that require customer-initiated remediation.

15.22 — Responsibility shifts for serverless (D2.T2.1)

Domain 2 • Task 2.1

↑ Back to Question 15.22

A team deploys an AWS Lambda function to process customer uploads. Which responsibility clearly remains with the customer when using this serverless service?

Options & Rationales

A ✗ — Applying operating system patches to the virtual machines that run the Lambda functions.

Rationale: The provider maintains the underlying execution infrastructure and performs maintenance tasks like OS patching for serverless runtimes, so customers do not patch those VMs.

B ✗ — Automatically creating and managing encryption keys for the customer's data without any configuration.

Rationale: While AWS offers key management services, customers must choose and configure encryption options and key usage policies for their data; it is not entirely automatic without customer decisions.

C ✗ — Scaling the function's code to meet an increase in concurrent requests without any customer input.

Rationale: Although Lambda automatically scales execution instances, customers still design code and concurrency settings; the core scaling of infrastructure is handled by AWS, not the customer.

D ✓ — Ensuring the function's application code and the permissions it uses follow least-privilege principles.

Rationale: Customers own and must secure their code, dependencies, and IAM permissions for Lambda functions; this includes granting minimal privileges and protecting application logic.

Explanation

Serverless platforms move responsibility for managing servers and the execution environment to the cloud provider, but users keep responsibility for what they deploy and how it is configured. In this scenario, the customer supplies the function implementation and must control who and what that function can access. The correct choice emphasizes that the team must secure application code, dependencies, and identity permissions and apply the principle of least privilege.

Why the other options are wrong:

- The provider is responsible for maintaining the execution environment and underlying hosts; customers do not apply OS patches for Lambda runtime hosts.
- Encryption support exists, but customers choose key management approaches and configure encryption settings; AWS does not unilaterally create and manage all keys without customer intent.
- Lambda provides automatic scaling of execution instances, so the cloud handles scaling, but that does not remove the customer's duty to write scalable code and set concurrency or throttling preferences.

Key Concepts

- **Shared responsibility for serverless:** The cloud provider operates and maintains infrastructure and execution environment; the customer manages application code and access control.
- **Least privilege:** Grant functions only the permissions they need to perform tasks to reduce risk from compromised code.

Common Pitfalls

- Assuming the provider secures application logic or permissions by default leads to under-protected code and excessive privileges. Avoid conflating infrastructure maintenance with application security.

Glossary

- **Lambda function:** A user-provided unit of code executed by the serverless platform.
- **Least privilege:** A security principle that limits access rights to the minimum necessary.

15.23 — Boundaries of shared responsibility (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 15.23

A company runs web applications on AWS managed services and wants to know which security task stays with their team rather than AWS. Which task is the customer's responsibility?

Options & Rationales

A ✓ — Managing user accounts, permissions, and multi-factor authentication for the application's users.

Rationale: Identity and access setup for application users is the customer's responsibility; AWS provides the platform but customers control who can access their resources.

B ✗ — Physically securing the data center buildings where the cloud servers reside.

Rationale: Physical protection of data center facilities is managed by the cloud provider, not by the customer.

C ✗ — Patching and maintaining the underlying hypervisor used to host virtual machines.

Rationale: Maintenance of the virtualization layer is handled by the cloud provider as part of infrastructure responsibilities.

D ✗ — Designing and operating the global fiber network that connects AWS Regions.

Rationale: The provider is responsible for the global networking infrastructure; customers do not operate that network.

Explanation

Cloud security duties are split so customers know which protections they must implement. The customer is responsible for controlling access to their accounts and resources: creating and managing user identities, assigning permissions, and enforcing multi-factor authentication for their application users. Actions like user provisioning, role policies, and access reviews fall to the customer because they determine who should access their data and services.

Tasks such as securing physical data center sites, maintaining the hypervisor that hosts virtual machines, and operating the provider's global network are handled by the cloud provider. These are infrastructure-level controls the provider implements and operates on behalf of all customers.

Why the correct choice is right and others are wrong. The correct task involves user accounts and authentication, which are rooted in the customer's responsibility to secure access to their resources. Physical facility security, hypervisor patching, and the global connectivity fabric are infrastructure services the cloud provider owns and operates; customers do not perform those tasks.

Key Concepts

- **Access and Identity Management:** Customers configure user accounts, permissions, and authentication methods to control who can reach their applications and data.
- **Provider Infrastructure Controls:** Physical security, hypervisor maintenance, and global network operation are provided and maintained by the cloud vendor as part of the managed platform.

Common Pitfalls

- Assuming the provider configures application-level access controls for you; customers must set identity and authorization for their own users.
- Confusing managed services (where the provider handles more) with responsibilities that still require customer configuration, such as user roles and permissions.

15.24 — Lambda pricing factors: requests/duration (D4.T4.1)

Domain 4 • Task 4.1 ↑ Back to Question 15.24

An online photo service runs a Lambda-based image API with approximately 10,000,000 invocations per month. Traffic is very steady. The average execution time is 200 ms with a 1 GB memory setting. A three-month change freeze prevents any code, memory, runtime architecture, or networking changes. Latency is acceptable; the only goal is to reduce Lambda compute charges with minimal effort.

Which is the BEST approach?

Options & Rationales

A ✗ — Enable Provisioned Concurrency to pre-initialize the function and lower duration-based costs.

Rationale: Provisioned Concurrency adds an hourly charge and you still pay normal request and duration charges. It targets latency, not cost reduction, and can increase total spend.

B ✗ — Migrate the function to arm64 (AWS Graviton) for a lower per-ms compute rate.

Rationale: arm64 on Graviton can lower compute rates, but changing the runtime architecture violates the change freeze.

C ✗ — Lower Amazon CloudWatch Logs retention to 7 days to cut Lambda costs.

Rationale: This reduces CloudWatch Logs charges, not Lambda compute charges. The goal is specifically to reduce Lambda compute spend.

D ✓ — Purchase a Compute Savings Plan sized to the steady baseline; it automatically reduces eligible Lambda compute charges.

Rationale: Compute Savings Plans can reduce Lambda compute charges for

steady workloads and require no code or configuration changes, fitting the change freeze and minimal-effort goal.

Explanation

AWS Lambda uses a pay-for-value model based on request count and compute duration (GB-seconds). Duration pricing scales with the configured memory; more memory increases the per-ms rate but can shorten runtime. The Lambda Free Tier provides 1,000,000 requests and 400,000 GB-seconds each month. For steady workloads, Compute Savings Plans can reduce Lambda compute charges without any code or configuration changes. Provisioned Concurrency is a latency optimization that adds a separate hourly charge while normal request and duration charges still apply. Logging (CloudWatch Logs) and other supporting services are billed separately from Lambda compute.

Using the scenario's steady workload, monthly compute usage can be estimated:

$$\text{GBs} = R \times t \times m = 10{,}000{,}000 \times 0.2 \times 1 = 2{,}000{,}000 \text{ GB-seconds}$$

After the Free Tier:

$$\text{GBs}_c = \max(2{,}000{,}000 - 400{,}000, 0) = 1{,}600{,}000 \text{ GB-seconds}, \quad R_c = \max(10{,}000{,}000 - 1{,}000{,}000, 0) =$$

Variables used:

- **R:** total requests (count)
- **t:** average duration (seconds)
- **m:** memory setting (GB)
- **GBs:** total compute usage (GB-seconds)
- **GBs_c, R_c:** chargeable amounts after Free Tier

Given the change freeze, the team cannot optimize code paths, adjust memory, or switch to arm64. Because traffic is steady and latency is acceptable, the most effective, low-effort way to reduce Lambda compute charges is adopting a Compute Savings Plan, which automatically applies discounted rates to eligible Lambda compute usage. Enabling Provisioned Concurrency focuses on latency and adds an hourly charge. Adjusting CloudWatch Logs retention affects logging costs, not Lambda compute charges.

Key Concepts

- **Lambda pricing meters:** Charges are driven by request count and GB-seconds; memory size affects the per-ms rate and effective CPU/network allocation.
- **Free Tier:** 1,000,000 requests and 400,000 GB-seconds per month reduce billable usage first.
- **Compute Savings Plans:** For steady workloads, they reduce Lambda compute charges without code changes and apply automatically to eligible usage.
- **Provisioned Concurrency:** Improves cold-start latency but adds an hourly

charge while normal request/duration charges still apply.

Common Pitfalls

- Assuming Provisioned Concurrency lowers costs because it reduces cold starts; it adds a separate hourly charge and is for latency, not cost savings.
- Trying to switch to arm64 during a change freeze, or thinking CloudWatch Logs retention changes reduce Lambda compute charges.

15.25 — Third-party auditors validate controls (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 15.25

A retailer is building a PCI cardholder data environment on AWS using Amazon S3. They download the AWS PCI DSS Attestation of Compliance (AoC) from AWS Artifact. Their plan is to use Amazon S3 in 5 AWS Regions. The AoC lists Amazon S3 as in scope for 3 of those Regions. How many planned Region deployments are not covered by the AWS AoC and therefore require additional due diligence on provider controls?

Options & Rationales

A ✗ — 0 Regions

Rationale: Incorrect. Assuming full coverage is a common pitfall. The AoC lists only 3 of the 5 planned Regions, so there are Regions outside the provider's attestation.

B ✗ — 3 Regions

Rationale: Incorrect. This results from a subtraction error. With 5 planned Regions and 3 in scope, only 2 Regions are out of scope, not 3.

C ✗ — 5 Regions

Rationale: Incorrect. That would mean none of the planned Regions are covered, which contradicts the AoC stating 3 Regions are in scope.

D ✓ — 2 Regions

Rationale: Correct. The AoC covers 3 of the 5 planned Regions, so 5 − 3 = 2 Region deployments fall outside the provider's attested scope and need extra due diligence.

Explanation

Auditor attestations and certifications for AWS (such as PCI DSS, SOC 1/2/3, and ISO 27001 series) validate the design and operating effectiveness of AWS provider controls over a defined reporting period and scope. Each report states which services and Regions are covered. Customers access these documents via AWS Artifact to perform vendor due diligence and map AWS's controls to their own compliance requirements.

Under the shared responsibility model, AWS secures the cloud (infrastructure, managed services, and foundational controls), while customers secure what they build in the cloud (identity, configuration, encryption, logging, and

evidence). A key step is verifying that the specific services and Regions a workload will use are included in an auditor's scope. If a planned Region or service is not listed, the provider's controls for that combination are not covered by the attestation—so the customer cannot rely on that report for those parts and must conduct additional due diligence. Regardless of scope, customers must always implement and evidence their own responsibilities (for example, least privilege with IAM, encryption with KMS, logging with CloudTrail, and configuration baselines with AWS Config).

In this scenario, the company plans to use Amazon S3 in 5 Regions, but the PCI DSS AoC lists S3 in scope for only 3 of those Regions. The number of planned Region deployments not covered by the provider attestation is computed as $5 - 3 = 2$. Those 2 Regions require extra due diligence (e.g., confirming provider controls or adjusting the deployment), and the company must still implement and document its own controls everywhere.

Why other counts are wrong. claiming zero Regions out of scope assumes provider compliance applies everywhere—a common pitfall. Claiming five Regions out of scope ignores the stated coverage of three Regions. Claiming three Regions out of scope is a simple subtraction error given the scenario's numbers.

Variables used:

- N_{plan}: Planned Regions for Amazon S3 [count] = 5
- N_{scope}: Regions in AoC scope for Amazon S3 [count] = 3
- N_{out}: Regions not covered [count] = $N_{plan} - N_{scope} = 2$

Key Concepts

- **Shared Responsibility Model:** AWS secures the cloud; customers secure configurations, access, encryption, logging, and evidence for their workloads.
- **Auditor Report Scope:** Attestations list covered services and Regions, plus test procedures, exceptions, and coverage dates.
- **AWS Artifact:** Central portal to obtain AWS compliance reports for vendor due diligence.
- **Verify Coverage:** Only rely on provider attestations where your exact service/Region is in scope; implement your own controls everywhere.

Common Pitfalls

- Assuming AWS compliance automatically makes a workload compliant or covers all Regions, leading to missed gaps in service/Region scope.

15.26 — Permissions for programmatic automation (D3.T3.1)

Domain 3 • Task 3.1 ↑ Back to Question 15.26

A development team needs an automated process on EC2 instances to read objects from a private S3 bucket. The access should be time-limited, auditable, and avoid embedding long-lived credentials on the instances. Which option is

the BEST choice to grant the EC2 instances the needed permissions?

Options & Rationales

A ✓ — Attach an IAM role to the EC2 instances using an instance profile that grants read access to the bucket.

Rationale: An IAM role provided through an instance profile gives EC2 temporary credentials that rotate automatically, are auditable, and avoid hard-coded long-lived keys on the instance.

B ✗ — Create an IAM user with programmatic access and place its access key and secret on each EC2 instance.

Rationale: IAM users use long-lived credentials which increase exposure risk and require manual rotation; they do not provide the automatic, time-limited credentials desired.

C ✗ — Attach a resource-based policy to the S3 bucket allowing the EC2 instance ID to access objects.

Rationale: S3 resource policies grant permissions to principals (like roles or accounts), not to EC2 instance IDs directly, so this does not provide temporary, auditable instance credentials.

D ✗ — Configure the application on the instance to call AWS STS with hard-coded IAM user credentials to obtain temporary tokens.

Rationale: This still relies on embedded long-lived credentials to call STS, reintroducing the credential exposure the scenario seeks to avoid.

Explanation

The primary requirement is secure, temporary, and auditable programmatic access from EC2 to an S3 bucket without placing long-lived credentials on the instance. Attaching an identity designed for delegation to compute resources meets these needs by providing time-limited credentials that are rotated by AWS and recorded in audit logs.

Why the correct choice fits. Attaching an IAM role to the EC2 instances via an instance profile gives the running workloads temporary credentials tied to that role. AWS automatically rotates the credentials and actions taken using the role are recorded in AWS audit logs, enabling traceability. This avoids embedding static access keys on instances and supports least-privilege by scoping a role policy to only the read actions required on the specific bucket.

Why the incorrect choices fail. Creating an IAM user and placing its access keys on instances introduces long-lived secrets that must be manually rotated and are more likely to be exposed; this contradicts the requirement for time-limited credentials. Granting permissions in an S3 resource policy must reference IAM principals (for example a role or account); you cannot directly grant to an EC2 instance identifier, so it does not solve the credential management need. Using STS with hard-coded IAM user credentials still

depends on embedded long-lived keys to obtain temporary tokens, which again fails the goal of avoiding long-lived credentials on the instance.

Key Concepts

- **IAM role for compute:** A role is an identity that workloads can assume to obtain temporary credentials appropriate for the tasks they perform.
- **Instance profile:** The mechanism by which an IAM role is associated with an EC2 instance so the instance can use the role's temporary credentials.
- **Temporary credentials:** Short-lived access tokens reduce exposure from leaked keys and are rotated automatically.

Common Pitfalls

- Assuming a resource policy can target an EC2 instance directly; S3 resource policies require IAM principals.
- Thinking STS solves credential exposure if it is invoked using long-lived embedded keys; the initial secret would still be exposed.

Glossary

- **Principal:** An entity (user, role, or service) that can perform actions in AWS.
- **Audit logs:** Records of API activity (for example, CloudTrail) used to trace which principal performed actions and when.

15.27 — Trust & Safety handles abuse reports (D4.T4.3)

Domain 4 • Task 4.3 ↑ Back to Question 15.27

An e-commerce company's login API is being hit by thousands of credential-stuffing attempts from a single public IP that reverse-resolves to an Amazon EC2 address. The company does not control that AWS account and wants AWS to investigate and stop the activity at its source. What is the BEST action to take with AWS?

Options & Rationales

A ✗ — Open a highest-severity AWS Support technical case asking AWS to disable the attacking instance.

Rationale: AWS Support helps operate workloads and uses severity levels; it is not the abuse enforcement channel. Trust & Safety handles AUP violations separately and does not require a paid plan.

B ✓ — Submit an abuse report to AWS Trust & Safety with the EC2 source IP, concise log excerpts, and precise UTC timestamps so AWS can notify the owning account.

Rationale: AWS Trust & Safety handles AUP violations. Effective reports include source IP, short logs, and UTC timestamps. They notify the resource owner and can restrict resources if needed.

C ✗ — Enable AWS Shield Advanced and add an AWS WAF rule to block the IP so AWS is alerted to take down the source.

Rationale: Shield Advanced and WAF can mitigate traffic to your app but do not notify Trust & Safety or stop abuse at the source. They are not the correct reporting path.

D ✗ — Use WHOIS to identify and email the customer using that EC2 IP because AWS cannot contact other customers due to privacy.

Rationale: Direct outreach is unreliable. Trust & Safety will contact the owning AWS account while honoring privacy and can enforce the Acceptable Use Policy if needed.

Explanation

AWS Trust & Safety is the specialized function that receives and investigates reports of abuse that may violate the AWS Acceptable Use Policy (AUP), such as phishing, malware, command-and-control, unauthorized scanning, DDoS launch activity, or credential stuffing. When abuse originates from resources you do not control (for example, an EC2 instance in another account), the correct path is to submit an abuse report through the Trust & Safety abuse reporting form. Effective submissions include clear indicators: the source IP address (or instance ID/S3 URL), concise log excerpts demonstrating the behavior, and precise timestamps in UTC with time zone details. Trust & Safety triages the report, opens a case with the resource owner via the AWS Support Center, sets remediation expectations, and, if necessary, can implement temporary network or content blocks to stop harm. This process is separate from standard AWS Support severity levels and does not require a paid Support plan.

Opening a standard, high-severity AWS Support case is not appropriate for abuse enforcement because Support focuses on helping customers operate workloads, not AUP violations. Adding AWS Shield Advanced or AWS WAF can help mitigate traffic to your application but does not notify AWS to investigate the offending source or stop it at origin. Attempting to contact the owner directly via WHOIS is unreliable and unnecessary; AWS can identify and notify the owning account while honoring privacy.

