The evolving café architecture

Version	Business Reason for Update	Technical Requirements and Architecture Update
V1	Build a static website for a small business.	Host the website on Amazon S3.
V2	Update the website to support dynamic content and online ordering.	Deploy the web application and database on Amazon EC2.
V3	Reduce the effort to maintain the database and secure its data.	Separate web and database layers. Migrate the database to Amazon RDS on a private subnet.
V4	Enhance the security of the web application.	Use Amazon VPC features to configure and secure public and private subnets.
V5	Ensure that the website can handle an expected increase in traffic and remain highly available and resilient to failure.	Add a load balancer, implement auto scaling on the EC2 instances, and distribute compute and database instances across two Availability Zones.
V6	Automate deployments so that the café can consistently deploy, manage, and update café resources across Regions.	Build a version-controlled CloudFormation template to deploy the network and application layers. Deploy the CloudFormation stack to another Region.
V7	Add reporting capabilities while reducing the operational and maintenance burden, improving performance, and reducing costs.	Deploy Lambda functions that connect to the Amazon RDS database and generate a report based on a schedule.

Cloud Computing Roles Summary

In the field of cloud computing, various roles exist, each with specific skills and responsibilities. Understanding these roles is essential for anyone looking to start or transition into a cloud computing career. Below are some common roles in cloud computing:

---

1. IT Professional

- **Skills and Responsibilities:**
  - Generalists who manage applications and production environments.
  - Highly technical with varying experience in cloud technologies.
  - May specialize in areas like security or storage.
- **Job Titles:**
  - IT Administrator
  - Systems Administrator
  - Network Administrator

---

2. IT Leader

- **Skills and Responsibilities:**
  - Lead teams of IT professionals.
  - Oversee day-to-day operations and manage budgets.
  - Stay informed about technologies and choose new ones for projects.

    - Hands-on in early project stages, then delegates tasks.
- **Job Titles:**
    - IT Manager
    - IT Director
    - IT Supervisor

---

3. Developer

- **Skills and Responsibilities:**
    - Write, test, and fix code.
    - Focus on application-level project development.
    - Work with APIs and SDKs, using sample code.
    - May specialize in areas such as security or storage.
- **Job Titles:**
    - Software Developer
    - System Architect
    - Software Development Manager

---

4. DevOps Engineer

- **Skills and Responsibilities:**
    - Build and maintain the infrastructure for applications, often in the cloud.
    - Follow guidelines provided by cloud architects.
    - Experiment to enhance deployment processes.
- **Job Titles:**
    - DevOps Engineer
    - Build Engineer
    - Reliability Engineer

---

5. Cloud Architect

- **Skills and Responsibilities:**
    - Stay updated on new technologies and determine which to use.
    - Provide documentation, processes, and tools for developers.
    - Facilitate developer innovation while managing costs, performance, reliability, and security.
- **Job Titles:**
    - Cloud Architect
    - Systems Engineer
    - Systems Analyst

Course Perspective

Throughout this course, participants will take on the perspective of a cloud architect. This involves understanding the architecture of applications, selecting appropriate technologies based on business needs, and addressing challenges related to resource management, cost optimization, and best practices for performance, reliability, and security. The roles described align with the principles outlined in the AWS Well-Architected Framework, which will be discussed in detail throughout the course.

Section 1 **Cloud Architecting**

Introduction to Cloud Architecting

Overview of Cloud Computing and AWS

- **Early Challenges**: In the early 2000s, Amazon faced difficulties in creating an ecommerce service for third-party sellers. Their initial attempts at building a scalable and highly available shopping platform were hindered by poorly planned architectures and long development times, often taking three months just to establish database and storage components.
- **Transition to Cloud Services**: To address these issues, Amazon introduced well-documented APIs, allowing for better organization and development. In 2006, they launched Amazon Web Services (AWS), starting with core services like Amazon Simple Queue Service (SQS), Amazon Simple Storage Service (S3), and Amazon Elastic Compute Cloud (EC2).

What is Cloud Architecture?

- **Definition**: Cloud architecture involves designing solutions that leverage cloud services to fulfill an organization's technical requirements and business objectives.
- **Analogy**: Similar to constructing a building, cloud architecture requires a solid foundation. Here's how the process parallels construction:
    1. **Customer Requirements**: The customer defines the business needs.
    2. **Architectural Design**: The cloud architect creates a design blueprint to meet these needs.
    3. **Implementation**: The delivery team constructs the solution based on the architect's design.
- **Benefits**: Well-architected systems enhance the likelihood that technology solutions will align with and support business goals.

Role of a Cloud Architect

- **Key Responsibilities**:
    - **Planning**: Develop a technical cloud strategy in collaboration with business leaders and analyze solutions to meet business needs.
    - **Research**: Investigate cloud service specifications, workload requirements, and existing architectures; design prototype solutions.

    - **Building**: Create a transformation roadmap with milestones, manage adoption and migration efforts.
- **Engagement**: Cloud architects work closely with decision-makers to identify business goals and ensure that technology deliverables align with these objectives. They collaborate with delivery teams to ensure the appropriate implementation of technology features.
- **Knowledge**: They possess in-depth knowledge of architectural principles and are responsible for:
    - Developing cloud strategies based on business needs.
    - Assisting with cloud migration.
    - Reviewing workload requirements and addressing high-risk issues.
    - Implementing the AWS Well-Architected Framework.

Key Takeaways

- **Cloud Architecture**: It is about applying cloud characteristics to create solutions that meet technical needs and business use cases.
- **AWS Services**: These services enable the creation of highly available, scalable, and reliable architectures.
- **Role of Cloud Architects**: They play a crucial role in managing cloud architectures, ensuring that solutions are well-aligned with business objectives and leveraging best practices for implementation.

Section 2 AWS Well-Architected Framework

Overview

The **AWS Well-Architected Framework** offers a systematic approach to evaluate cloud architectures and implement best practices. Developed from reviewing thousands of customer architectures, it consists of **six pillars**: Operational Excellence, Security, Reliability, Performance Efficiency, Cost Optimization, and Sustainability.

Six Pillars of the AWS Well-Architected Framework

1. **Operational Excellence**
    - Focuses on running and monitoring systems to deliver business value.
    - Emphasizes continuous improvement of processes and procedures.

    - Encourages viewing the entire workload as code, enabling automation and efficient operations.
2. **Security**
    - Ensures the protection of information, systems, and assets while delivering business value.
    - Stresses implementing a strong identity foundation, maintaining traceability, and applying security across all layers.
    - Involves risk assessment and mitigation strategies to prepare for security events.
3. **Reliability**
    - Addresses the ability of systems to recover quickly from disruptions and dynamically meet compute demand.
    - Helps mitigate issues such as misconfigurations and network problems.
    - Promotes designs that ensure high availability, fault tolerance, and redundancy.
4. **Performance Efficiency**
    - Aims to maximize performance by efficiently utilizing computational resources and maintaining that efficiency as demand changes.
    - Encourages the democratization of advanced technologies by using vendors to manage complexity.
    - Advocates for “mechanical sympathy,” which involves understanding how systems operate best.
5. **Cost Optimization**
    - Focuses on measuring efficiency and eliminating unnecessary expenses.
    - Emphasizes adopting the right consumption model and considering managed services to lower costs.
    - Views cost optimization as an ongoing iterative process that evolves throughout the production lifecycle.
6. **Sustainability**
    - Concentrates on building architectures that maximize efficiency and minimize waste.
    - Addresses the long-term environmental, economic, and societal impacts of business activities.
    - Involves reducing the downstream impact by choosing efficient hardware, software, and programming practices.

Using the AWS Well-Architected Tool (WA Tool)

- The **AWS Well-Architected Tool** helps evaluate workloads against AWS best practices.
- It provides:
    - **Guidance**: Access to knowledge used by AWS architects.
    - **Action Plan**: Step-by-step recommendations for building better workloads.
    - **Consistent Process**: A method to review and measure cloud architectures, enabling identification of next steps for improvement.

The tool allows users to define their workloads and answer questions across the six pillars, delivering insights that help minimize operational costs and enhance system reliability.

Key Takeaways

- The AWS Well-Architected Framework provides a structured way to evaluate and improve cloud architectures.
- Each of the six pillars has foundational questions to assess alignment with cloud best practices.
- The AWS WA Tool offers resources for reviewing workloads and implementing best practices effectively.

Section 3 best practices for building solutions on AWS:

1. Using Loosely Coupled Components

- **Independent Components:** Design architectures with independent components to avoid disruptions and scaling issues.
- **Intermediaries:** Use managed solutions like load balancers (e.g., Elastic Load Balancing) and message queues to handle failures and scale components.

2. Designing Services, Not Servers

- **Leverage AWS Services:** Use a wide range of AWS services beyond traditional servers, such as containers and serverless solutions (e.g., AWS Lambda, Amazon SQS, Amazon DynamoDB).
- **Managed Solutions:** Consider using managed services to reduce overhead and improve performance.

3. Choosing the Right Database Solution

- **Match Technology to Workload:** Select database solutions based on specific needs like read/write requirements, storage needs, latency, and query nature.
- **AWS Recommendations:** Use AWS guidance for selecting an appropriate database service based on your application environment.

4. Avoiding Single Points of Failure

- **Assume Failure:** Design architectures that can recover from failures by implementing secondary databases or replicas.
- **Disposable Resources:** Treat resources as disposable and design applications to support hardware changes.