Key Concepts

- **AWS Trust & Safety:** Dedicated channel for AUP/policy enforcement; investigates abuse, not general troubleshooting, and can restrict resources if remediation is not prompt.
- **Effective abuse report contents:** Source IP or resource identifier, concise logs, and precise UTC timestamps to enable rapid triage and correlation.
- **Separation from AWS Support:** Abuse handling is outside standard severity levels and does not require a paid Support plan.
- **Shared responsibility and privacy:** AWS notifies the owning account and enforces policy while respecting customer privacy; you supply evidence.

Common Pitfalls

- Opening a generic Support ticket or providing vague descriptions without evidence/timestamps, which delays action and may not reach Trust & Safety.

15.28 — Cost optimization: rightsizing & demand match (D1.T1.2)

Domain 1 • Task 1.2 ↑ Back to Question 15.28

A web application experiences large, unpredictable spikes in user requests during the day and minimal traffic at night. The operations team wants to reduce idle compute costs while keeping performance consistent. Which is the best solution?

Options & Rationales

A ✗ — Purchase Reserved Instances sized for peak traffic to get a lower hourly rate and keep the same number of instances running.

Rationale: Reserved Instances lower unit cost for steady usage but do not automatically change capacity, so they do not prevent paying for idle peak-sized capacity.

B ✗ — Manually add and remove EC2 instances each day based on expected traffic forecasts to match capacity to demand.

Rationale: Manual changes increase operational effort and risk of slow response to unexpected spikes, making it less reliable and efficient than automated scaling.

C ✓ — Use Amazon EC2 Auto Scaling with a target-tracking policy that adjusts instances automatically based on a CPU or request metric.

Rationale: Auto Scaling with target-tracking automatically changes capacity in response to observed load, reducing idle instances while maintaining performance.

D ✗ — Place an Elastic Load Balancer in front of fixed-size instances to distribute traffic evenly, without changing instance count.

Rationale: A load balancer spreads traffic but does not change the number of instances, so it does not reduce costs from idle capacity.

Explanation

Matching supply to demand means automatically adjusting compute capacity to current usage so you pay only for what you need while keeping performance targets. Amazon EC2 Auto Scaling is designed to add or remove instances in response to metrics. A target-tracking policy continuously monitors a chosen metric (for example, average CPU utilization or request count per instance) and changes capacity to keep that metric near the target. This prevents long periods of overprovisioning during low load and avoids under-provisioning during spikes.

A recommendation that uses automated, metric-driven scaling is therefore preferable. Purchasing capacity discounts without dynamic scaling reduces unit cost but leaves you paying for unneeded instances when load falls. Manually changing instance counts requires human intervention and cannot reliably react to sudden demand shifts. Using a load balancer without scaling helps distribute traffic but does not change the amount of provisioned capacity.

Key Concepts

- **Auto Scaling:** Automatically adjusts the number of instances to meet current demand, improving cost efficiency and availability.
- **Target-tracking policy:** Chooses a metric and target value; Auto Scaling increases or decreases capacity to keep the metric near that target.
- **Reserved Instances:** Provides lower pricing for committed instance usage but does not provide automatic capacity changes.

Common Pitfalls

- Assuming a load balancer reduces cost: it only distributes traffic and does not remove idle resources.
- Confusing cost savings with elasticity: discounted long-term capacity still wastes money if capacity remains larger than needed during low load.

This guidance favors automated, metric-based scaling to align capacity with real-time demand, reducing idle compute costs while maintaining performance.

15.29 — AWS Artifact provides reports on-demand (D2.T2.2)

Domain 2 • Task 2.2 ↑ Back to Question 15.29

Which AWS service gives organizations a centralized portal to download formal audit and compliance documents (for example, security attestations and certificates) on demand to support their own regulatory reviews?

Options & Rationales

A ✗ — AWS Config

Rationale: AWS Config records and evaluates resource configurations over time; it does not provide downloadable third-party audit reports or certification documents.

B ✗ — AWS CloudTrail

Rationale: CloudTrail captures account API activity and event history for auditing, but it does not supply formal compliance certificates or external audit reports.

C ✗ — AWS Organizations

Rationale: AWS Organizations manages multiple accounts and organizational policies; it does not serve as a repository for official audit or certification documentation.

D ✓ — AWS Artifact

Rationale: AWS Artifact is the centralized portal that lets customers retrieve formal compliance and audit documents such as third-party attestation reports and certificates.

Explanation

AWS provides a service that acts as a single place where customers can obtain authoritative compliance documents to support their own audits and regulatory needs. This service delivers evidence such as independent audit attestations and compliance certificates that organizations often reference during assessments.

Why the correct choice fits. AWS Artifact is specifically designed to offer on-demand access to official compliance documentation. It centralizes downloadable reports and certificates, making it straightforward for customers to retrieve the evidence they need for internal or external compliance activities.

Why the other choices are incorrect. AWS Config is a configuration tracking and assessment tool that shows how resources change and whether they meet rules; it does not provide external audit certificates. AWS CloudTrail records API calls and user activity for auditing and forensic analysis but does not issue compliance attestations or formal certification documents. AWS Organizations helps manage and govern multiple AWS accounts and apply policies across them, but it is not a repository for audit or certification artifacts.

Key Concepts

- **Centralized compliance access:** A service that aggregates official compliance documents so customers can retrieve them without contacting support. This reduces administrative overhead during audits.
- **Audit evidence vs. activity logs:** Audit evidence refers to formal third-party reports and certifications; activity logs are time-sequenced records of API calls and events used for investigation and monitoring.

Common Pitfalls

- Confusing logging or configuration tools with a document repository leads to the wrong choice. Another mistake is assuming governance or account management services provide downloadable compliance certificates.

Glossary

- **Audit attestation:** A formal report from an independent auditor stating that controls meet a specified standard.
- **Compliance certificate:** A credential issued to indicate conformance with a regulatory or standards body.

15.30 — Post-migration optimization for cost/perf (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 15.30

Select TWO actions that demonstrate right-sizing of compute resources after a migration to reduce cost while maintaining performance

Options & Rationales

A ✗ — Purchase a long-term Savings Plan to lower hourly compute costs for stable baseline usage.

Rationale: Buying Savings Plans reduces cost for steady usage patterns but is a purchasing optimization, not a right-sizing activity that changes resource capacity.

B ✗ — Use a content delivery network to cache static assets closer to users and decrease origin load.

Rationale: Using a CDN improves latency and origin load but does not directly resize compute instances as part of right-sizing.

C ✗ — Move infrequently accessed files to a lower-cost object storage tier to reduce storage spend.

Rationale: Moving data to cheaper storage is a storage-tiering optimization, not a compute right-sizing action.

D ✗ — Enable Auto Scaling to add and remove instances automatically in response to short-term traffic spikes.

Rationale: Auto Scaling manages dynamic capacity but is an elasticity feature; it does not by itself change provisioned instance sizes to eliminate long-term overprovisioning.

E ✓ — Consolidate multiple lightly used workloads onto fewer instances and adjust instance types to match combined demand.

Rationale: Combining low-traffic workloads and choosing appropriately sized instances reduces idle capacity and cost while keeping necessary performance headroom.

F ✓ — Reduce instance families or sizes for underutilized virtual machines after observing low CPU and memory metrics.

Rationale: Selecting smaller instance families or sizes for VMs with sustained low utilization matches capacity to demand and lowers cost without adding operational complexity.

Explanation

Right-sizing focuses on matching compute capacity to actual workload demand so performance is met with minimal wasted cost. That means changing the provisioned compute characteristics (for example, switching to a smaller instance family or size) and consolidating idle workloads so fewer instances carry the same combined load. Observability data (CPU, memory, and other utilization metrics) guides safe reductions.

Why the correct actions work:

- Reducing instance families or sizes for consistently low-utilization virtual machines directly lowers hourly charges and often decreases associated

license or attached-storage costs while maintaining required performance for observed workloads.
- Consolidating multiple lightly used workloads onto fewer instances reduces the number of running instances and eliminates idle capacity; after consolidation, choosing instance types that match the combined CPU and memory profile avoids overprovisioning.

Why the other options are not right-sizing:

- Enabling Auto Scaling provides elasticity for fluctuating demand but does not itself reduce long-term provisioned size when workloads are consistently overprovisioned. It addresses scaling behavior rather than resizing existing instances.
- Purchasing a Savings Plan lowers cost for stable usage but does not change resource capacity; it is a purchasing optimization, not a capacity adjustment.
- Moving infrequently accessed files to cheaper storage is a storage-tiering activity, and using a CDN is an edge/performance optimization—both are valuable but separate from compute right-sizing.

Key Concepts

- **Observability-driven changes:** Use metrics (CPU, memory, latency) to identify sustained underutilization before resizing to avoid performance regressions.
- **Capacity matching:** Choose instance types and sizes that align with actual resource profiles instead of default or peak-based guesses.

Common Pitfalls

- Reducing resources based on short-term low usage can cause performance problems during occasional spikes; verify sustained patterns before resizing.
- Confusing cost-saving purchases (Savings Plans) with capacity changes; both reduce cost but tackle different causes.

Glossary

- **Right-sizing:** Adjusting the provisioned compute capacity to match workload demand to minimize cost while preserving performance.
- **Auto Scaling:** A mechanism to scale capacity in and out automatically in response to demand signals.

15.31 — Protecting the root user (MFA) (D2.T2.3)

Domain 2 • Task 2.3 ↑ Back to Question 15.31

An organization is hardening their AWS account root identity to reduce risk of account takeover. Select TWO actions that directly add a second authentication factor to the root account (Select TWO).

Options & Rationales

A ✗ — Turn on CloudTrail to log all management API calls for the account.

Rationale: Recording API activity helps with detection and forensics after events, but logging does not add a second factor to authentication.

B ✓ — Register a hardware one-time-password device for the root account.

Rationale: A physical OTP device provides a possession-based second factor tied to the root account and prevents access without the physical token.

C ✗ — Store root credentials in an encrypted secret vault and share them with the ops team.

Rationale: Securely storing credentials improves management and auditing, yet it does not provide an additional authentication factor for root sign-in.

D ✗ — Create an administrative IAM user and use that identity for daily tasks.

Rationale: Using an IAM admin reduces reliance on the root account for routine work but does not itself add a second factor to the root credential.

E ✗ — Apply a resource-based policy to prevent root from accessing specific services.

Rationale: Resource policies can limit access patterns but cannot add multi-factor authentication to the root identity.

F ✓ — Enable a virtual time-based MFA authenticator app for the root account.

Rationale: A time-based authenticator app on a trusted device supplies the required second factor without needing extra hardware, protecting the root login.

Explanation

Multi-factor authentication (MFA) requires two different types of proof before granting access: something you know (a password) and something you have (a physical token or an authenticator app). Adding MFA to the root identity directly changes the sign-in process so attackers need both the password and the second factor to succeed.

Registering a hardware one-time-password device for the root account introduces a physical possession factor—an independent device that generates time-limited codes which must be presented during sign-in. Enabling a virtual time-based authenticator app provides the same possession-based protection using software on a trusted device; it generates short-lived codes that complement the root password.

The other choices are useful security practices but do not meet the specific requirement of adding a second authentication factor to the root account. Creating and using an administrative IAM user reduces routine dependence on root, but it does not change root's authentication setup. Storing credentials in an encrypted vault improves secret handling and auditability, yet the vault protects the secret at rest rather than adding an authentication factor during sign-in. Enabling CloudTrail gives visibility into actions for detection and investigation, and resource-based policies can limit access, but neither mechanism enforces an additional authentication step for root.

Key Concepts

- **Multi-Factor Authentication (MFA):** Requires two distinct authentication factors (typically password plus a possession factor), reducing the chance of credential-only compromise.
- **Hardware MFA device:** A physical token that generates one-time passwords and must be physically present to authenticate, providing strong possession assurance.
- **Virtual MFA authenticator:** A software-based TOTP app on a trusted device that generates time-limited codes, offering possession-based second-factor protection without extra hardware.

Common Pitfalls

- Assuming that logging, policies, or secret storage automatically provide multi-factor protection for sign-in is incorrect; they improve detection, control, or secret handling but do not change authentication factors. Another mistake is relying solely on creating an admin IAM user instead of securing the root identity with MFA; both are important but address different risks.

Glossary

- **TOTP:** Time-Based One-Time Password, a short-lived numeric code generated by hardware or software authenticators.
- **Possession factor:** An authentication element the user must physically have (device or token) to prove identity.

15.32 — Lightsail for simple VPS use cases (D3.T3.3)

Domain 3 • Task 3.3 ↑ Back to Question 15.32

A startup hosts a WordPress marketing site on a single Amazon Lightsail instance. They want higher availability and HTTPS with minimal management and predictable monthly cost. They do not need Application Load Balancer features and prefer to stay on Lightsail for now.

Review the exhibit and choose the best next step.

Table 7: Exhibit: Current state and goals

Item	Current	Goal/Constraint
App	WordPress on one Lightsail instance	Stay on Lightsail; minimal changes
Networking	Static public IP; Lightsail DNS	Public HTTPS without managing certs on instance

Item	Current	Goal/Constraint
Availability	Single AZ	Survive an AZ failure; add a second instance
Cost	Low, predictable	Keep predictable monthly pricing
Advanced features	None	No path-based routing; no PrivateLink needed

Options & Rationales

A ✓ — Add a Lightsail load balancer with TLS termination and place two Lightsail instances in different Availability Zones behind it.

Rationale: Meets HTTPS without per-instance certs, provides simple multi-AZ load balancing, and preserves Lightsail's predictable pricing and simplified management. Trade-off: fewer features than an ALB.

B ✗ — Export the instance to Amazon EC2, place it behind an Application Load Balancer, and use an Auto Scaling group across AZs.

Rationale: Would deliver HTTPS and multi-AZ, but contradicts the preference to stay on Lightsail now and adds complexity intended for more advanced requirements.

C ✗ — Peer Lightsail to a VPC and migrate the database to Amazon RDS while keeping the single Lightsail instance with its static IP.

Rationale: Improves database management but does not add load balancing or multi-AZ redundancy for the web tier, and does not remove per-instance certificate management.

D ✗ — Install an SSL/TLS certificate on the Lightsail instance and open port 443 in the instance-level firewall.

Rationale: Enables HTTPS, but requires managing certs on the instance and leaves a single-instance, single-AZ point of failure.

Explanation

Amazon Lightsail is designed for small, predictable workloads with simplified operations. It includes easy-to-use building blocks beyond a single instance, such as a managed load balancer with TLS termination and support for running instances in different Availability Zones. This aligns directly with the scenario's goals: add HTTPS without managing certificates on individual instances and improve availability while maintaining predictable monthly pricing and minimal changes.

Using a Lightsail load balancer provides managed TLS termination, so certificates are handled at the load balancer rather than on each instance.

Placing two Lightsail instances in different Availability Zones behind that load balancer delivers simple multi-AZ resilience. This approach keeps the team on Lightsail, satisfying the constraint of avoiding a move to Amazon EC2 for now, and maintains the service's opinionated, predictable pricing model.

Migrating to Amazon EC2 with an Application Load Balancer and Auto Scaling is a good pattern when teams need more granular instance types, custom networking, or ALB-specific features. However, the scenario explicitly states those advanced features are not needed and the team wants to stay on Lightsail, so that option adds unnecessary complexity now.

Peering Lightsail to a VPC to reach services like Amazon RDS is useful when the database requirements outgrow Lightsail's simplified controls. While that improves database management, it does not by itself add compute high availability or eliminate per-instance certificate management on the web tier.

Installing a certificate directly on the single Lightsail instance enables HTTPS but fails the availability goal (a single point of failure remains) and contradicts the desire to avoid managing certs on instances.

Key Concepts

- **Lightsail load balancer:** Provides simple load distribution and TLS termination, enabling HTTPS without per-instance certificate management.
- **Multi-AZ resilience:** Running instances in different Availability Zones increases availability compared to a single-instance setup.
- **When to move to EC2/ALB:** Export to EC2 for granular instance types, custom networking, Auto Scaling, or ALB features; otherwise, Lightsail's components often suffice.
- **VPC peering from Lightsail:** Enables access to services like Amazon RDS but does not replace the need for web-tier load balancing.

Common Pitfalls

- Assuming database migration alone improves web-tier availability; it does not add load balancing across instances.
- Enabling HTTPS on a single instance meets encryption needs but leaves a single point of failure and increases certificate management overhead.

Glossary

- **TLS termination:** Offloading TLS/HTTPS encryption/decryption to the load balancer instead of each backend instance.
- **Availability Zone (AZ):** A distinct, isolated location within a Region; deploying across AZs reduces the impact of a localized failure.

15.33 — Cross-account access with roles & external ID (D2.T2.3)

Domain 2 • Task 2.3

↑ Back to Question 15.33

A company will allow a third-party SaaS vendor to write backups to an Amazon S3 bucket in the company's AWS account. The company will use a cross-account IAM role for delegation. Which TWO configuration choices are NOT recommended?

Options & Rationales

A ✗ — Use short role session durations appropriate for automation, and require MFA when humans assume the role.

Rationale: Avoid excessively long sessions and add MFA for human assumptions to reduce risk and improve control.

B ✓ — Configure the role's trust policy with Principal set to "*" so any account can assume it.

Rationale: Trusting a wildcard Principal is a common pitfall. Scope the Principal to the vendor's AWS account or specific role ARN to enforce least privilege.

C ✗ — Monitor role assumptions with AWS CloudTrail and create CloudWatch alarms on unusual activity.

Rationale: Monitoring role usage with CloudTrail and CloudWatch provides auditability and detection of misuse.

D ✓ — Share long-term access keys with the vendor instead of using AWS STS AssumeRole temporary credentials.