5. Optimizing for Cost

- **Variable Expenses:** Take advantage of AWS's pay-as-you-go model to only pay for what you use.
- **Resource Monitoring:** Regularly assess resource sizes and types, monitor usage metrics, and shut down unused resources.

6. Using Caching

- **Improve Performance:** Minimize redundant data retrieval by temporarily storing frequently accessed data (e.g., using Amazon CloudFront with Amazon S3).
- **Reduced Costs:** Caching can lower latency and reduce costs by serving repeated requests from closer locations.

7. Securing Your Entire Infrastructure

- **Layered Security:** Build security into every layer of your infrastructure through managed services, logging, data encryption, access controls, and multi-factor authentication (MFA).
- **Granular Access Control:** Use principles like least privilege to limit access across your infrastructure components.

8. Key Takeaways for Building Solutions on AWS

- Evaluate trade-offs based on empirical data.
- Follow best practices including scalability, automation, treating resources as disposable, using loosely-coupled components, and optimizing for cost.

This framework and its associated best practices aim to enhance the reliability, efficiency, and security of architectures built on AWS, ultimately leading to better operational performance and cost management.

Section 4 AWS Global Infrastructure:

AWS Global Infrastructure Overview

1. **Key Components:**

    - **Regions:** Geographical areas that consist of two or more Availability Zones (AZs).
    - **Availability Zones:** Isolated locations within a Region, made up of one or more data centers designed for fault isolation.
    - **Local Zones:** Extensions of Regions that bring services closer to large population centers for low-latency applications.
    - **Data Centers:** Facilities where data is processed and stored, typically containing tens of thousands of servers.
    - **Points of Presence (PoPs):** Network locations that reduce latency by caching content closer to end users.
2. **Infrastructure Details:**
    - AWS spans **102 Availability Zones** across **32 geographic regions**.
    - Each Region is connected via a private global network backbone, offering lower costs and consistent network latency.
    - AWS data centers provide high availability, redundancy, and load balancing during failures.
3. **Choosing Regions and Availability Zones:**
    - The choice of Region often depends on compliance and latency requirements.
    - AWS recommends deploying applications across multiple AZs for resilience against failures.
4. **Local Zones and PoPs:**
    - Local Zones enable running latency-sensitive applications closer to end users.
    - PoPs enhance content delivery by caching popular data at edge locations, significantly improving performance.
5. **Key Takeaways:**
    - The AWS Global Infrastructure includes Regions, AZs, and edge locations to support scalable, reliable applications.
    - Effective architecture design considers the geographic location of resources to meet performance and compliance needs.

This summary encapsulates the main elements of the AWS Global Infrastructure, providing an overview of its architecture and operational considerations.

Module check

1. What is the best definition of cloud architecture?

- ○ Combining frontend and backend software and components to create highly available and scalable web services that meet the needs of an organization
- ◉ Applying cloud characteristics to a solution that uses cloud services and features to meet technical and business requirements
- ○ Designing applications in cloud-based, shared IT infrastructure by using virtual machines and fault-tolerant data stores in the cloud
- ○ Relocating traditional on-premises data centers to internet-accessible data centers that a vendor manages

2. The AWS Well-Architected Framework has five pillars. Two of the pillars are security and operational excellence. What are the other pillars of the Well-Architected Framework? (Select THREE.)

- ☑ Reliability
- ☐ Governance
- ☐ Privacy
- ☐ Performance efficiency
- ☐ Risk management
- ☐ Cost optimization

1,5,6

3. Which actions are consistent with the operational excellence pillar of the AWS Well-Architected Framework? (Select TWO.)

- [x] Review and improve processes and procedures on a continuous cycle.
- [ ] Ensure operations personnel document changes to the infrastructure.
- [x] Apply software engineering principles and methodology to infrastructure as code.
- [ ] Evaluate organizational structures and roles to identify skill gaps.
- [ ] Plan and manage the full lifecycle of hardware assets.

4. An application requires a frontend web tier of multiple servers that communicate with a backend application tier of multiple servers. Which design most closely follows Amazon Web Services (AWS) best practices?

- ( ) Assign a dedicated application server and a dedicated connection to each web server.
- ( ) Create multiple instances that each combine a web frontend and application backend in the same instance.
- (•) Design the web tier to communicate with the application tier through the Elastic Load Balancing service.
- ( ) Create a full mesh network between the web and application tiers, so that each web server can communicate directly with every application server.

KEYBOARD NAVIGATION

5. A solutions architect is developing a process for handling server failures. Which process most closely follows Amazon Web Services (AWS) best practices?

- Operations detects a system failure. They notify the systems administrator, who provisions a new server by using the AWS Management Console.
- Amazon CloudWatch detects a system failure. It notifies the systems administrator, who provisions a new server by using the AWS Management Console.
- Operations detects a system failure. They trigger automation to provision a new server.
- Amazon CloudWatch detects a system failure. It triggers automation to provision a new server

d

6. A company wants to change some functionality of their website. They are unsure of what will happen if they make the change. Which approach most closely follows Amazon Web Services (AWS) best practices?

- Change the production site while it is online. Use backups to undo the change.
- Change the production site during offline maintenance hours. Use backups to undo the change.
- Provision a new server and make changes to it. Use DNS to gradually migrate users to the new server. Shut down the original server after all users migrate.
- Test the change on an existing development server. Change the production

7

. A company stores read-only data in Amazon Simple Storage Service (Amazon S3). Most users are in the same country as the company headquarters. Some users are located around the world. Which design decision most closely follows Amazon Web Services (AWS) best practices?

- ○ Use a bucket in the AWS Region closest to the company headquarters.
- ○ Use a bucket in the AWS Region that has the lowest average latency for all users.
- ○ Replicate objects across buckets in AWS Regions around the world. Users access the bucket in the AWS Region closest to them.
- ◉ Use a bucket in the AWS Region closest to the company headquarters. All users access the data through Amazon CloudFront.

KEYBOARD NAVIGATION

8. A consultant must access a large object in an S3 bucket. They need a day to access the file. Which method for granting access most closely follows Amazon Web Services (AWS) best practices?

- ◉ Create a presigned URL to the object that expires in 24 hours, and give it to the consultant.
- ○ Enable public access on the S3 bucket. Give the object URL to the consultant.
- ○ Copy the object to a new S3 bucket. Enable public access on the new bucket. From the new bucket, get the object URL, and give it to the consultant.
- ○ Create a user account for the consultant. Grant the user account permissions to access the S3 bucket through the AWS Management Console.

9. What are the main considerations that influence which AWS Regions to use? (Select TWO.)

- ☐ Security and access control
- ☑ Latency reduction for end users
- ☐ Protection against localized natural disasters
- ☐ Application resiliency during system failures
- ☑ Compliance with laws and regulations

10. What are the main considerations that influence which Availability Zones to use? (Select TWO.)

- [ ] Security and access control
- [ ] Latency reduction for end users
- [x] Protection against localized natural disasters
- [x] Application resiliency during system failures
- [ ] Compliance with laws and regulations

Section 1 AWS Shared Responsibility Model

- **Customer Responsibilities (Security IN the Cloud)**:
    - **Data Management**: Customer data, applications, identity and access management.
    - **Configuration**: Operating system, network, and firewall settings.
    - **Encryption**: Client-side and server-side data encryption, as well as ensuring data integrity and authentication.
    - **Networking**: Protecting networking traffic through encryption and identity assurance.
- **AWS Responsibilities (Security OF the Cloud)**:
    - **Infrastructure**: Securing the foundational services (compute, storage, database, networking) and the global infrastructure (Regions, Availability Zones, Edge Locations).

Security as a Well-Architected Framework Pillar

- **Six Pillars of the Well-Architected Framework**:
    - Security
    - Operational Excellence
    - Reliability
    - Performance Efficiency
    - Cost Optimization
    - Sustainability
- **AWS Well-Architected Tool**: Helps implement best practices from the framework.

Design Principles for the Security Pillar

1. **Strong Identity Foundation**: Enforce least privilege and separate duties for authorization.
2. **Protect Data in Transit and at Rest**: Use encryption and tokenization based on data sensitivity.
3. **Security at All Layers**: Apply multiple security controls across all layers of the architecture.
4. **Minimize Direct Data Access**: Reduce the need for direct access to sensitive data.
5. **Maintain Traceability**: Monitor and audit actions in real time.
6. **Prepare for Security Events**: Have policies and tools ready for incident management.
7. **Automate Security Best Practices**: Use automated security mechanisms to enhance scalability and cost-effectiveness.

Key Security Practices

- **User Permissions**: Use policies to manage access to AWS resources based on roles and needs.
- **Principle of Least Privilege**: Grant minimum permissions required for tasks.
- **Data Protection**:
    - **In Transit**: Use TLS to protect data during transfer.
    - **At Rest**: Implement client-side and server-side encryption to protect stored data.

Takeaways

- Security is a shared responsibility between AWS and customers.
- Implementing a strong identity foundation and protecting data are crucial to enhancing security posture in the cloud.

This summary encapsulates the essential aspects of AWS security responsibilities and principles, highlighting the collaborative nature of cloud security between AWS and its customers.

Section 2 IAM Overview

- **IAM User**: Represents an individual (person or application) with a permanent set of credentials to access AWS services.
- **IAM Group**: A collection of IAM users granted identical permissions, simplifying permission management.
- **IAM Role**: An identity with temporary permissions used for specific tasks; it does not have long-term credentials.
- **IAM Policy**: A document defining permissions, specifying what resources can be accessed and the level of access granted.