Rationale: Sharing long-term keys increases risk. Use STS AssumeRole to issue short-lived, auditable credentials instead.

E ✗ — Attach a permissions policy that allows only s3:PutObject to the designated backup bucket.

Rationale: This follows least privilege by limiting actions and scope to the specific S3 bucket used for backups.

F ✗ — Add a Condition in the trust policy requiring sts:ExternalId with a unique value shared with the vendor.

Rationale: Requiring an external ID mitigates the confused deputy problem. The vendor must provide the exact sts:ExternalId to assume the role.

Explanation

Cross-account access is best implemented with an IAM role in the resource account. The role has two parts: a trust policy (who can assume) and a permissions policy (what temporary credentials can do). External principals in another AWS account obtain short-lived credentials by calling AWS STS AssumeRole, which avoids sharing long-term access keys and provides clear CloudTrail auditability.

When delegating to a vendor or SaaS provider, include an external ID in the trust policy using a Condition (such as StringEquals on sts:ExternalId). The customer generates a unique value and shares it with the vendor. STS checks

that the AssumeRole request includes the exact external ID, mitigating the confused deputy problem, especially when a vendor serves many customers.

Least privilege should apply in both the trust and permissions policies. In the trust policy, scope the Principal to the vendor's AWS account or a specific role ARN, not a wildcard. In the permissions policy, grant only the actions and resources required, such as s3:PutObject to a single backup bucket. Operational controls include monitoring role use with AWS CloudTrail and CloudWatch, setting short session durations (avoiding excessively long sessions), and requiring MFA for human role assumptions when appropriate.

The configurations that are not recommended are a trust policy that uses a wildcard Principal, which allows unintended entities to assume the role, and sharing long-term access keys with the vendor instead of using STS AssumeRole, which increases exposure and undermines temporary, least-privilege access.

Key Concepts

- **IAM role trust vs. permissions policy:** Trust policy defines who can assume; permissions policy defines allowed actions for temporary credentials.
- **AWS STS AssumeRole:** Issues short-lived credentials that reduce risk versus long-term keys and support clear auditing.
- **External ID (sts:** ExternalId): Trust policy condition that mitigates the confused deputy risk for third-party vendors.
- **Least privilege:** Narrow Principals and limit actions/resources (for example, only s3:PutObject to a specific bucket).

Common Pitfalls

- Using a wildcard Principal in the trust policy, enabling unintended access.
- Sharing long-term access keys instead of using temporary STS credentials.
- Omitting the external ID for vendor access or setting excessively long session durations.

15.34 — Apply model during compliance audits (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 15.34

Which AWS service is commonly relied on as the primary tamper-evident source of API call records when an auditor needs evidence of who performed management-plane actions?

Options & Rationales

A ✗ — Amazon CloudWatch Logs — stores application and system logs for real-time metrics and alarms.

Rationale: CloudWatch Logs is for metrics and operational logs but is not the primary, tamper-evident API-audit record source.

B ✗ — AWS Config — tracks resource configuration changes and provides a history of resource states.

Rationale: AWS Config records configuration snapshots and change history, useful for compliance of resource state, but does not serve as the main API call audit trail.

C ✓ — AWS CloudTrail — records account API activity and can provide tamper-evident event logs for auditing.

Rationale: CloudTrail captures management-plane API calls and, when combined with features such as log file integrity validation and protected storage (for example, S3 Object Lock or CloudTrail Lake), can provide tamper-evident event logs that auditors rely on.

D ✗ — Amazon GuardDuty — provides threat detection and alerts based on telemetry and anomalies.

Rationale: GuardDuty analyzes telemetry to surface threats and findings, but it is an analysis service rather than the primary source of raw API call records for audits.

Explanation

When auditors need authoritative evidence of management-plane actions in an AWS account, the primary source is the service that records API calls with an audit-focused design. AWS CloudTrail captures who made API requests, when, and from where, and writes events to logs that can be retained and validated as audit evidence. Other services provide complementary information but are not the central API call ledger.

Why the correct choice is right and others are wrong. CloudTrail is designed to record account-level API activity and is the standard evidence source for auditing management-plane actions. CloudWatch Logs is focused on application and system logging and alerting rather than serving as the canonical account API audit trail. AWS Config provides configuration snapshots and change history useful for compliance of resource state, but it does not act as the primary log of API calls. Amazon GuardDuty produces findings from telemetry to highlight suspicious activity but does not replace the detailed API event records that auditors request.

Key Concepts

- **API activity ledger:** A service that records management-plane API calls (who, what, when) and is commonly used as primary audit evidence.
- **Tamper-evident audit logs:** Logs protected with integrity mechanisms such as CloudTrail log file integrity validation and write-once storage (for example, S3 Object Lock or CloudTrail Lake) so that attempts to alter them can be detected.

Common Pitfalls

- Confusing monitoring or analysis services with the raw API call record: a threat detection or metrics service may use API data but does not replace the canonical audit log.

- Assuming configuration history is the same as API call records: resource configuration history shows state changes but does not always include the full API call context required for auditor questions.

15.35 — Six AWS CAF perspectives (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 15.35

A company migrating to AWS needs to standardize resource tagging, allocate costs by business unit, enforce change control, and set guardrails using service control policies. According to the AWS Cloud Adoption Framework (AWS CAF), which perspective should lead this work?

Options & Rationales

A ✗ — Platform perspective

Rationale: Platform focuses on landing zones, networking, compute, storage, and databases. It builds technical foundations but does not own tagging/cost policies or change control governance.

B ✓ — Governance perspective

Rationale: Governance focuses on portfolio, risk, compliance, and financial management. In practice, it establishes tagging, cost allocation, change control, and guardrails with service control policies.

C ✗ — Security perspective

Rationale: Security addresses identity, detection, protection, data privacy, and incident response. While related to protection, it does not lead tagging, cost allocation, or change-control guardrails.

D ✗ — Operations perspective

Rationale: Operations covers the operational model, observability, automation, backup, and resilience (e.g., CloudWatch, AWS Backup). It is not the lead for tagging, cost, and policy guardrails.

Explanation

AWS Cloud Adoption Framework (AWS CAF) organizes migration guidance into six perspectives that align people, process, and technology. The business-focused perspectives are Business, People, and Governance; the technology-focused perspectives are Platform, Security, and Operations. Each perspective clarifies responsibilities so teams can sequence work and avoid gaps during migration.

Governance is responsible for portfolio, risk, compliance, and financial management. In practical terms during migration, Governance establishes standards and controls such as resource tagging, cost allocation practices, change control, and guardrails with service control policies. Those activities ensure accountability and prevent cost surprises while aligning work to business risk and compliance requirements.

By contrast, Platform builds the technical foundations like the landing zone, VPC design, and selecting managed services. Security focuses on identity, detection, protection, data privacy, and incident response, applying measures such as encryption and logging. Operations handles the operational model, observability, automation, backup, and resilience, including metrics/alarms and runbooks. These are essential but do not lead the tagging, cost allocation, and change-control guardrail efforts.

Therefore, the Governance perspective is the correct choice because it directly owns the controls cited in the scenario.

Key Concepts

- **AWS CAF:** A guidance model with six perspectives that align people, process, and technology for migration.
- **Governance perspective:** Owns portfolio, risk, compliance, and financial management; implements tagging, cost allocation, change control, and guardrails with service control policies.
- **Platform/Security/Operations roles**: Platform builds technical foundations; Security handles protection and monitoring; Operations runs and automates day-to-day operations.

Common Pitfalls

- Assuming Security leads all guardrails because they sound protective; in CAF, tagging, cost allocation, and change control guardrails are led by Governance.

15.36 — When to require MFA (D2.T2.3)

Domain 2 • Task 2.3

↑ Back to Question 15.36

Which statement best describes the IAM condition key aws:MultiFactorAuthPresent?

Options & Rationales

A ✓ — It evaluates to true only when the request uses MFA-authenticated credentials, allowing you to require MFA for console and AWS STS sessions.

Rationale: aws:MultiFactorAuthPresent indicates whether MFA was used for the request. It is commonly used to enforce MFA for console access and for STS AssumeRole/GetSessionToken sessions.

B ✗ — It sets a maximum time since the last MFA verification to limit session duration.

Rationale: That behavior corresponds to aws:MultiFactorAuthAge. MultiFactorAuthPresent only checks whether MFA was used, not how recently.

C ✗ — It enables MFA for all users at AWS IAM Identity Center sign-in by itself.

Rationale: Identity Center MFA is configured within IAM Identity Center settings. The aws:MultiFactorAuthPresent key does not control Identity Center sign-in policy.

D ✗ — It automatically replaces all long-lived IAM user access keys with short-lived credentials.

Rationale: Short-lived credentials are obtained via AWS STS (e.g., AssumeRole, GetSessionToken). A condition key does not create or rotate credentials.

Explanation

Multi-factor authentication (MFA) adds a second verification factor to reduce account-takeover risk. In AWS, you can require MFA in multiple ways depending on the access path. For IAM policies and role assumptions, condition keys let you express requirements that must be true for a request to be allowed. The aws:MultiFactorAuthPresent condition key evaluates whether the request was made with MFA. If true, the requester authenticated with a second factor and the resulting session or credentials are MFA-backed. You can use this key in identity-based policies or role trust policies to require MFA for sensitive actions, console access, or for obtaining temporary credentials via AWS Security Token Service (STS) operations such as AssumeRole or GetSessionToken.

To control how recently MFA was performed, a separate key, aws:MultiFactorAuthAge, can be used to enforce a maximum age since the last MFA challenge. This is different from merely checking presence. For workforce sign-in through AWS IAM Identity Center, MFA is configured directly in Identity Center; IAM condition keys do not turn on MFA there. Finally, creating short-lived, MFA-backed sessions is done by using STS to issue temporary credentials; condition keys themselves do not generate or rotate credentials—they only gate access based on conditions.

Why the correct statement fits. It captures that aws:MultiFactorAuthPresent is a boolean indicator used to require MFA for console and STS-derived sessions. The statement about limiting time since MFA confuses it with aws:MultiFactorAuthAge. The statement about enabling Identity Center MFA misplaces control; that is configured in Identity Center, not via IAM condition keys. The statement about automatic short-lived credentials attributes STS functionality to a condition key, which is incorrect.

Key Concepts

- **MFA presence vs. age:** aws:MultiFactorAuthPresent checks if MFA was used; aws:MultiFactorAuthAge constrains how recent the MFA was.
- **STS temporary credentials:** Use AssumeRole/GetSessionToken to obtain short-lived, MFA-backed credentials for programmatic access.
- **Role trust policy conditions:** You can require MFA when assuming high-privilege roles by adding conditions that reference MFA keys.
- **Identity Center MFA:** Enforced at sign-in through IAM Identity Center configuration, separate from IAM condition keys.

Common Pitfalls

- Confusing presence (was MFA used?) with age (how recently MFA was used?).
- Assuming IAM condition keys can enable or configure MFA in Identity Center or automatically rotate credentials.

Glossary

- **IAM condition key:** A policy attribute evaluated at request time to allow or deny actions based on context.
- **MFA-backed session:** A session where the user authenticated with a second factor; often required for sensitive operations.

15.37 — Marketplace simplifies software procurement (D3.T3.8)

Domain 3 • Task 3.8 ↑ Back to Question 15.37

Select TWO benefits that are provided when an organization uses AWS Marketplace consolidated billing for third-party software purchases

Options & Rationales

A ✗ — Purchases through Marketplace always include automatic software updates managed by AWS.

Rationale: Software update responsibilities depend on the vendor and product type; Marketplace billing does not guarantee automatic updates.

B ✓ — Marketplace charges are combined with other AWS service costs to simplify internal chargeback and cost allocation.

Rationale: Including Marketplace fees on the AWS bill helps link third-party spend with account tagging and cost reports for easier chargeback.

C ✗ — AWS Marketplace issues separate invoices per vendor to help reconcile each supplier individually.

Rationale: This contradicts consolidated billing; Marketplace centralizes charges rather than creating separate invoices per vendor.

D ✓ — All Marketplace software charges appear on a single AWS invoice for centralized payment.

Rationale: Consolidated billing aggregates Marketplace charges into one invoice, simplifying payment and invoice management across multiple purchases.

E ✗ — Marketplace automatically deploys vendor images into a private network using PrivateLink.

Rationale: PrivateLink is a networking feature and Marketplace billing does not perform network deployments.

F ✗ — Marketplace converts all license types to a single universal license managed by AWS.

Rationale: Licensing terms remain defined by each software vendor; Marketplace does not standardize licenses into one universal type.

Explanation

Consolidated billing for Marketplace means third-party software charges are combined with the AWS account's billing so organizations receive a unified invoice and can manage payments centrally. This feature does not change how software is deployed, how vendors handle updates, or the original licensing terms; it only affects how charges are presented and processed.

Why the correct answers are right. The statements about a single AWS invoice and simplified internal chargeback describe the billing consolidation benefit: Marketplace purchases appear with other AWS service charges, enabling centralized payment and simpler cost attribution.

Why the incorrect answers are wrong. The other choices describe deployment, network, update, or licensing behaviors that are outside billing responsibilities. Marketplace billing does not perform network plumbing like PrivateLink, does not enforce vendor update policies, does not create separate vendor invoices when billing is consolidated, and does not standardize vendor license models.

Key Concepts

- **Consolidated billing:** Aggregates charges from Marketplace purchases and AWS services into a single invoice for an account or payer family, reducing separate vendor invoices.
- **Cost allocation & chargeback:** When charges are on the AWS invoice, organizations can use account tagging and Cost Explorer to attribute third-party spend to teams or projects for internal billing.

Common Pitfalls

- Confusing billing features with deployment or licensing functionality can lead to incorrect assumptions about what Marketplace manages.
- Assuming that Marketplace changes vendor responsibilities (like updates or license terms) rather than only affecting procurement and invoicing.

Glossary

- **Marketplace charge:** A fee for third-party software procured through AWS Marketplace that can appear on the AWS invoice.
- **Chargeback:** Internal process to attribute cloud costs to business units or projects for accounting purposes.

15.38 — Redshift petabyte-scale data warehouse (D3.T3.7)

Domain 3 • Task 3.7

↑ Back to Question 15.38

A company needs interactive BI dashboards that run complex SQL joins across billions of rows with high user concurrency during business hours. Most historical data is stored as partitioned Parquet files in Amazon S3, while current-day transactions reside in Amazon Aurora. The team wants sub-second to seconds-level response times, to absorb short-lived peaks without overprovisioning, and to avoid cluster management while paying only for processing used. Which approach best meets these goals?

Options & Rationales

A ✗ — Amazon Athena to run serverless SQL directly on S3; pay-per-query for low operations overhead and ad hoc analytics.

Rationale: Athena is serverless and cost-efficient for file-oriented queries, but Redshift is preferred for sustained, high-concurrency dashboards with complex joins and low latency.

B ✗ — Provisioned Amazon Redshift RA3 cluster using Concurrency Scaling and Redshift Spectrum to balance performance and data lake access.

Rationale: This can deliver strong performance, but it still requires cluster sizing and management, which the scenario explicitly wants to avoid.

C ✗ — Amazon RDS for PostgreSQL with read replicas to offload reporting from production and handle analytics at scale.

Rationale: RDS/Aurora serve as operational stores; the scenario calls for a data warehouse with S3 integration and high-concurrency analytics, which aligns with Redshift rather than RDS.

D ✓ — Amazon Redshift Serverless with Redshift Spectrum to query S3 and federated queries to join Aurora; automatic scaling and no cluster management for consistent, low-latency analytics.

Rationale: Redshift Serverless removes cluster management, scales automatically for bursts, and supports Spectrum and federated queries—matching the performance, elasticity, and integration needs described.

Explanation

The workload describes interactive BI dashboards with complex joins, petabyte-scale history, and high concurrency. Amazon Redshift is AWS's fully managed, columnar, MPP data warehouse designed for analytical (OLAP) workloads that require consistent, low-latency queries at scale. It tightly integrates with an S3 data lake via Redshift Spectrum (query open formats in S3 without moving data) and can join to operational data in Amazon RDS/Aurora using federated queries. Two deployment patterns are relevant: provisioned clusters (e.g., RA3 nodes with managed storage) and Redshift Serverless.

The requirement to avoid cluster management and pay only for processing used, while still handling bursts without overprovisioning, maps directly to Redshift Serverless. Serverless eliminates cluster sizing, scales automatically for peak periods, and is billed for actual processing. In this architecture, Spectrum

can access historical S3 data and federated queries can pull current-day records from Aurora, enabling unified analytics with the sub-second to seconds-level performance Redshift is known to deliver for high-concurrency dashboards.

A provisioned RA3-based Redshift cluster with Concurrency Scaling and Spectrum could also meet performance needs, but it conflicts with the stated constraint to avoid cluster management and pre-sizing. Athena is serverless and cost-effective for file-oriented queries on S3, but compared to Athena, Redshift excels when sustained, high-concurrency analytics and complex joins with predictable low latency are required. Using Amazon RDS for analytics is misaligned here: the context positions RDS/Aurora as operational stores that Redshift can federate to; the workload needs a warehouse with S3 lake integration and high-concurrency performance features provided by Redshift.

Key Concepts

- **Amazon Redshift Serverless:** Removes cluster management, bills for processing used, and automatically scales to absorb bursts.
- **Redshift Spectrum:** Queries open data formats in Amazon S3 alongside warehouse tables, reducing data movement and cost.
- **Federated queries (Redshift):** Join data from operational stores like Amazon RDS and Aurora within Redshift queries.
- **Redshift vs. Athena:** Redshift is preferred for sustained, high-concurrency, complex analytics with low-latency responses; Athena suits serverless, file-oriented queries.

Common Pitfalls

- Assuming "serverless" always implies Athena, overlooking Redshift Serverless for high-concurrency, low-latency analytics.
- Choosing RA3 clusters for flexibility but ignoring the explicit "no cluster management" requirement.

15.39 — CloudFront CDN for global distribution (D3.T3.5)

Domain 3 • Task 3.5 ↑ Back to Question 15.39

A company pointed its website traffic to an Amazon CloudFront distribution to reduce latency and lighten load on the origin. After deployment, the origin server still receives nearly every user request and global users report unchanged latency. Which is the most likely reason CloudFront is not reducing origin traffic as expected?

Options & Rationales

A ✗ — The CloudFront distribution is deployed only to a single Availability Zone, so most users reach the origin instead of an edge location.

Rationale: CloudFront uses edge locations (points of presence) worldwide rather than Availability Zones; distributions are not limited to a single AZ, so this is not the cause.

B ✗ — The distribution was created without an origin configured, so CloudFront cannot cache content and always contacts the origin.

Rationale: A CloudFront distribution requires a configured origin; if an origin were missing CloudFront would not serve requests, so continual origin traffic would not occur for a valid distribution.

C ✗ — Edge locations are geographically distant from users, so requests are routed to the origin instead of being served from cache.

Rationale: Edge locations are globally distributed to serve nearby users; geographic distance alone does not cause CloudFront to bypass cache and forward every request to the origin.

D ✓ — The origin's responses include headers that instruct caches not to store content, so CloudFront forwards each request to the origin.

Rationale: If responses carry cache-control or similar directives that prevent caching, CloudFront will not keep objects at edge locations and will fetch from the origin for each request, keeping origin load high and not improving latency.

Explanation

CloudFront reduces latency and origin load by keeping copies of objects at edge locations so user requests can be served from nearby caches. For caching to work, responses from the origin must be cacheable. When the origin includes HTTP headers (for example, cache-control: no-store or similar) or when cache rules are configured to not cache those responses, CloudFront will forward each request to the origin rather than serving from an edge cache. That behavior results in high origin traffic and no improvement in latency.

Why the other choices are incorrect. Stating that a distribution is limited to a single Availability Zone misunderstands CloudFront architecture; it operates from globally distributed edge locations, not AZ-limited endpoints. A missing origin would prevent CloudFront from serving requests, so repeatedly hitting the origin is not consistent with a properly deployed distribution with no caching. Suggesting edge locations are simply too distant is inaccurate because CloudFront uses many worldwide points-of-presence to serve nearby users; distance alone does not force bypassing caches.

Key Concepts

- **Caching at the edge:** Content stored at nearby points-of-presence allows CloudFront to serve users without contacting the origin for each request, reducing latency and origin load.
- **Cacheability rules:** Responses must be allowed to be cached (by headers or distribution cache behavior) for CloudFront to keep objects at edge locations.

Common Pitfalls

- Assuming CloudFront automatically caches everything: if origin headers or cache behavior settings prevent caching, CloudFront will not reduce origin traffic.

Glossary

- **Edge location**: A geographically distributed point-of-presence where cached content is stored to serve user requests with lower latency.
- **Origin**: The source server or storage location that holds the original version of content.

15.40 — Snowball/Snowcone for large data transfer (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 15.40

A media company must migrate 600 TB of archived video from its on-premises data center to Amazon S3 in the us-east-1 Region within 10 days. The network team can allocate at most 200 Mb/s sustained bandwidth for the migration. The data consists of millions of small files. The security team requires encryption using AWS KMS keys and a tamper-resistant, trackable process. Which is the BEST solution to meet the deadline and requirements?

Options & Rationales

A ✓ — Order multiple AWS Snowball Edge Storage Optimized devices for offline transfer to Amazon S3; perform parallel local copies and let AWS import the data. Data is encrypted with AWS KMS and devices have end-to-end tracking.

Rationale: Snowball supports tens to hundreds of TB per device and completes in days via shipping. It meets the 10-day deadline, KMS encryption, and tracking requirements; parallel copy mitigates small-file overhead.

B ✗ — Use AWS DataSync over the existing 200 Mb/s internet link to transfer directly to Amazon S3 during off-peak hours.

Rationale: At 200 Mb/s, 600 TB requires months, not days, using $T_{\text{net}} = D/B$. Small files further reduce effective throughput, so it cannot meet 10 days.

C ✗ — Provision a new 1 Gbps AWS Direct Connect and use AWS DataSync to migrate the data to Amazon S3 within the 10-day window.

Rationale: Direct Connect typically has multi-week lead time. Even at 1 Gbps line rate, 600 TB needs roughly 56 days, so it still misses the 10-day requirement.

D ✗ — Request multiple AWS Snowcone devices and run AWS DataSync on them to seed data, then trickle incremental changes over the limited link.

Rationale: Snowcone is 8–14 TB per device and intended for constrained edge sites. Migrating 600 TB would require dozens of units and operational overhead; Snowball is designed for this scale.

Explanation

When network bandwidth or timelines make online transfer impractical, the AWS Snow Family enables secure, offline bulk data movement. AWS Snowball Edge Storage Optimized is designed for tens to hundreds of terabytes per device and petabyte-scale migrations. Devices ship to the site preconfigured; you

perform a high-speed local copy, ship them back, and AWS imports the data into Amazon S3 in your selected Region. Data at rest is encrypted with AWS KMS keys, and devices are tamper-resistant with end-to-end tracking.

A quick feasibility check uses $T_{\text{net}} = D/B$. For 600 TB over 200 Mb/s:

$$T_{\text{net}} = \frac{4.8 \times 10^{15} \text{ bits}}{2.0 \times 10^{8} \text{ bits/s}} \approx 2.4 \times 10^{7} \text{ s} \approx 278 \text{ days}$$

Variables used:

- T_{net}: time to transfer [seconds]
- D: data volume [bits]
- B: effective network throughput [bits/s]

This is far beyond the 10-day requirement, so an offline approach is needed. AWS Snowball completes in days when you account for local copy and shipping, and multiple devices can be used in parallel to meet deadlines. It also satisfies the security requirement via AWS KMS encryption and provides a tamper-resistant, trackable chain of custody. For many small files, planning for parallel, multithreaded copies (and optionally packaging small files) helps maintain throughput during the local copy.

Using AWS DataSync over the existing 200 Mb/s link cannot meet the 10-day window because the theoretical minimum is in the hundreds of days, and small-file overhead typically reduces effective throughput. Standing up a 1 Gbps AWS Direct Connect does not solve the timeline: it has nontrivial provisioning lead time and, even at full utilization, 600 TB would take roughly 56 days. AWS Snowcone, while rugged and able to run DataSync, offers only 8–14 TB per device and is intended for constrained edge sites, making it impractical for 600 TB at a central data center.

Key Concepts

- **Offline bulk transfer with AWS Snowball:** Purpose-built for petabyte-scale migrations when networks or timelines make online transfer impractical; integrates with Amazon S3.
- **Transfer time estimate $T_{\text{net}} = D/B$:** A quick calculation identifies when online transfer cannot meet deadlines.
- **Security features of Snow devices:** Data encrypted with AWS KMS keys, tamper-resistant hardware, and end-to-end tracking to satisfy compliance.
- **Device selection:** Snowball for tens to hundreds of TB per device; Snowcone (8–14 TB) for constrained edge sites, not large data center exits.

Common Pitfalls

- Assuming higher parallelism or off-peak scheduling can overcome a hard bandwidth cap; physics of D/B still apply and small files degrade throughput further.

15.41 — AWS Shield DDoS protections (D2.T2.2)

Domain 2 • Task 2.2 ↑ Back to Question 15.41

A small e-commerce site expects occasional large traffic spikes that could be simple volumetric DDoS events. Which AWS offering automatically provides baseline detection and mitigation for these common network floods without extra enrollment?

Options & Rationales

A ✗ — AWS Shield Advanced — enhanced mitigation with near-real-time reporting and cost protection.

Rationale: Shield Advanced offers stronger protections and additional features, but it requires subscription and is not the baseline automatic offering.

B ✓ — AWS Shield Standard — built-in, always-active protection against typical volumetric DDoS floods.

Rationale: Provides baseline, always-active DDoS detection and automatic mitigation for common volumetric attacks at no additional enrollment cost.

C ✗ — AWS WAF — a managed web application firewall that automatically blocks volumetric network floods.

Rationale: WAF protects at the application layer (HTTP/S) and requires rules; it does not automatically provide baseline volumetric network flood mitigation for all resources.

D ✗ — Security groups — stateful host-level filters that inherently stop large-scale DDoS traffic.

Rationale: Security groups control instance-level traffic and are not designed to detect or mitigate large volumetric DDoS attacks across the network.

Explanation

AWS provides multiple layers of protection against denial-of-service events. The baseline service included for AWS resources offers continuous monitoring and automated response to straightforward, high-volume network floods so customers do not need to subscribe to additional DDoS services for common attacks. A higher-tier subscription delivers advanced analytics, rapid operational support, and financial safeguards for prolonged or sophisticated attacks, but it is optional and requires enrollment. Other controls such as web application firewalls and security group rules protect different layers (application or instance) and usually need explicit configuration; they are not a substitute for the provider's built-in, always-running volumetric mitigation.

Key Concepts

- **Baseline DDoS protection:** Continuous provider-side monitoring and automated mitigation aimed at common high-volume network floods, provided without separate subscription for standard coverage.

- **Advanced DDoS subscription:** A paid option that adds greater visibility, operational response, and cost protections for complex or sustained attacks.
- **Layered defenses:** WAF and security groups address application-layer rules and instance-level filtering respectively, and require customer configuration.

Common Pitfalls

- Confusing application-layer WAF rules with automatic network-level DDoS mitigation leads to choosing WAF as the baseline DDoS answer.
- Assuming instance-level controls like security groups will stop large-scale volumetric floods; they are not designed for provider-scale DDoS mitigation.

Glossary

- **Volumetric attack:** A DDoS that overwhelms bandwidth or network capacity with large traffic volumes.
- **Baseline protection:** Always-on, automatic defensive measures provided by the cloud platform for common threats.

15.42 — Trade-offs among Well-Architected pillars (D1.T1.2)

Domain 1 • Task 1.2

↑ Back to Question 15.42

A web application experiences variable traffic spikes daily. The team wants to maintain fast response times during peaks while avoiding high costs when traffic is low. Which approach best balances performance and cost?

Options & Rationales

A ✗ — Provision a fixed large fleet of instances sized for peak load to ensure headroom.

Rationale: Always-sized-for-peak ensures performance but incurs unnecessary expense during low traffic because unused capacity remains running.

B ✗ — Use only Spot Instances without scaling to minimize cost regardless of availability.

Rationale: Spot instances lower cost but can be interrupted; without scaling or fallbacks, performance may suffer during spikes or interruptions.

C ✗ — Deploy a single very large instance and rely on its capacity to handle all traffic.

Rationale: A single large instance creates a single point of failure and may be expensive compared to scaling out when needed.

D ✓ — Configure Auto Scaling to add and remove instances based on real-time load.

Rationale: Auto Scaling automatically adjusts capacity to match demand, improving responsiveness during peaks and reducing running instances (and cost) when load decreases.

Explanation

Auto Scaling is a mechanism that adjusts compute capacity automatically based on observed demand, which helps maintain application responsiveness during traffic increases while reducing running resources and cost when demand falls. It supports horizontal scaling (adding/removing instances) and can use policies tied to metrics such as CPU or request rates.

Why the correct choice fits. Configuring Auto Scaling aligns capacity with real-time load: more instances launch during peaks to preserve performance, and instances terminate during quiet periods to lower costs. This approach implements a dynamic balance between the performance-efficiency and cost-optimization concerns.

Why the other options are less appropriate. Provisioning a fixed large fleet guarantees headroom but keeps many instances idle during low traffic, increasing costs without improving efficiency. Relying solely on Spot Instances reduces expense but introduces interruption risk; without scaling or fallback instances, interruptions or sudden spikes can harm performance. Using a single very large instance concentrates risk (no redundancy) and often costs more than scaling out only when needed.

Key Concepts

- **Auto Scaling:** Automatically adjusts the number of compute resources to match demand, enabling responsiveness and cost control.
- **Performance vs. Cost trade-off:** Increasing capacity improves response times but raises costs; dynamic scaling aims to optimize both.

Common Pitfalls

- Assuming lowest cost option always meets performance needs; interruptions or insufficient capacity can hurt user experience.
- Thinking a single large resource is both cheapest and most reliable; it creates availability and scalability risks.

Glossary

- **Horizontal scaling:** Adding or removing multiple instances to change total capacity.
- **Spot Instances:** Lower-cost spare capacity that can be reclaimed by the provider, suitable for flexible, interruption-tolerant workloads.

15.43 — VPC purpose: resource isolation (D3.T3.5)

Domain 3 • Task 3.5

↑ Back to Question 15.43

A company runs Amazon EC2 instances in private subnets within a custom Amazon VPC. The instances must download objects from Amazon S3 and call a third-party SaaS API without sending any traffic over the public internet. The company wants all communication to stay within the VPC using private connectivity. Which approach best meets these requirements?

Options & Rationales

A ✗ — Deploy NAT Gateways in each Availability Zone and route private subnets through them to reach S3 and the SaaS API over the internet.

Rationale: NAT Gateways enable outbound connections from private subnets to the internet. This violates the requirement to avoid public internet paths.

B ✗ — Attach an Internet Gateway and assign public IPs to the instances so they can access S3 and the SaaS API directly.

Rationale: An Internet Gateway with public IPs provides direct internet access, exposing resources rather than keeping communication private within the VPC.

C ✓ — Use a Gateway VPC endpoint for Amazon S3 and an AWS PrivateLink interface endpoint to the SaaS; update routes and security groups for private access.

Rationale: Gateway VPC endpoints keep S3 traffic private within the VPC. AWS PrivateLink interface endpoints provide private, VPC-contained access to partner SaaS—no public internet path.

D ✗ — Use a Gateway VPC endpoint for S3 and a NAT Gateway for the SaaS API; keep instances in private subnets.

Rationale: The S3 path is private via the Gateway endpoint, but SaaS traffic still egresses via NAT (internet), failing the "no public internet" requirement.

Explanation

A VPC is isolated by default; connectivity to external services is explicit. Public subnets gain internet access only with an Internet Gateway (IGW). Private subnets can initiate outbound connections using a NAT Gateway, which still sends traffic to the internet even though the instances do not have public IPs. When the goal is to keep traffic to AWS services and partner SaaS off the public internet and contained within the VPC, VPC endpoints are the appropriate choice.

For Amazon S3, a Gateway VPC endpoint provides private connectivity from the VPC to S3 without traversing the public internet. Traffic is routed to the endpoint through route tables associated with the private subnets. For many other AWS services and partner SaaS offerings, AWS PrivateLink provides interface endpoints—elastic network interfaces in your subnets with private IPs—that enable VPC-contained communication to those services.

Therefore, combining a Gateway VPC endpoint for S3 with an AWS PrivateLink interface endpoint for the partner SaaS satisfies the requirement: instances remain in private subnets, and all communication stays within the VPC using private connectivity rather than public internet paths.

Approaches that rely on NAT Gateways or an Internet Gateway route traffic over the internet and do not meet the stated requirement. Using a NAT Gateway from private subnets enables outbound internet access, which conflicts with the "no public internet" constraint. Attaching an Internet Gateway and

assigning public IPs similarly exposes resources to the internet. Mixing a Gateway endpoint for S3 with a NAT Gateway for the SaaS still sends the SaaS API calls over the internet, so it only partially addresses the need.

Key Concepts

- **VPC endpoints:** Use Gateway endpoints for Amazon S3 (and DynamoDB) and AWS PrivateLink interface endpoints for many other services and partner SaaS to keep traffic private in the VPC.
- **NAT Gateway:** Allows private subnets to initiate outbound connections to the internet; instances remain private but egress still uses the public internet path.
- **Internet Gateway:** Enables direct internet access for resources with appropriate routes and public IPs; not suitable when traffic must avoid the public internet.

Common Pitfalls

- Assuming NAT Gateways keep traffic private because instances lack public IPs; NAT still sends egress to the internet.
- Relying on security groups or network ACLs for privacy; they filter traffic but do not create private connectivity to services.

15.44 — Amazon Connect cloud contact center (D3.T3.8)

Domain 3 • Task 3.8 ↑ Back to Question 15.44

A company wants a simple way for customer service staff to answer calls and chats from any location without installing client software. Which AWS service feature best provides an in-browser agent console for handling customer interactions?

Options & Rationales

A ✗ — Amazon Chime SDK — a real-time communication toolkit for building custom audio/video applications.

Rationale: Enables custom real-time features but requires development effort to build an agent console; not an out-of-the-box in-browser agent UI.

B ✗ — AWS Systems Manager Session Manager — a browser-based shell for managing servers remotely.

Rationale: Offers remote shell access to instances, not a tool for handling customer contact center interactions.

C ✗ — Amazon Connect Contact Flows — configurable rules that direct incoming interactions to endpoints.

Rationale: Handles routing logic for contacts but does not itself provide the agent-facing browser console for answering calls or chats.

D ✓ — Amazon Connect Contact Control Panel — a web-based agent console for accepting and managing calls, chats, and tasks.

Rationale: Provides a web console purpose-built for agents to handle voice and chat without local client installs, matching the requirement.

Explanation

The requirement is a ready-to-use agent interface accessible from a web browser so staff can take customer calls and chats without installing software. The Contact Control Panel concept provides an agent-facing web console specifically designed to accept and manage interactions; it includes call controls, chat handling, and task management in the browser.

Other choices are focused on different problems. The real-time communications toolkit enables building audio/video features but needs custom development to produce an agent console. The managed remote shell tool is for server administration and unrelated to customer interactions. Routing configuration governs how contacts are directed to agents or queues but is not the agent user interface itself.