Authentication Credentials

- **AWS Management Console**: Requires a username and password for sign-in.
- **AWS CLI and API Calls**: Use an access key (combination of an access key ID and secret key) for programmatic access.

Best Practices for Secure Access

1. **Principle of Least Privilege**: Assign minimum permissions necessary for users.
2. **Enable MFA**: Adds an extra security layer for the root user and other accounts.

3. **Use Temporary Credentials**: Prefer temporary credentials over long-term ones for human users.
4. **Rotate Access Keys**: Regularly rotate access keys to maintain security.
5. **Use Strong Passwords**: Ensure all accounts have complex passwords.
6. **Secure Local Credentials**: Store sensitive credentials securely.
7. **Utilize AWS Organizations**: Manage multiple AWS accounts centrally.
8. **Enable AWS CloudTrail**: Monitor actions taken in the account for security auditing.
9. **Protect the Root User**: Limit use of the root user account and monitor its activity.

Protecting the Root User

- **Create an Admin User**: For daily tasks, create a user with administrative privileges instead of using the root account.
- **Use MFA**: Always set up multi-factor authentication for the root user.

IAM User and Group Management

- **Attach Policies to Groups**: For efficient access control, attach IAM policies to groups rather than individual users.

IAM Roles

- **Temporary Security Credentials**: Roles provide temporary credentials without long-term commitments.
- **Common Use Cases**: Used for applications running on EC2, cross-account access, and mobile applications.

Examples of IAM Role Usage

1. **EC2 Instance Access**: An IAM user accesses an application on an EC2 instance by assuming a role.
2. **S3 Bucket Access**: An application on EC2 accesses an S3 bucket through an attached role.
3. **Cross-Account Access**: An IAM user in one account accesses resources in another account through a trusted role.

Key Takeaways

- Utilize IAM for fine-grained access control.
- Avoid using the root account for everyday tasks.
- Use groups for managing user permissions effectively.
- Roles allow for temporary access to resources as needed.

This summary encapsulates the fundamental concepts and best practices related to IAM in AWS, emphasizing the importance of security and proper access management.

Section 3 authorizing users with IAM policies in AWS:

IAM Policies and Permissions

1. **Types of Policies**:
    - **Identity-based Policies**: Attached to IAM users, groups, or roles. These policies specify what actions the identity can perform.
    - **Resource-based Policies**: Attached to AWS resources. They define what actions specific users or groups can perform on those resources.
2. **Policy Structure**:
    - Policies are defined in JSON format.
    - Each policy specifies the resources and operations that are allowed or denied.
    - Follow the **principle of least privilege**, granting only the permissions necessary for a user or resource to perform their tasks.

Permission Evaluation Logic

1. **Default Deny**: By default, all requests are denied.
2. **Explicit Allow and Deny**:
    - An **explicit allow** will override the default deny.
    - An **explicit deny** will override any explicit allow.
3. **Implicit Deny**: If there is no applicable explicit allow or deny, access is denied by default.
4. **Conflict Resolution**: In cases where one policy allows an action and another denies it, the more restrictive (deny) policy takes precedence.

IAM Policy Simulation

- The **IAM policy simulator** is a tool that helps test and troubleshoot policies to determine if access will be granted to IAM entities.

Examples of Policies

1. **Example 1**:
    - **Identity-based Policy**: Grants Bob permission to GET, PUT, and LIST objects in bucket X.
    - **Resource-based Policy**: Allows Bob to GET and LIST, but denies PUT. Therefore, Bob cannot PUT objects into bucket X despite the identity-based policy.
2. **Example 2**:

    - Bob's identity-based policy allows LIST for bucket Y but does not specify GET or PUT.
    - The resource-based policy for bucket Y allows GET and LIST, but not PUT. Therefore, Bob can read objects from the bucket.

Key Takeaways

- A policy defines permissions associated with an identity or resource.
- IAM supports two types of policies (identity-based and resource-based) to control access.
- Permissions determine the outcome of requests: by default denied, overridden by explicit allows or denies.

This overview encapsulates the concepts of IAM policies and permissions, emphasizing their role in securing access to AWS resources.

Section 4 IAM (Identity and Access Management) policies

IAM Policy Structure and Concepts

1. **Policy Document Structure**:
    - **Version**: Specifies the policy language version (e.g., "2012-10-17").
    - **Statement**: Contains one or more individual permission statements.
        - **Effect**: Indicates whether access is allowed (`Allow`) or denied (`Deny`).
        - **Principal**: Defines the user, account, role, or federated user that the policy applies to (for resource-based policies).
        - **Action**: Lists the actions that are allowed or denied (e.g., `s3:GetObject`).
        - **Resource**: Specifies the resources to which the actions apply (e.g., Amazon Resource Names, or ARNs).
        - **Condition** (optional): Specifies conditions that must be met for the policy to apply.
2. **Types of Policies**:
    - **Resource-based Policy**: Attached to resources like S3 buckets or DynamoDB tables. Allows specifying the principal and can grant access to other accounts.
    - **Identity-based Policy**: Attached to IAM users, groups, or roles. Specifies what actions the principal can perform on specified resources.
3. **Policy Evaluation Logic**:

    - AWS applies a logical `OR` across statements when evaluating permissions. If any statement allows access, the permission is granted, unless there is an explicit deny.
4. **Explicit Deny and Allow**:
    - An explicit deny takes precedence over any allow statement. This means that even if a user has permission to access a resource, if a deny statement applies, access will be denied.
5. **Examples of Policies**:
    - **Resource-based Policy Example**: Allows any S3 action on specified S3 resources but denies all other actions on DynamoDB or S3 resources not listed.
    - **Identity-based Policy Example**: Allows a user to manage their IAM login profile, access keys, and SSH keys.
    - **Cross-account Policy Example**: Allows one AWS account to access resources in another account.
6. **CIDR and IP Conditions**:
    - Policies can include conditions based on IP addresses using CIDR notation, allowing or denying actions based on the source IP address.
7. **Key Takeaways**:
    - IAM policies are JSON documents defining permissions.
    - Key elements include effect, action, and resources, which together determine what permissions are granted or denied.
    - Understanding the structure and evaluation of IAM policies is crucial for managing access to AWS resources securely.

Conclusion

IAM policies play a vital role in securing access to AWS resources. By understanding their structure and how to apply conditions effectively, you can implement a robust access control strategy in your AWS environment.

Module checks

1. Amazon Simple Storage Service (Amazon S3) provide a good solution for which of the following use cases?

- ○ A data warehouse for business intelligence
- ○ An internet-accessible storage location for video files that an external website accesses
- ○ Hourly storage of frequently accessed temporary files
- ○ A cluster for traditional Apache Spark and Apache Hadoop installations to process big data

2. A company is interested in using Amazon Simple Storage Service (Amazon S3) alone to host their website, instead of a traditional web server. Which types of content does Amazon S3 support for static web hosting? (Select THREE.)

- ☑ HTML files and image files
- ☑ Client-side scripts
- ☐ Server-side scripts
- ☐ Dynamic HTML files
- ☑ Video and sound files

3. Which scenarios represent a good use for Amazon Simple Storage Service (Amazon S3)? (Select TWO.)

- [ ] Housing the root volume of a live operating system
- [ ] Providing a mountable file system for Linux-based workloads
- [x] Backing up critical data
- [ ] Exposing a virtual tape library to on-premises backup systems
- [x] Storing computation and analytics data

4. A company wants to use an S3 bucket to store sensitive data. Which actions can they take to protect their data? (Select TWO.)

- [x] Enabling server-side encryption on the S3 bucket before uploading sensitive data
- [ ] Enabling server-side encryption on the S3 bucket after uploading sensitive data
- [x] Using client-side encryption to protect data in transit
- [ ] Using Secure File Transfer Protocol (SFTP) to connect directly to Amazon S3

5. A company must create a common place to store shared files. Which requirements does Amazon Simple Storage Service (Amazon S3) support? (Select TWO.)

- [x] Recover deleted files
- [x] Maintain different versions of files
- [ ] Lock a file so that only one person at a time can edit it
- [ ] Attach comments to files
- [ ] Compare file contents between files

6. A customer service team accesses case data daily for up to 30 days. Cases can be reopened and require immediate access for 1 year after they are closed. Which solution meets the requirements and is the most cost-efficient?

- Store all case data in S3 Standard so that it is available whenever needed.
- Store case data in S3 Standard. Use a lifecycle policy to move the data into S3 Standard-Infrequent Access (S3 Standard-IA) after 30 days.
- Store case data in S3 Standard. Use a lifecycle policy to move the data into Amazon S3 Glacier after 30 days.
- **Store case data in S3 Intelligent-Tiering to automatically move data between tiers based on access frequency.**

7. Which Amazon Simple Storage Service (Amazon S3) unaccelerated data transfers have an associated cost? (Select TWO.)

- [ ] IN from the internet
- [x] OUT to the internet
- [x] OUT to other AWS Regions
- [ ] OUT to other AWS services in the same AWS Region
- [ ] OUT to Amazon CloudFront

8. A company is migrating 100 terabytes (TB) of data from their on-premises data center to Amazon Simple Storage Service (Amazon S3). The company connects to Amazon Web Services (AWS) by using a single 155 megabits per second (Mbps) internet connection. Which data transfer option is the fastest and most cost-effective?

- AWS Management Console
- Amazon S3 multipart uploads
- **AWS Snowball**
- AWS Snowmobile

9. A video producer must regularly transfer several video files to Amazon Simple Storage Service (Amazon S3). The files range from 100–700 MB. The internet connection has been unreliable, causing some uploads to fail. Which solution provides the fastest, most reliable, and most cost-effective way to transfer these files to Amazon S3?

- ○ AWS Management Console
- ○ AWS Snowmobile
- ◉ Amazon S3 multipart uploads
- ○ AWS Snowball

10. Which qualities vary by AWS Region? (Select TWO.)

- ☑ Cost-effectiveness of workloads
- ☐ Data privacy
- ☐ High availability of workloads
- ☑ Service and feature availability
- ☐ Capacity for more customers

Section 1 Defining Amazon S3:

Types of Storage

1. **Block Storage**: Data is stored in fixed-sized blocks, managed by applications and file systems. Each block is identifiable and can be stored efficiently across systems.
2. **File Storage**: Data is organized in a hierarchical structure, resembling a shared network drive.
3. **Object Storage**: Data is stored as objects with associated attributes and metadata. Each object contains data, metadata, and a unique identifier (object key).

Amazon S3 Overview

- **Object Storage Service**: Amazon S3 is designed to store massive amounts of unstructured data.
- **Buckets and Objects**:
    - Data is stored as objects within buckets defined by the user.
    - Each bucket has a globally unique name and is associated with a specific AWS Region.
    - Each object can range in size from 0 bytes to a maximum of 5 TB.
    - Objects include a key, version ID, content (immutable), metadata, and sub-resources.

Components of Amazon S3

- **Bucket**: A container for objects that organizes the S3 namespace and manages storage costs and access control.
- **Object**: The basic entity stored in S3, including data and metadata.
- **Object Key**: A unique identifier for each object in a bucket.

Folder Structure Using Prefixes

- Amazon S3 allows for a folder-like structure by using prefixes in object names.
- This helps organize objects within a bucket for easier retrieval.

Benefits of Amazon S3

- **Durability**: 99.999999999% durability, ensuring data is not lost and is redundantly stored across multiple devices.
- **Availability**: 99.99% availability, allowing quick access to data with virtually unlimited storage capacity.

- **High Performance**: Capable of handling thousands of transactions per second and scales automatically to high request rates.

Key Takeaways

- Amazon S3 is an object storage service that supports massive unstructured data storage, organized into unique buckets and objects.
- The service offers significant benefits, including durability, availability, and performance.

Section 2 use cases for Amazon S3:

Common Use Cases for Amazon S3

1. **Spikes in Demand**:
    - **Web Hosting**: Amazon S3 is ideal for hosting web content, particularly when there's a need to handle extreme spikes in traffic.
2. **Static Website Hosting**:
    - S3 can host static websites consisting of HTML files, images, and videos. This setup eliminates the need for a dedicated web server, as S3 can serve the content directly via HTTP URLs. It offers a cost-effective solution with high performance, scalability, and availability.
3. **Data Store for Computation and Analytics**:
    - Amazon S3 is suitable for large-scale analytics and computation tasks, such as financial transaction analysis and media transcoding. It can efficiently handle multiple concurrent transactions, making it a vital component in data processing workflows.
4. **Example Workflow**:
    - A compute cluster (like Amazon EC2 or EMR) extracts raw data from S3, processes it, and stores the transformed data back in another S3 bucket. Analytics tools, such as Amazon QuickSight, can then analyze this data for insights.
5. **Backup and Archiving**:
    - Due to its durability and scalability, Amazon S3 is effective for data backup and archival purposes. Data from on-premises data centers and EC2 instances can be backed up to S3 buckets.
    - **Cross-Region Replication**: To enhance durability, S3 supports cross-Region replication, automatically copying objects from one bucket to others in different Regions.
    - **Long-term Storage Options**: Users can transition long-term data from S3 standard storage to Amazon S3 Glacier for cost-effective archival.

Key Benefits of Amazon S3

- **Durability**: Ensures data integrity with high durability ratings (11 nines).
- **Availability**: Offers 99.99% availability, ensuring quick access to data.
- **High Performance**: Capable of handling thousands of transactions per second.

Summary of S3's Capabilities

Amazon S3 serves a broad range of needs, such as:

- Storing and distributing media files (videos, photos, music).
- Hosting static content.
- Supporting data processing for analytics.
- Providing backup and archival solutions.

This versatility makes Amazon S3 a foundational service for many cloud-based applications and workflows.

Section 3 Move data - Amazon S3

Amazon S3 Overview

- **Object Storage**: Amazon S3 allows for an unlimited number of objects to be stored in a bucket. Objects are automatically encrypted during upload using server-side encryption (SSE-S3).
- **Upload Methods**: Objects can be uploaded via:
    - **AWS Management Console**: A user-friendly interface that allows drag-and-drop uploads (max 160 GB).
    - **AWS Command Line Interface (CLI)**: Command-line operations for bulk uploads or downloads.
    - **AWS SDKs**: Programmatic uploads using various supported programming languages.
    - **Amazon S3 REST API**: For direct PUT requests to upload data.
- **Multipart Upload**: For files over 100 MB, multipart uploads improve throughput, allow parts to be uploaded independently, and support pause/resume functionality.

Key Features of Amazon S3

- **Transfer Acceleration**: Enhances upload speed over long distances by routing data through globally distributed CloudFront edge locations. It can significantly improve upload times for large files.
- **Use Cases**:
    - **Media Hosting**: Storing and distributing videos, photos, and music.

    - **Static Website Hosting**: Hosting static HTML sites without needing a dedicated server.
    - **Data Store for Analytics**: Supporting large-scale analytics and computations.
    - **Backup and Archiving**: Reliable solutions for data backup with options like cross-Region replication and integration with Amazon S3 Glacier for long-term storage.

AWS Transfer Family

- **Overview**: A fully managed service for transferring files to/from Amazon S3 and Amazon EFS using protocols like SFTP, FTPS, FTP, and AS2.
- **Benefits**:
    - Scales in real-time without the need for infrastructure changes.
    - Seamless integration with AWS services for analytics and processing.
    - Serverless file transfer workflows to automate uploads.
    - Pay-as-you-go pricing with no upfront costs.

Common Use Cases for Transfer Family

- **With Amazon S3**:
    - Data lakes for third-party uploads.
    - Subscription-based data distribution.
    - Internal data transfers.
- **With Amazon EFS**:
    - Data distribution and supply chain management.
    - Content management and web-serving applications.

Key Takeaways

- Utilize various methods for uploading objects to S3, including the Management Console, CLI, SDKs, and REST API.
- Implement multipart upload for large files to optimize transfer efficiency.
- Leverage Transfer Acceleration for faster uploads over long distances and use Transfer Family for secure file transfers into/out of AWS storage.

This summary encapsulates the various functionalities and use cases of Amazon S3 and AWS Transfer Family, providing a clear understanding of their capabilities and applications.

Section 4 Storing Data s3

Amazon S3 Versioning

1. **Deleting Objects**:
    - When an object is deleted in a version-enabled bucket, all versions remain in the bucket, and a **delete marker** is inserted. For example, if `photo.gif` is deleted, versions can still be accessed using their IDs (e.g., `121212` or `111111`).
2. **Retrieving Versions**:
    - Requests for an object key return the most recent version. If the latest version is a delete marker, a 404 error is returned, indicating no object found.
    - Specific versions can be retrieved by using their version ID. For example, a GET request for `photo.gif` with version ID `121245` will return that specific version.
3. **Permanently Deleting Objects**:
    - Only the owner of the bucket can permanently delete a specific object version using its version ID. This action does not add a delete marker, making the version irrecoverable.

Cross-Origin Resource Sharing (CORS)

- CORS allows client web applications loaded in one domain to interact with resources in a different domain. By creating a **CORS configuration**, you specify:
    - Allowed origins.
    - Supported operations (e.g., GET requests).
- This is useful for scenarios such as hosting web fonts in S3 buckets that need to be accessed by webpages from different domains.

Amazon S3 Data Consistency Model

- Amazon S3 provides **strong read-after-write consistency** for all GET, LIST, PUT, and DELETE operations on objects, simplifying the migration of on-premises analytics workloads.
- If a PUT request is successful, any subsequent GET or LIST request will return the most recently written data.
- While object operations are strongly consistent, bucket configurations may have eventual consistency; for instance, a deleted bucket might still appear in listings shortly after deletion.

Key Takeaways

- **S3 Standard Storage** is ideal for cloud applications, dynamic websites, and big data analytics.
- Implementing an **S3 lifecycle policy** enables automatic transfer of data between storage classes without application changes.
- **Versioning** helps recover from unintended deletions and application failures.
- Understanding **CORS** is essential for building web applications that interact with S3 resources.
- The **data consistency model** in S3 supports seamless migration of existing workloads by eliminating the need for additional infrastructure changes.

This summary captures the core ideas and functionality of Amazon S3 versioning, CORS, and data consistency

Section 5

Amazon S3 Tools for Protection

1. **Block Public Access**: Prevents public access to buckets.
2. **IAM Policies**: Authenticate users for specific bucket/object access.
3. **Bucket Policies**: Define access rules for buckets or objects, useful for cross-account access.
4. **Access Control Lists (ACLs)**: Set specific rules for bucket/object access (less commonly used).
5. **S3 Access Points**: Customized network endpoints for applications.
6. **Preassigned URLs**: Provide temporary access to objects.
7. **AWS Trusted Advisor**: Checks bucket permissions for global access.