Key Concepts

- **Agent console:** A user interface used by contact-center staff to receive and manage customer interactions from a browser without additional client installs.
- **Contact routing vs. agent UI:** Routing defines where contacts go; the agent UI is what an agent uses to answer and handle those contacts.
- **Managed vs. custom solutions:** Use a managed agent console when you want minimal development and fast agent enablement.

Common Pitfalls

- Confusing routing or communication SDKs with an out-of-the-box agent interface leads to selecting solutions that require significant development work.
- Assuming remote server management tools provide customer-interaction features causes irrelevant choices.

Glossary

- **Agent console:** A web application used by contact-center agents to manage calls and chats.
- **Contact routing:** Rules or workflows that determine how incoming contacts are delivered to agents or queues.

15.45 — SES for transactional/marketing email (D3.T3.8)

Domain 3 • Task 3.8 ↑ Back to Question 15.45

Which Amazon SES feature uses a cryptographic signature added to outgoing messages so recipients can confirm the message was sent by the claimed domain and has not been tampered with?

Options & Rationales

A ✗ — Sender Policy Framework (SPF) DNS record

Rationale: SPF authorizes which mail servers can send for a domain via DNS, but it does not insert a cryptographic signature into each message to prove integrity.

B ✓ — DomainKeys Identified Mail (DKIM) signing

Rationale: DKIM adds a digital signature to message headers. Recipients use the signature and the sender's DNS records to confirm the domain and message integrity.

C ✗ — Dedicated sending IP address

Rationale: A dedicated IP isolates sending reputation for a customer; it does not cryptographically sign messages or prove the domain's authenticity.

D ✗ — Account suppression list

Rationale: A suppression list prevents sending to addresses with past bounces or complaints; it is unrelated to verifying message integrity or sender domain.

Explanation

This question tests awareness of email authentication capabilities and which feature specifically provides a per-message cryptographic signature to validate both sender domain and message integrity.

The key requirement is a mechanism that embeds a verifiable cryptographic signature in outgoing emails so recipients can check both the origin domain and that the content was not altered in transit. DKIM (DomainKeys Identified Mail) fulfills this requirement by signing message headers with a private key; recipients use the corresponding public key published in DNS to validate the signature. This provides message integrity and a link to the sending domain.

Why the other choices are incorrect:

- The option describing an authorization DNS record relates to SPF, which lists permitted sending servers but does not add a per-message cryptographic signature. SPF helps prevent spoofing by validating sending hosts, not message contents.
- Using a dedicated sending IP affects reputation management and deliverability by isolating traffic, but it does not cryptographically prove message integrity or domain ownership on each email.
- A suppression list is a safety mechanism to stop sending to addresses that previously bounced or complained; it manages deliverability and compliance, not message authentication.

Key Concepts

- **DKIM signing:** Attaches a digital signature to email headers so receivers can verify the signature against a public key in DNS, confirming origin and integrity.

- **SPF record:** A DNS policy that declares which mail servers may send mail for a domain; it checks sending hosts, not content signatures.
- **Dedicated IP address:** An isolated IP for sending that helps control reputation and deliverability.
- **Suppression list:** A list of addresses blocked from future sends due to prior bounces or complaints.

Common Pitfalls

- Confusing SPF with DKIM leads to thinking DNS alone signs messages; SPF authorizes servers but does not sign message contents.
- Believing reputation controls (like dedicated IPs) provide cryptographic verification is incorrect; they influence deliverability, not message authentication.

15.46 — GuardDuty detects anomalies (D2.T2.2)

Domain 2 • Task 2.2 ↑ Back to Question 15.46

A security analyst notices several management API calls from an IAM role at unusual hours. Which Amazon GuardDuty capability most directly helps detect this as abnormal by comparing activity to each resource's normal usage patterns?

Options & Rationales

A ✗ — Matching API calls against a curated blocklist of known malicious IP addresses.

Rationale: Blocklist matching helps detect traffic involving known bad actors but does not identify unusual activity that deviates from an account's normal patterns.

B ✗ — Relying on static signature rules like traditional antivirus to identify threats.

Rationale: Signature-based detection looks for known fingerprints and is less effective at finding novel behavior changes compared with learning-based profiling.

C ✗ — Scanning Amazon S3 buckets for public objects to find exposed data.

Rationale: S3 exposure scans detect publicly accessible data but do not compare runtime API usage patterns to detect anomalous management actions.

D ✓ — Using learned normal activity profiles (behavioral baselining) to flag deviations.

Rationale: GuardDuty creates baseline profiles of expected activity per resource and flags unusual deviations, helping detect atypical API behavior without prior signature rules.

Explanation

Behavioral baselining is the process of observing and recording normal activity patterns for resources (such as API call frequency, typical source IPs, or time-of-day usage) and then identifying actions that fall outside those learned norms. In this scenario, unusual management API calls at odd hours are best detected by a system that has established what 'normal' looks like for that IAM role or account and can flag deviations.

Why the correct choice fits. The answer that describes learned normal activity profiles explains a detection method that models expected behavior per resource and raises findings when activity diverges. This approach catches unexpected or subtle changes even when no prior signature or threat list exists.

Why the other choices are less appropriate. The option describing matching against a curated list of malicious IPs detects communications involving known bad endpoints but won't catch an otherwise unknown pattern of API use originating from legitimate infrastructure. The static signature-based approach relies on known fingerprints and misses new or subtle deviations in legitimate-looking API calls. The S3 exposure scan addresses data visibility risks, not runtime anomalies in management API usage.

Key Concepts

- **Behavioral baselining:** Establishes typical patterns for entities (users, roles, resources) and identifies deviations that may indicate compromise or misuse.
- **Threat intelligence lists:** Collections of known malicious identifiers (IP addresses, domains) used to detect interactions with known bad actors but not novel behavior.
- **Signature-based detection:** Identifies threats by matching predefined patterns; effective for known threats but limited for new or anomalous activity.

Common Pitfalls

- Assuming an IP blocklist will find all problems; many attacks use previously unseen infrastructure.
- Confusing data-exposure checks (like public S3 scans) with behavioral anomaly detection; they address different risk types.

Glossary

- **API call:** A request to an AWS service's management interface to perform actions (create, modify, delete resources).
- **Baselining:** Recording normal operational metrics over time to enable future anomaly detection.

15.47 — AWS security & compliance add value (D1.T1.1)

Domain 1 • Task 1.1 ↑ Back to Question 15.47

A regulated enterprise is moving to a multi-account strategy using AWS

Organizations. The security team must ensure that across all current and future accounts:

- AWS CloudTrail cannot be stopped or deleted
- Amazon S3 Block Public Access cannot be disabled at the account or bucket level

They want a centralized, preventive control with minimal ongoing operations. Which is the BEST solution?

Options & Rationales

A ✗ — Attach IAM permission boundaries to developer roles to block CloudTrail and S3 public access changes across accounts.

Rationale: Permission boundaries apply only to IAM principals they're attached to and do not restrict the root user. They are not an organization-wide preventive control.

B ✗ — Deploy AWS Config rules and conformance packs to auto-remediate when CloudTrail or S3 Block Public Access is changed.

Rationale: AWS Config is primarily detective. Auto-remediation is reactive and allows a window of noncompliance, so it does not meet the "cannot disable" preventive requirement.

C ✗ — Create Amazon CloudWatch alarms for CloudTrail and S3 configuration changes to notify security for manual remediation.

Rationale: CloudWatch provides detection and alerting only. It does not prevent the changes from occurring and requires manual response.

D ✓ — Use AWS Organizations SCPs to deny stopping/deleting CloudTrail and changing S3 Block Public Access, attached at the org root or OUs for all accounts.

Rationale: Service control policies provide preventive, centralized guardrails across accounts and apply to all principals, including the root user. This directly blocks the disallowed actions.

Explanation

The requirement is for preventive, centralized guardrails across many accounts that ensure critical controls (CloudTrail logging and S3 Block Public Access) cannot be disabled. In AWS, Service Control Policies (SCPs) in AWS Organizations are the governance mechanism that filter the maximum available permissions for accounts in an organization or OU. SCPs apply to all principals in member accounts, including the root user, and can explicitly deny sensitive actions such as stopping or deleting CloudTrail and modifying S3 Block Public Access. This satisfies the "cannot" and "centralized" requirements with minimal operational overhead after deployment.

IAM permission boundaries limit what specific IAM principals can do, but they do not affect the root user and must be attached per principal, which is neither

comprehensive nor centralized. AWS Config rules and conformance packs are primarily detective; even with auto-remediation, there is a period where the control is disabled, so they do not meet the strict preventive requirement. Amazon CloudWatch alarms likewise only detect and alert, relying on humans or automation to respond, which is not preventive.

Key Concepts

- **AWS Organizations SCP:** Organization-wide guardrails that define the maximum permissions an account can use; can explicitly deny sensitive actions and apply to all principals, including root.
- **Preventive vs. Detective Controls:** Preventive controls stop undesired actions from occurring; detective controls identify issues after they occur and may trigger remediation.
- **Least Privilege at Scale:** Centralized guardrails reduce attack surface and governance effort across many accounts by denying risky actions globally.

Common Pitfalls

- Assuming IAM permission boundaries can restrict the root user or act as an org-wide control; they cannot.
- Believing AWS Config auto-remediation guarantees "cannot disable" requirements; it is reactive and allows a window of noncompliance.

Glossary

- **Service Control Policy (SCP):** A policy type in AWS Organizations that sets permission guardrails for member accounts.
- **Permission Boundary:** An IAM feature that sets the maximum permissions a specific IAM principal can have; does not apply to root.

15.48 — IAM users vs groups vs roles (D2.T2.3)

Domain 2 • Task 2.3 ↑ Back to Question 15.48

A company wants to minimize long-term credentials and apply least privilege.

- An application running on Amazon EC2 must read objects from a private Amazon S3 bucket.
- External contractors authenticate with the company's identity provider and need short-term AWS Management Console access to selected services for a 3-month project.

Which TWO actions meet these goals using AWS best practices?

Options & Rationales

A ✗ — Create an IAM user for the EC2 application with access keys; store the keys on the instance and rotate them every 90 days.

Rationale: Using long-term access keys for applications is a known pitfall; prefer roles with temporary credentials via an instance profile.

B ✓ — Attach an IAM role to the EC2 instance using an instance profile that

allows s3:GetObject; the application uses temporary credentials issued by AWS STS.

Rationale: EC2 should use an instance profile role for temporary credentials, avoiding embedded access keys while granting only required S3 actions.

C ✓ — Set up workforce federation so contractors authenticate with the external IdP and assume IAM roles via AWS STS; attach permission policies to those roles.

Rationale: Federation issues temporary credentials and lets contractors assume roles defined by trust and permission policies, avoiding long-term IAM users.

D ✗ — Create individual IAM users for contractors with console passwords and add them to an IAM group that has the required permissions.

Rationale: This introduces long-term credentials and extra lifecycle overhead. With an external IdP, workforce federation to roles is the best practice.

E ✗ — Provide contractors the IAM role ARN and have them sign in directly as the role using the AWS sign-in URL; no separate user is required.

Rationale: Roles cannot be logged into directly. A trusted, authenticated principal must assume the role per its trust policy.

F ✗ — Create an IAM group for the EC2 instance and attach policies so the instance can authenticate as the group to access S3.

Rationale: Groups are not principals and cannot authenticate. EC2 should use an instance profile role for access to S3.

Explanation

Foundational IAM identities behave differently and are used for distinct purposes. An IAM user represents a long-term identity (typically a person) that can have a console password and optional access keys. An IAM group is only a container to manage permissions for multiple users; it is not a principal and cannot authenticate. An IAM role is a principal that does not have long-term credentials; instead, it is assumed by a trusted principal and provides temporary security credentials through AWS Security Token Service (AWS STS). Roles contain a trust policy (who can assume the role) and permission policies (what the role can do once assumed).

For applications on Amazon EC2, best practice is to avoid embedding access keys by using an instance profile role. The EC2 service presents the role to the instance so code retrieves temporary credentials automatically and only the scoped permissions apply. For external contractors who already authenticate to the company's identity provider, workforce federation should be used so they assume roles, gaining temporary credentials governed by the role's trust and permission policies. This minimizes long-term credentials and simplifies permission management under least privilege.

Attaching an IAM role to the EC2 instance via an instance profile aligns with these practices, providing temporary credentials for s3:GetObject without

storing keys on the instance. Configuring federation so contractors assume IAM roles also aligns, issuing temporary credentials tied to role policies and avoiding IAM users per contractor. Creating IAM users with access keys for the EC2 app is a documented pitfall that increases credential leakage risk and operational overhead. Creating IAM users for contractors and placing them in groups similarly introduces long-term credentials and administrative burden, despite being functionally possible. Instructing contractors to "sign in as a role" fails because roles cannot be logged into directly; they must be assumed by an authenticated principal permitted by the trust policy. Attempting to have an EC2 instance "authenticate as a group" is impossible because groups are not principals.

Key Concepts

- **IAM user vs role:** Users have long-term credentials; roles are assumed and provide temporary credentials via AWS STS.
- **IAM group:** A permissions container for users; not a principal and cannot authenticate.
- **Instance profile role (EC2):** Assigns a role to an EC2 instance so applications obtain temporary credentials without storing access keys.
- **Federation to roles:** External identities authenticate with an IdP and assume roles, enforcing least privilege with time-limited access.

Common Pitfalls

- Issuing long-term access keys to applications instead of using roles with temporary credentials.
- Trying to log in directly as a role or treating groups as authenticating identities, both of which are not supported.

15.49 — Multi-Region choice for low latency (D1.T1.1)

Domain 1 • Task 1.1 ↑ Back to Question 15.49

Which AWS service caches objects at edge locations to reduce origin latency for read-heavy global content when compute runs in a single Region?

Options & Rationales

A ✗ — Amazon Route 53 latency-based routing

Rationale: Directs users to the optimal Region based on latency but does not cache content at edge locations.

B ✓ — Amazon CloudFront

Rationale: A content delivery network that caches objects at edge locations, offloading origin latency and improving performance for read-heavy global content.

C ✗ — AWS Global Accelerator

Rationale: Directs users to the optimal application endpoint for performance, but it is not a caching service.

D ✗ — Amazon S3 Cross-Region Replication

Rationale: Replicates S3 objects between Regions for multi-Region data access; it does not provide edge caching.

Explanation

User-perceived latency is dominated by network distance and routing. Two common ways to improve performance for global users are: placing endpoints closer to users and caching. When workloads run in a single Region but must serve read-heavy global traffic, caching content near users is the most effective tactic. Amazon CloudFront is the AWS content delivery network that caches objects at edge locations worldwide, reducing round-trip time to the origin and improving page load times and API responsiveness. This directly supports the Performance Efficiency pillar and can also enhance Reliability by offloading origin load.

Amazon Route 53 latency-based routing and AWS Global Accelerator both direct users to an optimal Regional endpoint, which helps when you deploy endpoints in multiple Regions. However, neither service caches content. Amazon S3 Cross-Region Replication enables multi-Region copies of S3 objects, which can support low-latency reads from another Region, but it is replication between Regions—not edge caching—and is not a CDN.

Therefore, the only option that specifically caches objects at edge locations to reduce origin latency is Amazon CloudFront.

Key Concepts

- **Edge caching with Amazon CloudFront:** Caches objects at edge locations to reduce round-trip time to the origin and improve performance for global users.
- **Multi-Region routing vs. caching:** Routing (Route 53 latency-based routing, AWS Global Accelerator) sends users to the nearest healthy Regional endpoint; caching serves content from edge points of presence.
- **Replication vs. caching:** S3 Cross-Region Replication creates additional Regional copies for data proximity; it does not provide edge delivery.
- **Performance Efficiency pillar:** Placing data or cached copies closer to users reduces latency and variability for faster responses.

Common Pitfalls

- Confusing routing services (Route 53 latency-based routing or AWS Global Accelerator) with a CDN; routing does not cache content.
- Assuming S3 replication is equivalent to edge caching; replication is Regional, not at edge locations.

Glossary

- **Edge location:** A globally distributed site where a CDN like CloudFront serves cached content close to users.

- **Origin:** The backend source (for example, an application or S3 bucket) from which the CDN fetches content.

15.50 — Detective vs preventive vs corrective (D2.T2.2)

Domain 2 • Task 2.2 ↑ Back to Question 15.50

Which type of security control is primarily responsible for recording user and API activity so administrators can spot unusual behavior and perform audits (for example, using AWS CloudTrail)?

Options & Rationales

A ✗ — Preventive control — blocks access and enforces strict configuration to stop attacks.

Rationale: Preventive controls aim to stop incidents before they occur by limiting access and securing settings, rather than logging activity for later review.

B ✗ — Corrective control — removes threats and returns systems to a trusted state.

Rationale: Corrective controls act after an incident to remediate issues and restore services; they do not primarily provide detection or logging.

C ✓ — Detective control — captures and logs activity to identify anomalous behavior.

Rationale: Detective controls focus on monitoring and recording events (such as API calls) so that suspicious actions can be discovered and audited; CloudTrail is an example.

D ✗ — Compensating control — replaces technical controls with non-technical processes temporarily.

Rationale: Compensating controls are alternative measures used when standard controls cannot be implemented; they are not the main mechanism for activity logging.

Explanation

Detective controls are designed to observe, record, and alert on events so security teams gain visibility into what occurred. They enable analysts to spot deviations from normal patterns and support forensic review and compliance audits. AWS CloudTrail is a practical example that records API calls and related actions to create an audit trail.

Preventive controls differ because they aim to stop a threat before it affects systems by restricting access or hardening settings. Corrective controls operate after detection: they remove malicious artifacts, fix misconfigurations, and restore systems to an approved state. Compensating controls are temporary or alternative arrangements used when standard controls cannot be applied.