Approaches to Access Configuration

1. **Default Security Settings**: All new buckets and objects are private.
2. **Controlled Access**: Specific users granted access while others are denied.
3. **Public Access**: Not recommended for sensitive data; suitable for static websites only.

Considerations for Choosing a Region

- **Data Privacy Laws**: Compliance with regulations in the selected region.
- **User Proximity**: Lower latency improves user experience.
- **Service Availability**: Some AWS services may not be available in all regions.
- **Cost-Effectiveness**: Costs vary by region; consider overall expenses.

Amazon S3 Inventory

- Helps manage storage, audit, and report on object statuses.
- Provides scheduled reports in various formats (CSV, ORC, Parquet).

Cost Structure

- **Pay-as-you-go**: Charges based on stored data, requests made, and data transfer.
- **Free Tier**: Offers 5 GB of storage and limited requests for new AWS customers.

Example Activity

- **Controlled Access Setup**: Create folders for content and IAM policies for employee-level access.

Key Takeaways

- S3 buckets are private by default; controlled access is crucial for security.
- SSE-S3 is the standard encryption method.
- Consider multiple factors when selecting a region, including compliance and cost.
- Understand the cost structure to manage expenses effectively.

This summary encapsulates the essential features, configurations, and considerations related to Amazon S3.

Section 6

key points from the AWS Training and Certification Module on adding a storage layer with Amazon S3, particularly in the context of the AWS Well-Architected Framework principles:

AWS Well-Architected Framework for Storage

1. **Security**
    - **Data Protection**:
        - Enforce encryption at rest to ensure confidentiality.
        - Implement access control to limit data exposure.
        - Use regular backups and versioning to safeguard against data loss.
    - **Best Practices**:
        - S3 buckets and objects are encrypted by default.
        - Access is private by default; explicit permissions are required.
        - Versioning helps protect against accidental deletions or overwrites.
2. **Reliability**
    - **Failure Management**:
        - Utilize multi-Availability Zone (AZ) deployments for high availability.

        - Amazon S3 offers 99.999999999% (11 nines) durability and 99.99% (4 nines) availability.
        - Data is redundantly stored across multiple AZs, with regular integrity checks.
3. **Performance Efficiency**
    - **Architecture Selection**:
        - Understand available services and choose based on workload needs.
        - Amazon S3 is suitable for storing massive amounts of unstructured data.
        - Performance can be enhanced with S3 Transfer Acceleration and multipart uploads.
4. **Cost Optimization**
    - **Cost-Effective Resources**:
        - Conduct cost analyses based on usage patterns over time.
        - Utilize S3's various storage classes and lifecycle rules to manage costs effectively.
        - S3 Intelligent-Tiering automatically optimizes storage costs based on access patterns.

Key Takeaways

- Amazon S3 supports data protection through default encryption and access restrictions.
- Its architecture is designed for efficiency and can handle unstructured data at scale.
- Cost management features, such as lifecycle policies and intelligent tiering, aid in long-term savings.
- S3's reliability features, including durability and availability, ensure data resilience.

This summary encapsulates how the AWS Well-Architected Framework principles are applied to storage solutions using Amazon S3, focusing on security, reliability, performance, and cost-effectiveness.

Module check

1. Which attributes are reasons to choose Amazon Elastic Compute Cloud (Amazon EC2)? (Select TWO.)

- [x] Ability to run any type of workload
- [ ] Ability to run serverless applications
- [ ] Amazon Web Services (AWS) management of operating system patches
- [ ] AWS management of operating system security
- [x] Complete control of computing resources

2. What are the benefits of using an Amazon Machine Image (AMI)? (Select THREE.)

- [x] Selling or sharing software solutions packaged as an AMI
- [x] Using an AMI as a server backup for Amazon Elastic Compute Cloud (Amazon EC2) instances
- [ ] Automating security group settings for instances
- [x] Launching instances with the same configuration
- [ ] Migrating data from on-premises to Amazon EC2 instances

3. A system administrator must change the instance types of multiple running Amazon Elastic Compute Cloud (Amazon EC2) instances. The instances were launched with a mix of Amazon Elastic Block Store (Amazon EBS)-backed Amazon Machine Images (AMIs) and instance store-backed AMIs. Which method is a valid way to change the instance type?

- ( ) Change the instance type of an Amazon EBS-backed instance without stopping it.
- ( ) Change the instance type of an instance store-backed instance without stopping it.
- (•) Stop an Amazon EBS-backed instance , change its instance type, and start the instance.

4. A workload requires high read/write access to large local datasets. Which instance types would perform best for this workload? (Select TWO.)

- [ ] General purpose
- [ ] Compute optimized
- [x] Memory optimized
- [ ] Accelerated computing
- [x] Storage optimized

5. An application requires the media access control (MAC) address of the host Amazon Elastic Compute Cloud (Amazon EC2) instance. The architecture uses an AWS Auto Scaling group to dynamically launch and terminate instances. Which way is best for the application to obtain the MAC address?

- ( ) Write the MAC address in the application configuration file of each instance.
- ( ) Include the MAC address in the Amazon Machine Image (AMI) that is used to launch all of the instances in the AWS Auto Scaling group.
- ( ) Include the MAC address in a custom AMI for each instance in the AWS Auto Scaling group.
- (•) Use the user data of each instance to access the MAC address through the instance metadata.

KEYBOARD NAV

6. A transactional workload on an Amazon Elastic Compute Cloud (Amazon EC2) instance performs high amounts of frequent read and write operations. Which Amazon Elastic Block Store (Amazon EBS) volume type is best for this workload?

- ( ) General purpose solid state drive (SSD)
- ( ) Cold hard disk drive (HDD)
- (•) Provisioned IOPS solid state drive (SSD)
- ( ) Throughput optimized hard disk drive (HDD)

7. It is possible to create an NFS share on an Amazon EBS-backed Linux instance by installing and configuring an NFS server on the instance. In this way, multiple Linux systems can share the file system of that instance. Which advantages does Amazon Elastic File System (Amazon EFS) provide, compared to this solution? (Select TWO.)

- [ ] Strong consistency
- [x] Automatic scaling
- [x] High availability
- [ ] File locking
- [ ] No need for backups

8. Which feature does Amazon FSx for Windows File Server provide?

- (•) Fully managed Windows file servers
- ( ) Microsoft Active Directory (Microsoft AD) server for Windows file servers
- ( ) Backup solution for on-premises Windows file servers
- ( ) Amazon management agent for Windows file serve

9. Which descriptions of Amazon Elastic Compute Cloud (Amazon EC2) pricing options are correct? (Select TWO.)

- [x] On-Demand Instances enable you to pay for compute capacity by usage time, with no long-term commitments.
- [ ] Reserved Instances are physical servers that are reserved exclusively for your use.
- [ ] Savings Plans are budgeting tools that help you manage Amazon EC2 costs.
- [ ] Dedicated Hosts are servers that are dedicated to one purpose, such as a firewall.
- [x] Spot Instances offer spare compute capacity at discounted prices, and can be interrupted.

10. A company has three high-performance computing instances that must communicate with each other. The company would like to achieve maximum network performance between the instances. The most important requirement is that these systems do not share the same rack. Which placement strategy should they use?

- ( ) Partition
- (•) Spread
- ( ) Default
- ( ) Cluster

Section 1 Adding a Compute Layer Using Amazon EC2

AWS compute service categories, Amazon EC2, and the process of provisioning an EC2 instance:

1. Compute Service Categories Differentiators

- **Management Responsibilities**:
  - **VMs, Containers, VPS**: Offer more control over infrastructure, allowing for high customization of the environment.
  - **PaaS and Serverless**: Abstract infrastructure management, letting developers focus primarily on application code and functionality.
- **Scalability**:
  - **VMs, Containers, VPS**: Provide flexibility to scale manually or automatically based on workload demands.
  - **PaaS and Serverless**: Automatically manage scaling in response to traffic, reducing manual intervention.
- **Suitability for Workloads**:
  - **VMs, Containers, VPS**: Ideal for workloads requiring specific configurations, high control, or custom applications.
  - **PaaS and Serverless**: Best suited for applications with variable workloads that need rapid deployment and minimal management overhead.

2. Amazon EC2 Overview

- **Definition**: Amazon EC2 (Elastic Compute Cloud) provides virtual servers (EC2 instances) that can be provisioned within minutes.
- **Capabilities**:
  - Automatic scaling of capacity based on demand.
  - Pay-as-you-go pricing model for cost efficiency.

3. EC2 Instance Components

- **Physical Host**: The underlying server where EC2 instances run.
- **Availability Zones**: Geographic locations that contain multiple data centers to ensure redundancy and fault tolerance.
- **VMs**: Virtual machines that operate with their own OS and applications.
- **Operating Systems**: Various OS options, including Amazon Linux, Microsoft Windows, and macOS.
- **Hypervisor**: Software that manages VM access to physical resources like CPU, memory, and storage.

4. Storage Options

- **Instance Store**: Temporary storage physically attached to the host.
- **Amazon EBS (Elastic Block Store)**: Persistent storage that retains data even when instances are stopped.

5. Network Connectivity

- EC2 instances can connect to other AWS resources, allowing configuration of network access to balance security and accessibility.

6. Use Cases for Amazon EC2

- Complete control over computing resources.
- Cost optimization options through On-Demand, Reserved, and Spot Instances.
- Hosting a variety of applications, from simple websites to complex enterprise applications.

7. Steps for Provisioning an EC2 Instance

1. **Choose an Amazon Machine Image (AMI)**: The template for launching instances.
2. **Select Instance Type**: Based on CPU, memory, and storage requirements.
3. **Specify a Key Pair**: For secure SSH or RDP access.
4. **Network Placement**: Define how the instance connects to your network.
5. **Assign a Security Group**: Controls inbound and outbound traffic.
6. **Specify Storage Options**: Choose between instance store and EBS volumes.
7. **Attach an IAM Role**: For AWS service API calls.
8. **User Data**: Automate configurations and installations upon launch.

8. Key Takeaways

- Amazon EC2 allows running VMs with complete control over resources.
- Essential to choose an AMI and instance type during provisioning.
- Configuration settings include network, security, storage, and user data.

This summary encapsulates the differentiators of AWS compute service categories, the role of EC2, and the key steps for provisioning instances, emphasizing the flexibility and control that AWS provides for various workloads.

Section 2 Choosing an AMI to launch an EC2 Instance

1. Definition and Purpose of AMIs

- **AMIs** provide the necessary information to launch an Amazon EC2 instance, including:
    - **Template for the root volume:** Contains the guest operating system (OS) and potentially other software.
    - **Launch permissions:** Control access to the AMI.
    - **Block device mappings:** Specify additional storage volumes to attach to the instance at launch.

2. Benefits of Using AMIs

- **Repeatability:** AMIs allow efficient and precise instance launches.
- **Reusability:** Instances launched from the same AMI are identically configured, making it easy to create clusters or recreate environments.
- **Recoverability:** AMIs serve as restorable backups; a new instance can be launched from the same AMI if the original instance fails.

3. Choosing an AMI

When selecting an AMI, consider the following characteristics:

- **Region:** Each AMI exists in a specific AWS Region.
- **Operating System:** Options include various Linux distributions and Microsoft Windows.
- **Storage Type:** AMIs can be Amazon EBS-backed (persistent storage) or instance store-backed (temporary storage).
- **Architecture:** Options include 32-bit or 64-bit, x86 or ARM instruction sets.
- **Virtualization Type:** Use HVM AMIs for better performance.

4. Sources of AMIs

- **Quick Start:** AMIs provided by AWS.
- **My AMIs:** Custom AMIs created by the user.
- **AWS Marketplace:** Pre-configured AMIs from third parties.
- **Community AMIs:** AMIs shared by others, used at your own risk.

5. Instance Store-Backed vs. EBS-Backed AMIs

- **Boot Time:** EBS-backed instances boot faster.
- **Root Device Size:** EBS-backed instances can have a maximum root device size of 16 TiB; instance store-backed instances are limited to 10 GiB.
- **Stopping and Changing Instance Type:** EBS-backed instances can be stopped and have their type changed; instance store-backed instances cannot.
- **Cost Structure:** Different charging mechanisms for EBS and instance store-backed instances.

6. Instance Lifecycle

- An instance transitions through states: **pending**, **running**, **stopping**, **stopped**, and **terminated**. EBS-backed instances can be stopped and restarted, while instance store-backed instances cannot.

7. Creating New AMIs

- AMIs can be created from existing EC2 instances, allowing for customized configurations to be captured and reused.

8. EC2 Image Builder

- Automates the creation, management, and deployment of up-to-date AMIs, providing a graphical interface, version control, and security validations.

Key Takeaways

- AMIs are crucial for launching EC2 instances efficiently and consistently.
- Benefits include repeatability, reusability, and recoverability.
- Consider performance and source when choosing an AMI.

This overview captures the essential information about AMIs, their benefits, usage, and lifecycle in the context of Amazon EC2.

Section 3

Here's a summary of the key points regarding Amazon EC2 instance types and their configuration:

1. EC2 Instance Types Overview

- **Definition**: EC2 instance types define configurations of CPU, memory, storage, and network performance.
- **Importance**: Choosing the right instance type is crucial for matching workload performance and cost requirements.

2. Instance Type Configuration

- **Example Instance Types**:
  - **m5d.large**: 2 vCPUs, 24 GiB memory, 1 x 50 NVMe SSD, Up to 10 Gbps network performance.
  - **m5d.xlarge**: 4 vCPUs, 48 GiB memory, 1 x 100 NVMe SSD, Up to 10 Gbps network performance.
  - **m5d.8xlarge**: 32 vCPUs, 128 GiB memory, 2 x 600 NVMe SSD, 10 Gbps network performance.
- **Network Performance**: Generally more static across instance sizes; m5d.large and m5d.xlarge both support up to 10 Gbps, while m5d.8xlarge provides a consistent 10 Gbps.

3. Instance Type Naming Convention

- **Components**:
  - **Family**: e.g., `c` for compute optimized.
  - **Generation**: Indicated by a number (higher means newer and generally better).
  - **Processor Family**: Optional indicator (e.g., `g` for AWS Graviton).
  - **Size**: Indicates performance category (e.g., `xlarge`).

4. Suitability for Workloads

- **General Purpose**: Suitable for web/app servers, gaming, etc. (e.g., M5, T3).
- **Compute Optimized**: Best for compute-bound applications (e.g., C5).
- **Storage Optimized**: Designed for high-performance databases (e.g., I3).
- **Memory Optimized**: For applications with large datasets (e.g., R5).
- **Accelerated Computing**: For machine learning and AI (e.g., P3).
- **High Performance Computing (HPC)**: For complex simulations (e.g., Hpc7).

5. Choosing the Right Instance Type

- **Considerations**: Analyze workload requirements and cost. Start with a smaller instance and scale as necessary.
- **Resources**: Use the EC2 console's Instance Types page and AWS Compute Optimizer for recommendations.

6. AWS Compute Optimizer

- **Functionality**: Recommends optimal instance types and configurations based on workload patterns.
- **Classification**: Results are classified as Under-provisioned, Over-provisioned, Optimized, or None.

This summary captures the essential details regarding EC2 instance types, their configurations, naming conventions, suitability for different workloads, and strategies for selecting the right type based on performance and cost considerations.

Section 4

Shared File System Options for EC2 Instances

1. **Amazon EBS**:
    - Attaches to a single instance at a time.
    - Not suitable for sharing data among multiple instances.
2. **Amazon S3**:
    - An object store, not a block store.
    - Suitable for storing files, but changes overwrite entire files rather than individual blocks.
    - Not ideal for applications requiring high-throughput and consistent read/write access.
3. **Amazon EFS (Elastic File System)**:
    - Fully managed service for Linux workloads.
    - Scales automatically from gigabytes to petabytes.
    - Supports Network File System (NFS) protocols.
    - Allows multiple EC2 instances to access the file system simultaneously.
    - Common use cases include home directories, enterprise applications, application testing, database backups, web serving, media workflows, and big data analytics.
4. **Amazon FSx for Windows File Server**:
    - Fully managed shared file system for Microsoft Windows EC2 instances.
    - Supports NTFS and SMB protocols.
    - Integrates with Microsoft Active Directory for permissions and access control.
    - Suitable for home directories, lift-and-shift applications, media workflows, data analytics, web serving, and software development environments.

Key Takeaways

- **Storage Options**:
    - For a root volume: Use instance store or SSD-backed Amazon EBS.
    - For data volume serving a single instance: Use instance store or Amazon EBS.
    - For data volume serving multiple Linux instances: Use Amazon EFS.
    - For data volume serving multiple Windows instances: Use Amazon FSx for Windows File Server.

This summary provides a clear overview of the available shared file system options for EC2 instances, focusing on their suitability and primary use cases.

Section 5

Shared File System Options for EC2

1. **Amazon EBS (Elastic Block Store)**:
    - Attaches to one instance only; not suitable for sharing.
2. **Amazon S3 (Simple Storage Service)**:
    - Object storage; not ideal for high-throughput applications due to file overwriting behavior.
3. **Amazon EFS (Elastic File System)**:
    - Fully managed service for Linux workloads.
    - Supports NFS protocols and allows multiple EC2 instances to access the same file system.
    - Ideal for applications like web serving, media workflows, and big data analytics.
4. **Amazon FSx for Windows File Server**:
    - Fully managed shared file system for Windows instances.
    - Supports NTFS and SMB protocols; integrates with Active Directory.
    - Suitable for home directories, lift-and-shift applications, and media workflows.

Key Takeaways

- **Root Volume**: Use instance store or Amazon EBS.
- **Single Instance Data Volume**: Use instance store or Amazon EBS.
- **Multiple Linux Instances**: Use Amazon EFS.
- **Multiple Windows Instances**: Use Amazon FSx for Windows File Server.

This summary highlights the key options and considerations for shared file systems in Amazon EC2 environments.