Why the correct choice is right and others are wrong. The correct option

describes a control whose main purpose is to capture and log actions for audit and anomaly detection; this is the core function of detective controls. The preventive option describes measures that try to prevent incidents and therefore are not primarily about recording activity. The corrective option concerns remediation and recovery, not monitoring. The compensating option refers to substitute controls, which may include procedures but are not the standard mechanism for capturing user and API activity.

Key Concepts

- **Detective control:** Records and highlights suspicious activity to provide visibility and evidence for investigation.
- **Preventive control**: Designed to block unauthorized actions and reduce the chance of incidents occurring.
- **Corrective control:** Restores and repairs systems after a security event to recover confidentiality, integrity, and availability.

Common Pitfalls

- Confusing logging with prevention leads to thinking recorded events stop attacks; logs only help find and investigate incidents. Another mistake is assuming corrective actions detect issues rather than remediate them after detection.

15.51 — Use Well-Architected Tool for reviews (D1.T1.2)

Domain 1 • Task 1.2 ↑ Back to Question 15.51

A startup reviewed its three-tier web application in the AWS Well-Architected Tool. Leadership set a measurable reliability target of RTO = 15 minutes for the database tier and wants clear evidence of posture improvement as recommendations are implemented. Which actions should the team prioritize to directly connect these goals to architecture decisions and show progress over time?

Options & Rationales

A ✗ — Adopt Savings Plans for steady EC2 usage to make costs predictable and demonstrate reliability improvements.

Rationale: Savings Plans address cost optimization, not database RTO or evidence of risk reduction. They do not prove reliability improvements on their own.

B ✗ — Enable S3 Block Public Access and SSE-KMS to meet the database RTO requirement.

Rationale: These controls reduce S3 security risk and protect data, but they do not address database recovery time objectives.

C ✗ — Configure AWS Backup to take manual weekly snapshots of the database.

Rationale: The Tool recommends automated backups. Weekly manual snapshots do not align with a 15-minute RTO and leave long recovery gaps.

D ✗ — Add Amazon CloudWatch CPU alarms on the database to meet the 15-minute RTO.

Rationale: Alarms and runbooks aid operational excellence but do not change the database architecture needed to satisfy a strict RTO.

E ✗ — Place the web tier in an Auto Scaling group behind an Application Load Balancer with health checks to achieve the 15-minute database RTO.

Rationale: Health-checked load balancing improves the web/app tier, not the database tier's recovery objective.

F ✓ — Enable Amazon RDS Multi-AZ for the database to improve reliability toward the 15-minute RTO target.

Rationale: The Well-Architected guidance calls out Multi-AZ databases to improve reliability. This architectural change directly supports meeting a defined RTO.

G ✓ — Create a Well-Architected Tool milestone after implementing changes to snapshot the architecture and compare risk reduction over time.

Rationale: Milestones in the Tool capture point-in-time states so you can track and demonstrate posture improvements as recommendations are applied.

Explanation

The AWS Well-Architected Tool (WAT) is designed to connect business goals to architecture decisions and to provide a measurable path to improvement. Two capabilities from the guidance are central here: using reliability patterns such as Multi-AZ databases, and using milestones to snapshot the architecture and compare risk over time. With a defined database Recovery Time Objective (RTO), the most direct architectural action is to make the database resilient per the reliability pillar; for relational databases, the Tool highlights Multi-AZ as a key improvement. To demonstrate that changes are working and risks are trending down, milestones in WAT capture point-in-time states so teams can show before/after posture.

Savings Plans, while valuable for cost optimization and sustainability discussions, do not improve database recovery characteristics or produce evidence of risk reduction by themselves. AWS Backup is recommended in an automated fashion; weekly manual snapshots create long gaps that are incompatible with a 15-minute RTO. Security controls for Amazon S3—such as Block Public Access and SSE-KMS—address the security pillar for S3 data but are unrelated to a database recovery target. CloudWatch alarms and runbooks help operational excellence and observability but do not substitute for the underlying reliability architecture needed to satisfy a strict RTO. Similarly, health-checked load balancing and Auto Scaling improve the stateless tiers but do not change recovery characteristics of the database tier.

Key Concepts

- **Well-Architected Tool milestones:** Snapshot workload state to compare risks and measure posture changes after implementing recommendations.
- **Reliability pillar pattern (Multi-AZ databases):** Improves availability and recovery characteristics for the database tier to align with RTO targets.
- **Measurable targets:** Defining RTO links business goals to concrete design choices and trade-offs.
- **Scope-to-action mapping:** Findings become specific improvements (for example, Multi-AZ, backups) tied to pillars and effort.

Common Pitfalls

- Confusing cost optimizations (Savings Plans) or security controls (S3 SSE-KMS) with actions that satisfy a database RTO.
- Relying on alarms or manual weekly backups to meet a tight RTO; these do not replace resilient architecture patterns.

15.52 — RDS Multi-AZ for high availability (D3.T3.4)

Domain 3 • Task 3.4 ↑ Back to Question 15.52

An application uses Amazon RDS and must remain available with minimal downtime during an Availability Zone disruption. The team is planning their approach. Select TWO actions they should AVOID.

Options & Rationales

A ✗ — Running periodic failover drills to validate application behavior with the Multi-AZ standby

Rationale: Testing failover is a sound practice to confirm Multi-AZ behavior and application reconnection during events.

B ✗ — Enabling Multi-AZ so a synchronized standby exists in another Availability Zone with automatic failover

Rationale: This aligns with Multi-AZ design: a synchronized standby in a separate AZ and automatic failover minimize downtime during AZ issues.

C ✗ — Monitoring RDS events and metrics to confirm when a Multi-AZ failover occurs

Rationale: Observability complements Multi-AZ by confirming failover events and helping operations respond appropriately.

D ✗ — Planning for automatic failover to the standby during an Availability Zone disruption

Rationale: Expecting automatic failover is correct; Multi-AZ is designed to redirect to the standby when the primary becomes unavailable.

E ✓ — Placing the primary and its standby in the same Availability Zone to reduce inter-AZ costs

Rationale: This violates AZ isolation. Multi-AZ high availability relies on a standby in a different Availability Zone; co-locating both removes resilience to an AZ failure.

F ✓ — Relying on asynchronous read replicas instead of Multi-AZ for availability and failover

Rationale: This is an anti-pattern. Multi-AZ uses synchronous replication with automatic failover; read replicas are asynchronous and do not provide automatic failover for HA.

Explanation

Multi-AZ in Amazon RDS is a high-availability feature that maintains a second, synchronized database instance (standby) in a different Availability Zone. Because data is synchronously replicated, the standby is kept up to date, and RDS can perform automatic failover to the standby if the primary or its AZ becomes unavailable, reducing downtime for the application. The essence is AZ isolation, synchronous replication, and automatic failover.

Actions that align with this model include enabling Multi-AZ so there is a synchronized standby in another AZ, planning for automatic failover behavior during an AZ disruption, running periodic failover drills to validate application reconnect logic, and monitoring RDS events/metrics to understand when failovers occur. These support the intended HA capability.

By contrast, placing the primary and standby in the same Availability Zone undermines fault isolation. Multi-AZ's resilience assumes the instances are in different AZs; co-locating them means an AZ outage can impact both, defeating the goal of minimizing downtime. Similarly, substituting asynchronous read replicas for Multi-AZ is a misuse: read replicas are not kept in lockstep and do not provide automatic failover. They serve read scaling, not HA. Therefore, both co-locating primary and standby in one AZ and relying on asynchronous read replicas for availability are anti-patterns that contradict the Multi-AZ high-availability design.

Key Concepts

- **Multi-AZ separation:** The standby resides in a different Availability Zone to survive an AZ failure.
- **Synchronous replication:** Keeps the standby current so failover minimizes data loss and downtime.
- **Automatic failover:** RDS promotes the standby when the primary or its AZ is impaired.

Common Pitfalls

- Confusing read replicas with Multi-AZ and assuming they provide automatic failover and HA—they do not.

15.53 — Infra vs managed service responsibilities (D2.T2.1)

Domain 2 • Task 2.1

↑ Back to Question 15.53

A company must decide who is responsible for operating system patching for two deployment options: a virtual server on Amazon EC2 and a managed database on Amazon RDS. Which statement best describes the division of responsibility for OS patching?

Options & Rationales

A ✗ — The customer is responsible for OS patching on both Amazon EC2 and Amazon RDS instances.

Rationale: This is incorrect because managed database services include platform maintenance by AWS; customers do not perform OS patching for the managed RDS infrastructure.

B ✗ — AWS patches the OS for EC2 instances, but the customer patches the OS for Amazon RDS.

Rationale: This reverses the responsibilities: EC2 gives customers control of the guest OS, and RDS is the managed offering where AWS performs OS-level maintenance.

C ✓ — AWS applies operating system patching for Amazon RDS, while the customer is responsible for patching the OS on Amazon EC2 instances.

Rationale: Amazon RDS is a managed database service where AWS handles engine and underlying OS maintenance tasks, including OS patching. EC2 provides virtual servers, so the customer manages the guest OS and must apply patches.

D ✗ — Neither AWS nor the customer patches the OS; patching is automatic and not a shared responsibility.

Rationale: This is incorrect because patching responsibilities depend on the service model. For unmanaged virtual servers the customer patches, while for managed database services AWS handles platform patching.

Explanation

When comparing an unmanaged virtual server offering to a managed database service, the split of operational duties is determined by who controls the guest operating system. With virtual servers, the customer installs, configures, and maintains the OS and applications. With managed database services, AWS operates the database engine and the platform beneath it, including routine OS maintenance tasks like patching.

Why the correct statement is right. It matches the service model distinctions: the virtual server requires customer-managed OS lifecycle activities, while the managed database delegates platform maintenance to AWS, so the provider applies OS patches for the managed database environment.

Why the other statements are wrong. Saying the customer patches both misunderstands that managed services include platform maintenance by the provider. Stating AWS patches EC2 but not RDS reverses the actual boundaries. Claiming no one patches ignores that responsibility exists and is allocated by service type.

Key Concepts

- **Shared responsibility model:** Responsibilities for security and operations are divided between the cloud provider and the customer based on the managed level of the service; more managed services shift routine platform tasks to the provider.
- **Virtual server responsibility boundary:** For virtual machines, the customer controls the guest OS and must perform OS patching, updates, and configuration.
- **Managed service responsibility boundary:** For managed services, the provider handles the underlying platform and its maintenance, which typically includes OS patching for the managed components.

Common Pitfalls

- Confusing service names with levels of management; assuming all AWS services are fully managed leads to incorrect ownership assumptions.
- Assuming automation means no one ever patches; automation changes how patches are applied but does not remove responsibility allocation.

Glossary

- **Guest OS:** The operating system running inside a virtual machine that the tenant controls.
- **Managed service:** A cloud offering where the provider operates and maintains underlying infrastructure and platform components on behalf of the customer.

15.54 — Migration Evaluator for business case (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 15.54

In a migration business case built with AWS Migration Evaluator, which formula calculates the payback period P in years given the one-time migration cost and annual savings?

Options & Rationales

A ✓ — $P = C_{\text{migrate}}/S$

Rationale: Payback is the one-time migration cost divided by annual savings, yielding years to recover the investment.

B ✗ — $P = S/C_{\text{migrate}}$

Rationale: This is the inverse of the payback formula and does not represent years to recoup the migration cost.

C ✗ — $P = C_{\text{on-prem, annual}} - C_{\text{AWS, annual}}$

Rationale: This expression computes annual savings S, not the payback period.

D ✗ — $P = C_{\text{AWS, annual}} / C_{\text{on-prem, annual}}$

Rationale: This is a ratio of annual run rates and ignores the one-time migration cost; it is not a payback calculation.

Explanation

AWS Migration Evaluator builds a data-driven business case by ingesting server inventory and time-series utilization across CPU, memory, storage, and licensing. Using this telemetry, it rightsizes targets to Amazon EC2 (including Graviton-based instances), selects Amazon EBS gp3, and, when modernization is in scope, maps databases to Amazon RDS or Amazon Aurora. It then models projected AWS costs under pricing strategies such as On-Demand, EC2 Spot Instances for flexible workloads, and Compute Savings Plans for steady usage, and can reflect Multi-AZ resilience and regional pricing. The Business Case report compares current annual run-rate to the projected AWS spend, highlights idle capacity elimination and license optimization, and includes one-time migration effort so you can quantify savings and payback.

Annual savings are calculated as the difference between current on-premises costs and projected AWS costs. Payback is defined as how many years it takes for those annual savings to recover the one-time migration cost.

$$S = C_{\text{on-prem, annual}} - C_{\text{AWS, annual}}$$

$$P = \frac{C_{\text{migrate}}}{S}$$

Variables used:

- S: annual savings ($/year)
- $C_{\text{on-prem, annual}}$: current annual cost ($/year)
- $C_{\text{AWS, annual}}$: projected annual AWS cost ($/year)
- C_{migrate}: one-time migration and training cost ($)

The correct expression divides the one-time migration cost by annual savings to yield years. The expression that subtracts annual costs gives savings only. The inverted ratio S/C_{migrate} and the annual cost ratio ignore the definition of payback and do not incorporate the one-time cost correctly.

Key Concepts

- **Payback period:** Time to recover a one-time cost; computed as C_{migrate}/S.
- **Annual savings:** Difference between current and projected annual run-rate, $C_{\text{on-prem, annual}} - C_{\text{AWS, annual}}$.
- **Rightsizing impact:** Accurate telemetry-driven rightsizing and pricing strategy selection determine realistic S.

Common Pitfalls

- Using list-price On-Demand against peak-sized servers or ignoring licensing and elasticity inflates projected AWS costs and understates *S*. Run data collection long enough to capture peaks and seasonality.

15.55 — Tagging & cost allocation for showback (D1.T1.4)

Domain 1 • Task 1.4 ↑ Back to Question 15.55

Which two statements describe how resource tags help implement internal showback or chargeback reporting?

Options & Rationales

A ✓ — Tags let teams attach owner and environment labels to resources so costs can be reported back to the responsible group.

Rationale: Correct — applying owner and environment tags makes it possible to break down spend by team or purpose for internal reports.

B ✗ — Tagging enforces encryption of stored data, ensuring billed storage is always encrypted before allocation.

Rationale: Incorrect — tags are metadata and do not enforce encryption; encryption is configured separately through storage or key management settings.

C ✗ — Tags replace the need for cost reports by deleting untagged resources automatically to avoid unallocated costs.

Rationale: Incorrect — tags do not remove resources; cost reports are still needed and automated clean-up is a separate operational action.

D ✗ — Tags are only useful for access control and cannot be used to group cost data for internal reporting.

Rationale: Incorrect — while tags can help with access filtering, they are commonly used to categorize and group cost data for reporting.

E ✗ — Applying tags directly to billing invoices is required for chargeback to work because invoices accept only tagged line items.

Rationale: Incorrect — invoices do not accept tags directly; billing systems use tag-based reports to attribute costs, not tags embedded in invoices.

F ✓ — Consistent tagging enables automated grouping of resources so finance can allocate spending to business units for internal billing.

Rationale: Correct — uniform tag keys and values allow tools to aggregate costs per business unit, supporting internal chargeback or showback processes.

Explanation

Tags are key-value labels added to cloud resources to identify attributes such as owner, project, environment, or cost center. When tagging is applied consistently, reporting tools can group usage and expense records by those

labels so teams see which units generated which costs. This capability is the foundation for internal showback (informing owners of their consumption) and chargeback (allocating internal bills to owners).

Why the correct statements are right. Stating that tags let teams attach owner and environment labels explains the basic mechanism: metadata identifies the responsible party and workload purpose, enabling per-team or per-environment cost views. Stating that consistent tagging enables automated grouping emphasizes that uniform tag keys and values are required so cost data can be aggregated reliably and assigned to business units.

Why the incorrect statements are wrong. The statement that tags enforce encryption confuses metadata with configuration — tags do not change resource settings like encryption; those are controlled by service or key-management configuration. Saying tags replace cost reports or that invoices accept tagged line items misstates how billing works: tags are used by reporting tools to map costs, but invoices are produced from billing systems and are not themselves tagged. Claiming tags are only useful for access control ignores their common use for categorization and cost reporting.

Key Concepts

- **Resource tag:** A label consisting of a key and value attached to a cloud resource to record metadata such as owner or project. Tags are used for organization, filtering, and reporting.
- **Consistent tagging:** Using standard tag keys and controlled value formats so tools can reliably group and aggregate resources across accounts and services.
- **Showback vs. chargeback:** Showback provides visibility into who consumed resources without billing them; chargeback assigns costs to internal units. Both rely on accurate categorization of resources.

Common Pitfalls

- Assuming tags automatically enforce configuration or security controls leads to incorrect operational designs; tags are metadata only.
- Inconsistent tag naming or missing tags causes inaccuracies in internal cost reports and misattribution of spend.

Glossary

- **Tag key/value:** The pair used to label a resource, for example, Key=Owner, Value=AnalyticsTeam.
- **Internal billing:** The practice of reporting or charging cloud costs within an organization using aggregated usage data and labels.

15.56 — Console vs CLI vs SDKs access (D3.T3.1)

Domain 3 • Task 3.1 ↑ Back to Question 15.56

Which statement correctly describes how the AWS Management Console, the AWS CLI, and AWS SDKs relate when interacting with AWS services?

Options & Rationales

A ✗ — The Management Console uses a different authentication system than the CLI and SDKs, so permissions must be configured separately.

Rationale: This is incorrect because the console, CLI, and SDKs use the same IAM-based account and permissions model, not separate authentication systems.

B ✗ — The AWS CLI is only for manual interactive use and cannot be used for scripted automation or batch operations.

Rationale: This is false; the CLI is specifically designed to support scripting and automation as well as interactive commands.

C ✓ — They all send requests to the same AWS service APIs and rely on IAM for authentication and authorization.

Rationale: All three interfaces invoke the same backend AWS APIs; access control and credentials are enforced by IAM policies and mechanisms.