Section 6

Amazon EC2 Pricing Models

1. **On-Demand Instances**:
    - Pay per second or hour without long-term commitments.
    - **Recommended for**: Spiky workloads, experimentation, short-term applications.
2. **Reserved Instances**:
    - Commit to a 1-year or 3-year term for significant discounts on On-Demand prices.
    - **Recommended for**: Committed workloads, steady-state applications.
3. **Savings Plans**:
    - Flexible pricing model that offers discounts in exchange for a commitment to a consistent usage amount (in $/hour) over 1 or 3 years.
    - **Types**:
        - **Compute Savings Plans**: Most flexibility, applies to various services.
        - **EC2 Instance Savings Plans**: Less flexible, specific to instance family and region.
    - **Recommended for**: All EC2 workloads, particularly those needing flexibility.
4. **Spot Instances**:
    - Bid on unused EC2 capacity for substantial savings.
    - **Recommended for**: Fault-tolerant, flexible, or stateless workloads.
    - Can be interrupted with a 2-minute notice; billed in increments (1-second for Amazon Linux/Ubuntu, 1-hour for others).
5. **Capacity Reservations**:
    - Reserve compute capacity in a specific Availability Zone to ensure availability.
    - **Types**:
        - **On-Demand Capacity Reservations**: For workloads needing guaranteed capacity.
        - **EC2 Capacity Blocks for ML**: Reserve GPU instances for machine learning workloads.
6. **Dedicated Options**:
    - **Dedicated Instances**: Run on dedicated hardware isolated from other accounts.
    - **Dedicated Hosts**: Fully dedicated physical servers, useful for licensing and compliance.
    - **Recommended for**: Workloads requiring dedicated hardware or specific software licenses.

Cost Optimization Guidelines

- **Combine Models**: Use a mix of purchasing options to optimize costs.
    - **Reserved Instances or Savings Plans** for steady-state workloads.
    - **On-Demand Instances** for stateful spiky workloads.
    - **Spot Instances** for fault-tolerant, flexible workloads.

Key Takeaways

- **Models Include**: On-Demand, Reserved, Savings Plans, Spot Instances, and Dedicated Hosts.
- **Billing**: Per-second billing is available for specific models.
- **Optimization**: A combination of the pricing models helps reduce overall EC2 costs effectively.

This summary encapsulates the essential points regarding Amazon EC2's pricing strategies and recommendations for optimizing costs based on usage patterns

Section 6

AWS Well-Architected Framework Overview

The AWS Well-Architected Framework consists of six pillars, each addressing critical areas in cloud architecture. This summary emphasizes security, performance efficiency, cost optimization, and sustainability.

Key Principles

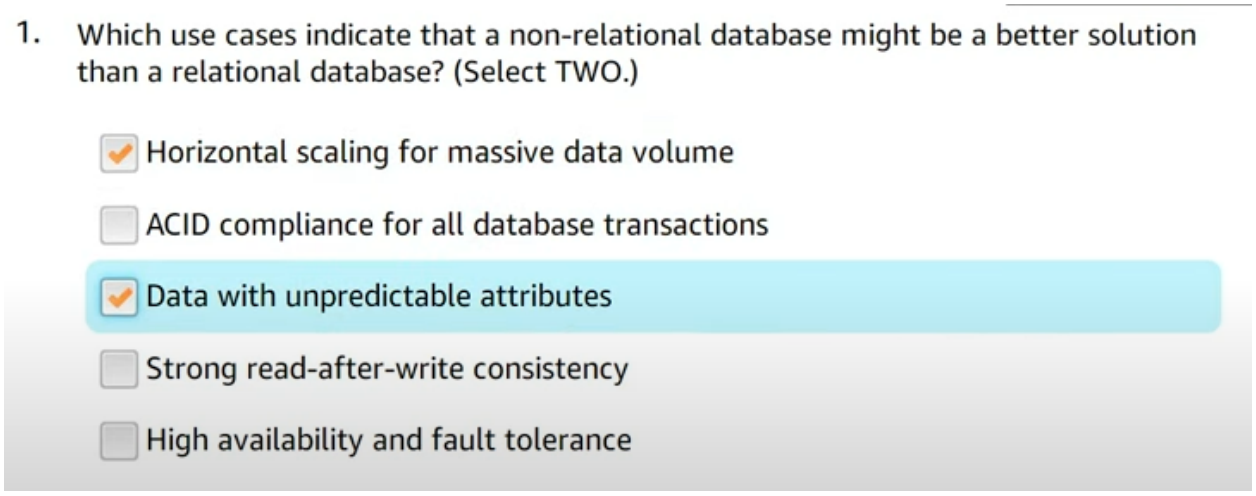
1. **Automate Compute Protection (Security)**
    - Implement automation for protective measures like vulnerability management and resource management to reduce human error and enhance security.
    - Utilize tools such as EC2 Image Builder and user data scripts to strengthen defenses against threats.
2. **Control Traffic at All Layers (Security)**
    - Establish robust traffic controls across your network, focusing on secure operation of workloads.
    - Use VPCs to manage network topology, employing security groups to regulate inbound and outbound traffic.
3. **Scale the Best Compute Options (Performance Efficiency)**
    - Choose the most suitable compute options based on workload requirements to optimize performance and resource efficiency.
    - Continuously evaluate compute choices to match application design and usage patterns.
4. **Configure and Right-Size Compute Resources (Performance Efficiency)**
    - Ensure compute resources are properly sized to meet performance requirements, preventing over- or under-utilization.
    - Optimize configurations to enhance customer experience while minimizing costs.
5. **Select the Correct Resource Type, Size, and Number (Cost Optimization)**
    - Choose appropriate resource types and sizes to meet technical needs at the lowest cost.

    - Engage in right-sizing as an iterative process influenced by usage patterns and external factors.
6. **Select the Best Pricing Model (Cost Optimization)**
    - Analyze workload requirements to determine the most cost-effective pricing models, including On-Demand, Reserved, Spot instances, and Savings Plans.
7. **Use the Minimum Amount of Hardware (Sustainability)**
    - Right-size resources to minimize environmental impact while maintaining performance.
    - Take advantage of AWS's flexible resource modification capabilities.
8. **Use Instance Types with the Least Impact (Sustainability)**
    - Monitor and adopt new instance types for enhanced energy efficiency, particularly for specific workloads.
9. **Use Managed Services (Sustainability)**
    - Shift operational responsibilities to AWS through managed services, allowing teams to focus on innovation while AWS maintains efficiency.

Key Takeaways

- Automate security measures and compute protection.
- Optimize and right-size compute resources to meet workload demands efficiently.
- Select cost-effective resource types and pricing models.
- Focus on sustainability by minimizing hardware usage and utilizing managed services.

Module Checks



2. Which statement that compares a database service that Amazon Web Services (AWS) manages with a database on an Amazon Elastic Compute Cloud (Amazon EC2) instance is true?

- ○ You do not need to configure backups for a database on a managed database service.
- ◉ AWS manages DB patches for a database on a managed database service.
- ○ AWS manages operating system (OS) patches for a database on an EC2 instance.
- ○ You do not need to configure backups for a database on an EC2 instance.

3. Which examples are good use cases for Amazon Relational Database Service (Amazon RDS)? (Select THREE.)

- ☐ Thousands of distributed concurrent writes per second
- ☑ An application that requires the database to enforce syntax rules
- ☑ An application that requires complex joins of data
- ☐ A petabyte-scale data warehouse
- ☑ Running a Microsoft SQL Server in Amazon Web Services (AWS)
- ☐ Database for serverless architectures

4. A small company is deciding which service to use for an enrollment system for their online training website. Choices are MySQL on Amazon Elastic Compute Cloud (Amazon EC2), MySQL in Amazon Relational Database Service (Amazon RDS), and Amazon DynamoDB. Which combination of use cases suggests using Amazon RDS? (Select THREE.)

- [ ] Data and transactions must be encrypted to protect personal information.
- [x] The data is highly structured.
- [x] Student, course, and registration data are stored in many different tables.
- [ ] The enrollment system must be highly available.
- [x] The company doesn't want to manage database patches.

5. Which scenarios are good use cases for Amazon DynamoDB? (Select THREE.)

- [x] Database for serverless architectures
- [x] Applications that require ACID transactions
- [ ] Applications that combine data from many tables
- [ ] Graph database to trace relationships between entities
- [x] Document database for JavaScript Object Notation (JSON)-based documents
- [ ] Binary large object (BLOB) storage

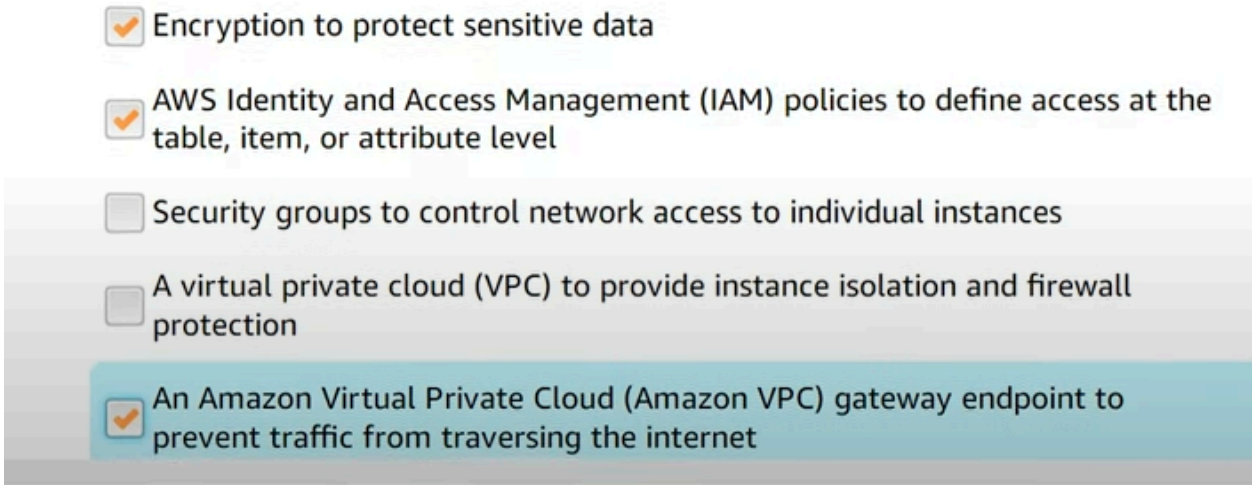
6. A small game company is designing an online game, where thousands of players can create their own in-game objects. The current design uses a MySQL database in Amazon Relational Database Service (Amazon RDS) to store data for player-created objects. Which use cases suggest that DynamoDB might be a better solution? (Select TWO.)