D ✗ — SDKs bypass IAM and authenticate directly to services using language-level credentials embedded in application code.

Rationale: SDKs use the AWS credentials and IAM for authentication and authorization; embedding credentials is a risk but does not replace IAM enforcement.

Explanation

All common AWS interfaces — browser-based management, command-line tools, and language SDKs — perform the same fundamental action: they issue requests to AWS service APIs. Identity and access control for those requests are governed by AWS Identity and Access Management (IAM). IAM determines who can call which APIs and what actions they may perform, regardless of whether the call originates from the console, a scripted CLI command, or an application using an SDK.

Key Concepts

- **Unified API surface:** AWS service functionality is exposed through service APIs; different clients (console, CLI, SDKs) are simply alternative ways to call those APIs.
- **IAM-based access control:** Authentication (proving identity) and authorization (permitting actions) are enforced by IAM policies and credentials rather than by the client interface.
- **Automation capability of CLI and SDKs:** The command-line tool and SDKs are suitable for automation and integration; they are not limited to interactive use.

Why the correct statement is right. The correct option states that all three methods target the same AWS APIs and that IAM enforces access. This reflects how AWS centralizes service access and permissions: the console issues API calls behind the scenes, the CLI sends API requests directly, and SDKs provide

language-native wrappers around API operations. IAM credentials or roles are used in every case to authenticate callers and evaluate permissions.

Why the other statements are wrong. The idea that the console uses a different authentication system is incorrect because all interfaces use IAM identities or federated credentials. Saying the CLI cannot be scripted misunderstands its design; scripting is a primary use case. Claiming SDKs bypass IAM is incorrect because SDKs rely on IAM credentials or roles to obtain permission to perform actions; embedding credentials in code is a poor practice but does not replace IAM enforcement.

Common Pitfalls

- Assuming different clients have separate permission models instead of a single IAM-based model.
- Believing the CLI is only for manual use; it is commonly used for automation and CI/CD.

Glossary

- **IAM:** AWS Identity and Access Management, the service that manages authentication and authorization for AWS resources.
- **SDK:** Software Development Kit, a language-specific library that wraps AWS service APIs for applications.

15.57 — Firewall Manager centralizes WAF rules (D2.T2.4)

Domain 2 • Task 2.4 ↑ Back to Question 15.57

Which capability best describes AWS Firewall Manager's primary purpose within an organization?

Options & Rationales

A ✗ — It provides per-instance host-based intrusion detection that inspects file system activity on EC2 instances.

Rationale: Host-based intrusion detection is not the role of Firewall Manager; it focuses on managing network and application-layer policy across accounts rather than inspecting instance file activity.

B ✗ — It serves as a managed DNS service to route user traffic to the nearest edge location.

Rationale: DNS routing and edge-location traffic management are functions of services like Amazon Route 53 and CloudFront, not Firewall Manager.

C ✓ — It centrally deploys and enforces uniform network and application protection policies across multiple accounts.

Rationale: Firewall Manager provides centralized creation and enforcement of security policies so administrators can apply consistent firewall and WAF rules across accounts and resources.

D ✗ — It is a centralized key management system for creating and rotating encryption keys across services.

Rationale: Key lifecycle and rotation are handled by AWS KMS; Firewall Manager manages firewall and WAF policies, not encryption key management.

Explanation

AWS Firewall Manager is designed to let administrators define security rules at a single place and have those rules applied across many AWS accounts and resources. This simplifies maintaining consistent protections, such as web application firewall rules or network-layer controls, across an organization. The service monitors for compliance with the defined policies and helps with remediation reporting so account-level deviations can be addressed.

Why the correct choice is right and others are wrong. The correct description captures Firewall Manager's role in applying and enforcing network and application protection policies across accounts, plus its compliance visibility. The option describing host-based intrusion detection is incorrect because that involves inspecting activity on individual instances, which is outside Firewall Manager's scope. The DNS routing option is incorrect because DNS and edge routing are provided by other services. The key management option is incorrect because encryption key creation and rotation belong to the key management service.

Key Concepts

- **Centralized policy management:** Define security rules once and propagate them to multiple accounts and supported resources to ensure uniform protection.
- **Enforcement and compliance reporting:** Continuously check whether resources meet the policy and surface noncompliant resources for remediation.

Common Pitfalls

- Confusing Firewall Manager with host-level or instance-based security products leads to selecting tools that do not provide organization-wide policy distribution.
- Assuming Firewall Manager manages unrelated functions like DNS routing or key rotation causes misaligned architecture choices.

15.58 — Org change management for adoption (D1.T1.3)

Domain 1 • Task 1.3 ↑ Back to Question 15.58

A company begins migrating workloads to AWS and wants to make sure teams can operate and support the new cloud environment. Which approach best focuses on developing the needed staff capabilities during adoption?

Options & Rationales

A ✓ — Create targeted, role-based training and assess skills gaps before each migration phase.

Rationale: Role-specific training with skills assessments aligns learning to job duties and identifies gaps early, enabling safer, faster adoption and effective handoff to operations.

B ✗ — Require a single-day, company-wide cloud overview for all employees and consider the work done.

Rationale: A one-off general briefing raises awareness but doesn't build the deep, role-specific skills needed for operating cloud services.

C ✗ — Immediately reassign existing staff away from cloud projects and hire consultants to run operations indefinitely.

Rationale: Offloading work to consultants may delay capability building; it doesn't establish long-term internal expertise and ownership.

D ✗ — Delay any training until after migration completes to avoid distracting delivery teams.

Rationale: Postponing training risks operational errors and knowledge gaps during migration when teams need skills most.

Explanation

When adopting cloud services, preparing people for new operational models is essential. The most effective approach focuses on training that maps directly to the responsibilities teams will have, and measuring skill levels to close gaps before critical phases. Providing role-based learning ensures engineers, operators, and administrators gain practical, relevant knowledge instead of superficial awareness. It also supports safe transitions and reduces reliance on external help.

Why the correct choice fits. Targeted, role-based training plus assessments equips specific job functions with required capabilities, identifies weaknesses early, and enables phased competency checks that match migration milestones.

Why the other choices are weaker:

- A single-day, company-wide overview only increases general awareness and does not produce the hands-on skills required for operating cloud resources.
- Permanently assigning consultants avoids building internal expertise and creates dependency rather than sustainable capability.
- Waiting until after migration completes leaves teams unprepared during the most critical period, increasing risk of outages and misconfigurations.

Key Concepts

- **Role-based training:** Instruction tailored to the tasks and responsibilities of specific job roles, producing practical, job-ready skills.

- **Skills assessment:** Structured evaluation that identifies knowledge or ability gaps so learning can be targeted before failures occur.
- **Sustainable capability:** Building internal expertise to operate and improve cloud environments without permanent external dependency.

Common Pitfalls

- Assuming a single overview session is sufficient for operational readiness leads to gaps in practical ability.
- Relying solely on external contractors prevents knowledge transfer and long-term ownership.

15.59 — Service Catalog for controlled self-service (D3.T3.1)

Domain 3 • Task 3.1 ↑ Back to Question 15.59

Which outcomes are provided by organizing approved IT services into a portfolio in AWS Service Catalog?

Options & Rationales

A ✓ — Defines which users or groups can access a curated set of preapproved products

Rationale: A portfolio associates access controls with grouped products so only authorized identities can see and provision those offerings.

B ✗ — Replaces IAM by storing user credentials required to launch resources

Rationale: Service Catalog integrates with identity services and roles but does not act as an identity store or replace IAM.

C ✗ — Provides a built-in cost-reporting dashboard with detailed billing for each product

Rationale: Service Catalog controls provisioning and access; cost reporting is handled by billing tools like Cost Explorer and Cost and Usage Reports.

D ✓ — Groups multiple product templates so administrators can manage approvals and visibility centrally

Rationale: Portfolios let administrators collect related product templates and control their visibility and lifecycle from a single place.

E ✗ — Automatically converts on-premises servers into managed AWS instances

Rationale: Service Catalog does not perform server migrations; it governs and provides templates for provisioning, but migration tools are separate services.

Explanation

AWS Service Catalog organizes approved, deployable templates into named collections so teams can centrally manage which offerings are available to which users. A portfolio is this collection; it is used to group related product templates and apply access rules. This enables administrators to control visibility, enforce organizational standards, and manage product versions

from one place. Service Catalog does not perform migrations, handle billing dashboards, or replace identity services — those responsibilities belong to other AWS services.

Why the correct choices are right and others are wrong. Organizing templates into a portfolio lets administrators manage product visibility and which users or groups can provision them, matching the portfolio's purpose of centralized governance. Grouping multiple product templates under a portfolio also simplifies approval workflows and lifecycle management, which supports consistent provisioning across the organization. Options that suggest automatic server conversion, built-in billing dashboards, or replacing IAM describe functions outside Service Catalog's scope; those are handled by migration, billing, and identity services respectively.

Key Concepts

- **Portfolio grouping:** A logical collection that holds multiple product templates and centralizes administration and visibility control.
- **Access control association:** Portfolios are used to set which identities or groups can view and provision the contained products.

Common Pitfalls

- Confusing provisioning governance with migration functionality leads to choosing migration as a portfolio benefit, which is incorrect. Service Catalog governs deployments; migration is a separate task.
- Assuming Service Catalog replaces billing or identity services misunderstands its governance role; it integrates with IAM and billing tools rather than providing those functions itself.

15.60 — Customer-only security tasks (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 15.60

Which of the following best defines data classification as a customer responsibility under the shared responsibility model?

Options & Rationales

A ✗ — Managing AWS infrastructure hardware lifecycle, including physical server maintenance in a region.

Rationale: Physical hardware maintenance is AWS's responsibility for the cloud infrastructure, not a customer task.

B ✗ — Issuing service-level encryption for all AWS managed service backups automatically.

Rationale: While encryption options exist, automatically issuing service-side encryption for managed backups is an AWS capability; customers control policies and key ownership.

C ✓ — Assigning sensitivity levels to datasets and specifying handling requirements for each category.

Rationale: Data classification means labeling data by sensitivity and defining how each label must be protected and accessed; this is the customer's responsibility.

D ✗ — Configuring AWS physical network cabling between Availability Zones.

Rationale: Physical network cabling and inter-AZ connectivity are handled by AWS as part of the global infrastructure, not by customers.

Explanation

Data classification is the process where an organization determines how sensitive each dataset is and what protections and access controls are required. This activity includes assigning labels or categories (for example, public, internal, confidential), documenting handling and retention rules, and ensuring that people and systems treat data according to its assigned category. Under the shared responsibility model, cloud providers manage the underlying physical infrastructure and some managed service controls, while customers are responsible for classifying their own information and applying appropriate protections.

Why the correct choice matches. The correct choice describes labeling data by sensitivity and specifying handling requirements, which captures the essence of data classification and the customer's governance responsibility. It emphasizes that owners must decide what data needs stronger controls and how it should be processed, stored, and shared.

Why the other choices are incorrect. One incorrect option describes physical server maintenance; that concerns the provider's operational duties for infrastructure and is not a customer task. Another incorrect option suggests automatically issuing service-level encryption for managed backups; although encryption features exist, selecting encryption strategy and key ownership is the customer's decision, but the phrasing implies a provider-managed action rather than customer classification. The final incorrect option mentions configuring physical network cabling between Availability Zones, which is an infrastructure responsibility managed by the cloud provider.

Key Concepts

- **Data classification:** Assigning categories or labels to information so appropriate protections, access controls, and retention policies can be applied.
- **Shared responsibility model:** The division of cloud security and governance tasks between the cloud provider (infrastructure, physical security) and the customer (data, identities, configurations).
- **Handling requirements:** Rules tied to data categories that define encryption, access restrictions, and lifecycle management obligations.

Common Pitfalls

- Confusing provider-managed infrastructure duties with customer governance tasks leads to mistaken assumptions about who must label and protect data.

- Assuming technical controls alone (like enabling encryption) complete classification; classification is a governance decision that drives control choices.

Glossary

- **Shared responsibility model:** Framework that clarifies which security and operational tasks the cloud provider performs and which remain the customer's responsibility.
- **Sensitivity label:** A marker that indicates the confidentiality or criticality of a dataset and informs protection requirements.

15.61 — Elastic Beanstalk for PaaS (D3.T3.1)

Domain 3 • Task 3.1 ↑ Back to Question 15.61

A team is deploying a web app on AWS Elastic Beanstalk and wants capacity to adjust automatically. Which actions are INCORRECT for using Elastic Beanstalk's built-in scaling?

Options & Rationales

A ✗ — Configure scaling rules to add or remove instances based on average CPU utilization.

Rationale: Defining simple metrics-based scale-out/in rules aligns with Elastic Beanstalk's automatic scaling.

B ✗ — Use the environment's scaling settings to set minimum and maximum instance counts.

Rationale: This uses Elastic Beanstalk's managed scaling controls to bound capacity and is the correct approach.

C ✓ — Edit the Auto Scaling group directly in the EC2 console to override environment settings long-term.

Rationale: Elastic Beanstalk manages the Auto Scaling group; direct edits cause configuration drift and can be overwritten by the service.

D ✓ — Disable the platform's scaling and manually start/stop EC2 instances over SSH to control capacity.

Rationale: Manually managing instances bypasses Elastic Beanstalk's primary scaling mechanism and creates operational risk; it is an anti-pattern.

E ✗ — Set a reasonable maximum instance limit to cap scale-out capacity.

Rationale: Limiting the maximum instance count is a valid way to control cost while still using managed scaling.

Explanation

Elastic Beanstalk is a managed application platform that handles provisioning and capacity adjustments for you. Its environments include an Auto Scaling group that changes the number of instances based on simple rules and bounds

you configure. The intended operating model is to declare capacity targets (for example, minimum and maximum instances and basic scale triggers), and the service automatically carries out scaling actions.

Using the environment's scaling settings and defining metric-based rules keeps configuration centralized and managed by the platform, which is the correct way to achieve automatic capacity adjustments. Setting a maximum instance limit is also appropriate because it constrains scale-out while still relying on automatic scaling.

Disabling the platform's scaling to manually start or stop EC2 instances is an anti-pattern because it circumvents the core managed capability and increases operational toil and risk. Editing the Auto Scaling group directly in the EC2 console is also a violation of the managed model; Elastic Beanstalk owns that resource and may overwrite out-of-band changes, creating configuration drift and unstable behavior.

Key Concepts

- **Elastic Beanstalk environment:** A managed context that provisions and controls application resources on your behalf.
- **Auto Scaling (managed by Beanstalk):** The primary mechanism for adjusting instance count automatically based on simple rules and limits.
- **Configuration drift:** Changes made outside the managed service (e.g., direct ASG edits) that can be reverted or conflict with desired settings.

Common Pitfalls

- Bypassing managed scaling by manually adding/removing instances or directly editing the Auto Scaling group, which leads to drift and unreliable capacity control.

15.62 — AWS-managed security tasks (D2.T2.1)

Domain 2 • Task 2.1 ↑ Back to Question 15.62

A media company stores large video files in Amazon S3. The security policy mandates encryption at rest, but the team wants the lowest operational overhead and does not need to control key policies or grants. Which option best meets the requirement?

Options & Rationales

A ✗ — Amazon S3 server-side encryption (SSE-KMS) using a customer managed KMS key

Rationale: Encrypts at rest but you must define and maintain KMS key policies and grants—more operational overhead than needed here.

B ✗ — Client-side encryption using customer managed keys in AWS KMS before uploading to S3

Rationale: Encrypts before upload but shifts all key management and cryptographic operations to your applications—the highest operational burden.

C ✗ — Use TLS for Amazon S3 service endpoints only

Rationale: Protects data in transit only; does not encrypt objects at rest in S3, so it fails the requirement.

D ✓ — Amazon S3 server-side encryption (SSE-S3) with AWS-managed keys

Rationale: Meets at-rest encryption with minimal effort; AWS manages keys and crypto operations. Trade-off: less granular key policy control.

Explanation

The requirement is encryption at rest with the lowest operational overhead and no need for custom key policies or grants. On Amazon S3, there are two primary server-side encryption choices. With server-side encryption using S3-managed keys (SSE-S3), AWS manages the encryption keys and all cryptographic operations on the service side. This satisfies at-rest encryption while removing key lifecycle work from the customer, which optimizes for simplicity.

With server-side encryption using AWS Key Management Service (SSE-KMS), AWS operates the KMS service and its hardware security modules, but the customer defines and maintains key policies and grants for the chosen KMS key. SSE-KMS is appropriate when you require granular key permissions and explicit grants, but that added control introduces additional operational effort. Client-side encryption also meets at-rest requirements, but it pushes all encryption and key management to the application, creating the highest overhead. Finally, Transport Layer Security (TLS) protects data in transit between clients and S3 endpoints; it does not encrypt objects at rest.

Given the scenario's focus on minimal effort and no need for key-policy control, server-side encryption with SSE-S3 is the most optimized choice. This aligns with the shared responsibility model: AWS operates service-side encryption systems and manages S3-managed keys, allowing customers to focus on data usage rather than cryptographic administration.

Key Concepts

- **Server-side encryption options in S3:** SSE-S3 offloads key and crypto operations to AWS; SSE-KMS gives customer-controlled key policies and grants.
- **Shared responsibility for encryption:** AWS manages the KMS service and service-side encryption systems; customers choose configurations and, for SSE-KMS, define key policies.
- **Encryption in transit vs at rest:** TLS protects data over the network; separate mechanisms (SSE-S3 or SSE-KMS) protect data stored in S3.

Common Pitfalls

- Assuming TLS alone satisfies at-rest encryption requirements.
- Choosing SSE-KMS when key-policy control isn't needed, adding unnecessary operational overhead.

Glossary

- **SSE-S3:** S3 server-side encryption where AWS manages keys and cryptographic operations.
- **SSE-KMS:** S3 server-side encryption integrated with AWS KMS; customer defines key policies and grants for the KMS key.

15.63 — Route 53 scalable DNS (D3.T3.5)

Domain 3 • Task 3.5 ↑ Back to Question 15.63

A company runs identical web applications in two AWS Regions behind Application Load Balancers. The company wants users to be directed to the Region with the lowest network latency, and Route 53 must automatically stop returning answers for any Region whose endpoints become unhealthy. Which Route 53 features should the company use?

Options & Rationales

A ✗ — Weighted routing policy with equal weights across Regions.

Rationale: Weighted routing splits traffic by configured weights (e.g., canary/A/B), not by lowest latency, so it does not meet the performance requirement.

B ✗ — Geolocation routing policy based on user country or continent.

Rationale: Geolocation routes by the requester's location, not measured latency, so it cannot guarantee the lowest-latency Region.

C ✓ — Route 53 health checks associated with DNS records to remove unhealthy endpoints from responses.

Rationale: Health checks monitor endpoints and stop returning unhealthy answers, enabling automatic failover behavior with the chosen routing policy.

D ✓ — Latency-based routing policy in Amazon Route 53 to direct users to the lowest-latency Region.

Rationale: Latency-based routing uses measured network latency to choose the nearest-performing Region, meeting the performance requirement.

E ✗ — Multi-value answer routing to return multiple healthy IPs to clients.

Rationale: Multi-value answer improves simple load distribution and health filtering but provides no latency-based selection.

F ✗ — Failover routing policy with a primary Region and a secondary Region.

Rationale: Failover routing is active-passive. It does not distribute to the lowest-latency Region across two active stacks.

Explanation

Route 53 offers multiple routing policies and health checking to improve performance and resilience. To meet the stated goals, two capabilities must work together: a policy that selects the best Region for performance, and a

mechanism to automatically exclude unhealthy endpoints. Latency-based routing chooses the Region that provides the lowest measured network latency to the user, optimizing performance across multiple AWS Regions. Route 53 health checks monitor endpoints (or CloudWatch alarms) and remove unhealthy answers from DNS responses. When these are combined, users are directed to the lowest-latency Region among only the healthy endpoints, and if a Region becomes unhealthy, its answers are no longer returned, effectively failing traffic over to the remaining healthy Region.

Other policies do not satisfy both requirements simultaneously. Weighted routing is intended for traffic shifting scenarios (such as canary or A/B testing) and does not consider latency. Failover routing is designed for active-passive architectures with a designated primary and secondary; it does not direct traffic to the lowest-latency Region in an active-active setup. Geolocation routing uses the requester's geographic location (country/continent), which is not a guarantee of network latency. Multi-value answer returns multiple healthy records to enable basic client-side load distribution but does not factor in latency metrics.

Key Concepts

- **Latency-based routing:** Selects the Region with the lowest measured latency to the requester, improving end-user performance for multi-Region deployments.
- **Route 53 health checks:** Monitor endpoints or CloudWatch alarms and remove unhealthy answers from DNS responses, enabling automatic failover of DNS answers.
- **Policy-to-requirement mapping:** Choose a routing policy that directly addresses the goal (performance vs. traffic shifting vs. location vs. active-passive DR).

Common Pitfalls

- Confusing geolocation or weighted routing with latency-based selection; neither guarantees lowest latency.
- Assuming failover routing supports active-active latency optimization; it is for active-passive designs only.

15.64 — IaC promotes repeatability & op excellence (D1.T1.2)

Domain 1 • Task 1.2 ↑ Back to Question 15.64

A company uses a template-based deployment system to create identical environments. Which outcome best illustrates the property where applying the same template multiple times leaves the environment unchanged after the first successful run?

Options & Rationales

A ✗ — Storing all templates in a versioned repository for history and rollback.

Rationale: Version control helps track changes and revert, but it does not guarantee repeated runs produce the same resulting environment.

B ✓ — Deterministic deployments that yield the same final state when run repeatedly.

Rationale: This describes the concept where rerunning the same template does not alter resources beyond the intended state, ensuring predictable, repeatable deployments.

C ✗ — Breaking templates into smaller files so teams can reuse parts across projects.

Rationale: Modular templates improve reusability and composition, but modularity alone does not ensure identical outcomes on repeated application.

D ✗ — Running automated checks to validate templates before creating resources.

Rationale: Pre-deployment validation reduces errors, yet it does not by itself make multiple applications converge to the same final state.

Explanation

The question targets the property that applying the same deployment template multiple times results in no unexpected changes after the initial application. This property means templates express the desired end state, and repeated execution reconciles the live environment to that declared state rather than creating differences on each run. The correct answer names this predictable, repeatable behavior.

The other choices describe useful infrastructure-as-code practices but not the specific property asked. Storing templates in a versioned repository provides change history and rollback capability but is about lifecycle management of the template itself, not the runtime effect of reapplying it. Splitting templates into reusable parts supports composition and maintainability, and validating templates before deployment prevents errors; neither guarantees that running the same template repeatedly will leave the environment unchanged.

Key Concepts

- **Desired-state declarations:** Templates describe the intended end state for resources so the deployment system can adjust the environment to match that state.
- **Idempotent operations:** Running the same configuration repeatedly should produce the same outcome, preventing unintended changes after the first successful execution.
- **Version control vs. runtime behavior:** Keeping templates in a repository manages changes over time, while idempotence addresses how the system behaves when the same template is applied multiple times.

Common Pitfalls

- A common mistake is confusing good practices with this specific property: using version control, modular templates, or testing are valuable, but assuming they ensure repeated runs leave the environment unchanged is incorrect. Another pitfall is thinking validation equals idempotence; validation checks structure and rules, not the runtime convergence behavior.

15.65 — Managed DBs vs self-managed on EC2 (D3.T3.4)

Domain 3 • Task 3.4 ↑ Back to Question 15.65

A startup runs a MySQL database on a single Amazon EC2 instance. Traffic is unpredictable with sudden spikes. The team wants to reduce operational burden (patching, backups), improve availability, and pay mostly for the database capacity actually consumed. The solution must handle bursts automatically, provide Multi-AZ durability with near-instant failover, enable point-in-time recovery and encryption, and require minimal application changes. Which is the BEST solution?

Options & Rationales

A ✗ — Amazon RDS for MySQL with Multi-AZ, read replicas, and Performance Insights; improves availability and visibility but uses fixed instances and manual scaling.

Rationale: Managed and reliable, but capacity is provisioned per instance. Read replicas help reads only; it does not auto-scale write capacity or minimize idle cost like serverless.

B ✗ — Amazon DynamoDB with on-demand capacity and global tables; scales automatically but requires moving from relational MySQL to NoSQL data modeling.

Rationale: Great for variable traffic, but it is NoSQL. Migrating a MySQL app typically requires reworking schemas and queries, violating the minimal-change requirement.

C ✗ — Self-managed MySQL on Amazon EC2; build your own replication, backups, patching, monitoring agents, and recovery runbooks.

Rationale: Retains undifferentiated heavy lifting and operational risk. You must design replication, patching, backups, monitoring, and recovery yourself; no automatic scaling or managed Multi-AZ.

D ✓ — Amazon Aurora Serverless v2 (MySQL-compatible) with Multi-AZ, automated backups, KMS encryption; scales capacity in fine-grained units to handle spikes and reduce idle cost.

Rationale: Aurora Serverless v2 is MySQL-compatible and automatically scales to match bursts. It delivers Multi-AZ durability, near-instant failover, automated backups/PITR, and integrated KMS/TLS—minimizing admin effort.

Explanation

Managed database services such as Amazon RDS, Amazon Aurora, and Amazon DynamoDB offload undifferentiated heavy lifting: provisioning, patching, backups, point-in-time recovery (PITR), monitoring, and Multi-AZ high availability. They integrate encryption at rest with AWS KMS and TLS in transit and improve the Operational Excellence, Reliability, and Security pillars.

For unpredictable spikes with a goal to pay primarily for capacity consumed and to minimize administration, Amazon Aurora Serverless v2 is the best fit when the workload is MySQL-compatible. It scales capacity in fine-grained units automatically, uses distributed, fault-tolerant storage, and provides Multi-AZ durability with near-instant failover. It also includes automated backups and PITR with integrated KMS/TLS, and typically requires minimal application changes compared to other replatforming options.

Amazon RDS for MySQL (provisioned) delivers managed backups, Multi-AZ failover, and Performance Insights, but you select fixed instance sizes and scale manually. Read replicas primarily improve read throughput and do not increase write capacity, so this option does not optimally address sudden, unpredictable spikes or idle-cost reduction.

Amazon DynamoDB with on-demand capacity can absorb bursts automatically and offers global tables for multi-Region replication, but it is a NoSQL service. Moving from MySQL often requires redesigning data models and queries, conflicting with the requirement for minimal application changes.

Continuing to self-manage MySQL on EC2 keeps the team responsible for replication, patching, backups, monitoring agents, and recovery runbooks, increasing operational risk and effort and not providing automatic scaling or managed Multi-AZ behavior.

Key Concepts

- **Managed databases vs. self-managed:** Managed services handle patching, backups, monitoring, and Multi-AZ, reducing operational toil and risk.
- **Aurora Serverless v2 scaling:** Automatically adjusts capacity in fine-grained units for bursty workloads, lowering idle cost while maintaining availability.
- **RDS read replicas:** Improve read scaling only; they do not scale writes and still require provisioned instance sizing.
- **NoSQL vs. relational:** DynamoDB offers elastic scale but typically requires data model and query redesign versus MySQL-compatible Aurora.

Common Pitfalls

- Assuming read replicas increase write capacity or provide automatic, fine-grained scaling for all traffic.
- Choosing DynamoDB purely for elasticity without considering the re-architecture from relational to NoSQL.

Glossary

- **Multi-AZ:** Architecture that replicates across Availability Zones for high availability and failover.
- **Point-in-time recovery (PITR):** Ability to restore data to a specific time within a retention window.

Appendix A — AWS CLF-C02 Curriculum Map & Legend

How to read the IDs

- **D#** = Domain, **T#.#** = Task.
- Example: **D2.T2.2** → Domain 2, Task 2.2.

Why you see breadcrumbs in Explanations

Explanations show a quick breadcrumb like *Domain 2 • Task 2.2.* This anchors each item to the official blueprint so you can review weaker areas deliberately.

Domain 1 — Cloud Concepts (24%)

- **Task 1.1:** Define the benefits of the AWS Cloud (D1.T1.1)
- **Task 1.2:** Identify design principles of the AWS Cloud (D1.T1.2)
- **Task 1.3:** Understand the benefits of and strategies for migration to the AWS Cloud (D1.T1.3)
- **Task 1.4:** Understand concepts of cloud economics (D1.T1.4)

Domain 2 — Security and Compliance (30%)

- **Task 2.1:** Understand the AWS shared responsibility model (D2.T2.1)
- **Task 2.2:** Understand AWS Cloud security, governance, and compliance concepts (D2.T2.2)
- **Task 2.3:** Identify AWS access management capabilities (D2.T2.3)
- **Task 2.4:** Identify components and resources for security (D2.T2.4)

Domain 3 — Cloud Technology and Services (34%)

- **Task 3.1:** Define methods of deploying and operating in the AWS Cloud (D3.T3.1)
- **Task 3.2:** Define the AWS global infrastructure (D3.T3.2)
- **Task 3.3:** Identify AWS compute services (D3.T3.3)
- **Task 3.4:** Identify AWS database services (D3.T3.4)
- **Task 3.5:** Identify AWS network services (D3.T3.5)
- **Task 3.6:** Identify AWS storage services (D3.T3.6)
- **Task 3.7:** Identify AWS AI/ML and analytics services (D3.T3.7)
- **Task 3.8:** Identify services from other in-scope AWS service categories (D3.T3.8)

Domain 4 — Billing, Pricing, and Support (12%)

- **Task 4.1:** Compare AWS pricing models (D4.T4.1)
- **Task 4.2:** Understand resources for billing, budget, and cost management (D4.T4.2)
- **Task 4.3:** Identify AWS technical resources and AWS Support options (D4.T4.3)

About the Author & Publisher

About the Author

Fuad Efendi (Éfendiev), M.Sc. (Mathematics, Lomonosov Moscow State University)
Series Editor, Mastery Exam Prep — Head Editor, Tokenizer Inc.

I began my career teaching mathematics and quantitative methods at the university level and later moved into industry roles leading engineering and cloud initiatives. That mix of **deep theory** and **hands-on delivery** shapes how I write about cloud: precise, practical, and focused on what helps you pass the exam **and** do real work.

As Series Editor for *Mastery Exam Prep*, I set the standards you see in this book: tight **blueprint alignment**, realistic **scenario design**, plain, direct language, and **consistent rationales** that explain not just *what* is correct, but *why*. Content is reviewed with AWS-certified professionals to verify service behavior and design guidance.

About the Publisher

These books are published by **Tokenizer Inc.** under the **Mastery Exam Prep** imprint, an independent publisher focused on professional certification and skills-based learning. Our aim is to produce resources that are:

- Aligned to official exam blueprints.
- Scenario-focused rather than trivia-driven.
- Written in clear, approachable language.

How We Create and Review Content

Readers sometimes ask whether books like this are "AI-generated." The honest answer is that we use modern tools—including AI assistants—for **drafting, refactoring, and checking** ideas, but every question, answer choice, and explanation is ultimately **selected, edited, and approved by humans**.

Behind each title is a structured workflow designed to mirror how real exams work, not how random question generators behave. For every exam we support, we:

- **Align to the official blueprint:** mapping questions and explanations to the exam's published domains, tasks, and weightings.
- **Design question styles and ratios:** using multiple question archetypes (scenario, comparison, troubleshooting, quantitative, definition, principles, multi-select, etc.) with realistic mixes of single-response and multi-response items.

- **Target real exam pacing:** crafting questions to be answerable in roughly **2–3 minutes**, not 10-second trivia and not half-hour puzzles.
- **Draft and refine by humans:** writing and iterating stems, options, and rationales to match the tone and difficulty of real-world exams.
- **Use tool-assisted checks:** applying internal tools and AI to surface potential ambiguities, edge cases, and coverage gaps.
- **Engage independent SME review:** working with certified subject-matter experts to audit question logic, service behavior, and design guidance.
- **Update iteratively:** incorporating reader feedback, errata, and changes to official exam objectives into future versions of the book.

In short, we treat AI as a **power tool**, not an author. The framework, standards, and final editorial decisions are set by humans, and the responsibility for accuracy, clarity, and fairness rests with experienced editors and subject-matter experts.

In addition to digital books (EPUB and PDF), Mastery Exam Prep offers companion online resources, quick-reference guides, and interactive practice through our website. Visit MasteryExamPrep.com for bonus materials, errata, and future titles in development.

Stay Connected with Mastery Exam Prep

Mastery Exam Prep focuses on expert-written, scenario-based practice and detailed explanations for today's most in-demand cloud and IT certifications. To discover new releases, updated editions, and companion tools such as our web-based practice app (and upcoming **Mastery Cloud** mobile apps), visit MasteryExamPrep.com or contact editor@masteryexamprep.com to stay informed about future developments and new titles.

Thanks, Feedback & How to Leave a Review

If this book helped you prepare for the **AWS Certified Cloud Practitioner (CLF-C02)** exam, we'd be grateful if you'd share your experience—either by leaving an honest review where you obtained the book, by sending a short testimonial by email, or by visiting MasteryExamPrep.com. Your feedback helps other candidates decide if this resource fits their learning style and helps us prioritize what to improve next, including future online resources and titles.

You're also welcome to suggest **future titles or exam coverage** you'd like to see (for example, other AWS certifications, Azure, GCP, security, or finance/professional designations).

Quality & Updates

We periodically publish corrections and enhancements. Minor fixes are released as quiet updates; larger improvements roll into new editions.

Found an issue, have a suggestion, or want to share a testimonial? Email editor@masteryexamprep.com with:

- The book title and format (for example, *CLF-C02 — Practice Exams, PDF* or *EPUB*)
- The app or reading software you're using (if applicable)
- The **question ID** (for example, **5.12**), section, or page number
- (Optional) A short testimonial we may quote in future editions or on our website

We'll review your note, queue any confirmed fixes for the next update, and greatly appreciate any feedback that helps make future editions better.

Ready for the Full 15-Exam Mastery Edition?

You've just worked through the **Lite Edition** of the **AWS Certified Cloud Practitioner (CLF-C02)** practice exam series, which contains **Exams 13–15** from the full 15-exam collection.

If this sample helped you:

- See how AWS concepts appear in realistic, exam-style scenarios
- Understand *why* certain answers are right or wrong
- Identify specific gaps in your knowledge and exam readiness…

…then you're exactly who the **full 15-exam book (975 questions)** is designed for.

What You Get in the Full Edition

- **15 full-length practice exams** (65 questions each), including the three exams you've just completed, all aligned to the official CLF-C02 domains.
- **Question-only section** for timed simulations without explanations, ideal for final exam rehearsals.
- **Dedicated explanations section** that walks through:
  - Why the correct option is best given the constraints
  - Why each distractor fails (cost, security, reliability, operations, or scope)
- Blueprint-driven coverage of:
  - Cloud concepts
  - Security and compliance
  - Technology and architecture
  - Billing, pricing, and support

The goal is not to overwhelm you with obscure edge cases, but to **build confidence** through many realistic scenarios that feel close to the real CLF-C02 exam.

Next Steps

- Get the **full 15-exam PDF/EPUB bundle** for this book at: masteryexamprep.com/books/aws-clf-c02/
- Explore more AWS exam resources and practice materials: masteryexamprep.com/exams/aws/
- Discover additional Cloud & IT learning paths and verticals: masteryexamprep.com/verticals/cloud-it/

Thank you for spending time with this Lite Edition.

If it helped you, feel free to share **MasteryExamPrep.com** with other learners.