- [ ] A set of common attributes that all player-created objects have
- [x] Unpredictable attributes for player-created objects
- [x] Large number of player-created objects, each with different attributes
- [ ] Quick search and retrieval of player-created objects
- [ ] High amount of read activity on player-created objects

7. Which techniques should you use to secure an Amazon Relational Database Service (Amazon RDS) database? (Select THREE.)

- [ ] AWS Identity and Access Management (IAM) policies to define access at the table, row, and column levels
- [x] Security groups to control network access to individual instances
- [ ] An Amazon Virtual Private Cloud (Amazon VPC) gateway endpoint to prevent traffic from traversing the internet
- [ ] A virtual private gateway (VGW) to filter traffic from restricted networks
- [x] A virtual private cloud (VPC) to provide instance isolation and firewall

8. Which techniques should you use to secure Amazon DynamoDB? (Select THREE.)



9. A company wants to migrate their on-premises Oracle database to Amazon Aurora MySQL. Which process describes the high-level steps?

- Use AWS Database Migration Service (AWS DMS) to migrate from the Oracle database to Amazon Aurora MySQL..
- Use AWS Database Migration Service (AWS DMS) to migrate the data, and then use AWS Schema Conversion Tool to convert the schema.
- **Use AWS Schema Conversion Tool to convert the schema, and then use AWS Database Migration Service (AWS DMS) to migrate the data.**
- Use AWS Schema Conversion Tool to synchronously convert the schema and migrate the data.

9. A company wants to migrate their on-premises Oracle database to Amazon Aurora MySQL. Which process describes the high-level steps?

- Use AWS Database Migration Service (AWS DMS) to migrate from the Oracle database to Amazon Aurora MySQL..
- Use AWS Database Migration Service (AWS DMS) to migrate the data, and then use AWS Schema Conversion Tool to convert the schema.
- **Use AWS Schema Conversion Tool to convert the schema, and then use AWS Database Migration Service (AWS DMS) to migrate the data.**
- Use AWS Schema Conversion Tool to synchronously convert the schema and migrate the data.

KEYBOARD NAVIGATION

10. You must perform a heterogeneous migration from your on-premises facility to a database in a virtual private cloud (VPC). You will use AWS Snowball Edge and AWS Database Migration Service (AWS DMS). At which point do you use AWS Schema Conversion Tool (AWS SCT)?

- (●) At the start, to extract data from the source database into the Snowball Edge, before shipping the device
- ( ) After extracting the data from the source database by using AWS DMS, but before shipping the Snowball Edge
- ( ) After the data is in the VPC, but before using AWS DMS to load the data into the target database
- ( ) After using AWS DMS to load the data into in the target database in the VPC

Section 1

Key Considerations for Database Selection

1. **Scalability**:
    - Assess the required throughput and ensure the database can scale efficiently without downtime.
    - Avoid underprovisioning (leading to application failures) or overprovisioning (increased costs).
2. **Storage Requirements**:
    - Determine the necessary storage capacity (gigabytes, terabytes, or petabytes) based on your data needs.
    - Different architectures support varying maximum data capacities.
3. **Data Characteristics**:
    - Understand your data model (relational, structured, semi-structured, etc.) and access patterns.
    - Consider latency needs and specific data record sizes.
4. **Durability and Availability**:
    - Define the level of data durability and availability required.
    - Consider regulatory obligations for data residency and compliance.

Types of Databases

- **Relational Databases**:
    - Structure: Tabular form (columns and rows), with strict schema rules.
    - Benefits: Data integrity, ease of use, and SQL compatibility.
    - Use Case: Ideal for online transactional processing and structured data.
- **Non-Relational Databases (NoSQL)**:
    - Structure: Varied models (key-value, document, graph), with flexible schemas.
    - Benefits: Scalability, high performance, and suitable for semi-structured/unstructured data.
    - Use Case: Effective for caching, JSON storage, and low-latency access.

AWS Database Options

- **Relational Databases**:
    - Amazon RDS, which offers multiple familiar database engines (e.g., Aurora, MySQL, PostgreSQL).
- **Non-Relational Databases**:
    - Amazon DynamoDB (key-value), Amazon Neptune (graph), Amazon ElastiCache (in-memory).

Managed Database Services

- Managed services like Amazon RDS reduce the administrative burden by handling tasks such as backups, scaling, and maintenance.
- As you move to managed services, your responsibilities shift primarily to optimizing queries and application efficiency.

Database Capacity Planning

- **Process**:
    1. Analyze current capacity.
    2. Predict future requirements.
    3. Decide on horizontal or vertical scaling.
- **Vertical Scaling**: Increasing resources of existing servers; may require downtime.
- **Horizontal Scaling**: Adding more servers to handle increased load without downtime.

Key Takeaways

- Consider scalability, storage, data characteristics, costs, and durability when selecting a database.
- Relational databases are suited for structured data and transactional applications, while non-relational databases offer flexibility for diverse data types.
- AWS managed services minimize operational responsibilities, allowing focus on application optimization.
-

Section 2

ey Considerations for Database Selection

1. **Scalability**:
    - Assess the required throughput and ensure the database can scale efficiently without downtime.
    - Avoid underprovisioning (leading to application failures) or overprovisioning (increased costs).
2. **Storage Requirements**:
    - Determine the necessary storage capacity (gigabytes, terabytes, or petabytes) based on your data needs.
    - Different architectures support varying maximum data capacities.
3. **Data Characteristics**:
    - Understand your data model (relational, structured, semi-structured, etc.) and access patterns.
    - Consider latency needs and specific data record sizes.
4. **Durability and Availability**:
    - Define the level of data durability and availability required.
    - Consider regulatory obligations for data residency and compliance.

Types of Databases

- **Relational Databases**:
    - Structure: Tabular form (columns and rows), with strict schema rules.
    - Benefits: Data integrity, ease of use, and SQL compatibility.
    - Use Case: Ideal for online transactional processing and structured data.
- **Non-Relational Databases (NoSQL)**:
    - Structure: Varied models (key-value, document, graph), with flexible schemas.
    - Benefits: Scalability, high performance, and suitable for semi-structured/unstructured data.
    - Use Case: Effective for caching, JSON storage, and low-latency access.

AWS Database Options

- **Relational Databases**:
    - Amazon RDS, which offers multiple familiar database engines (e.g., Aurora, MySQL, PostgreSQL).
- **Non-Relational Databases**:
    - Amazon DynamoDB (key-value), Amazon Neptune (graph), Amazon ElastiCache (in-memory).

Managed Database Services

- Managed services like Amazon RDS reduce the administrative burden by handling tasks such as backups, scaling, and maintenance.
- As you move to managed services, your responsibilities shift primarily to optimizing queries and application efficiency.

Database Capacity Planning

- **Process**:
    1. Analyze current capacity.
    2. Predict future requirements.
    3. Decide on horizontal or vertical scaling.
- **Vertical Scaling**: Increasing resources of existing servers; may require downtime.
- **Horizontal Scaling**: Adding more servers to handle increased load without downtime.

Key Takeaways

- Consider scalability, storage, data characteristics, costs, and durability when selecting a database.
- Relational databases are suited for structured data and transactional applications, while non-relational databases offer flexibility for diverse data types.

- AWS managed services minimize operational responsibilities, allowing focus on application optimization.

Section 3

Key Database Considerations

1. **Scalability**:
    - Ensure the database can handle current and future throughput without downtime. Avoid under- or overprovisioning.
2. **Storage Requirements**:
    - Determine the necessary capacity (gigabytes, terabytes, petabytes) based on data needs.
3. **Data Characteristics**:
    - Understand the data model (relational, structured, semi-structured) and access patterns, including latency requirements.
4. **Durability and Availability**:
    - Define needed data durability and availability levels. Consider regulatory compliance for data residency.

Database Types

- **Relational Databases**:
    - Structured data in tables with strict schemas; ideal for online transactional processing.
- **Non-Relational Databases (NoSQL)**:
    - Flexible schemas and various models (key-value, document, graph); suitable for semi-structured and unstructured data.

AWS Database Options

- **Relational**: Amazon RDS (supports multiple engines like Aurora, MySQL).
- **Non-Relational**: Amazon DynamoDB (key-value), Amazon Neptune (graph), Amazon ElastiCache (in-memory).

Managed Database Services

- AWS manages tasks like backups and scaling, allowing focus on query optimization and application efficiency.

Database Capacity Planning

- Analyze current capacity, predict future needs, and choose between vertical (increasing server resources) or horizontal scaling (adding servers).

Key Takeaways

- Consider scalability, storage, data characteristics, costs, and durability in database selection.
- Use relational databases for structured data; non-relational databases for flexibility and diverse data types.

Section 3

features and structure of Amazon DynamoDB based on the provided information:

Key Features of Amazon DynamoDB

1. **Serverless Performance with Limitless Scalability**:
    - **Global and Local Secondary Indexes**: Enables flexible data access through alternate keys, allowing for lower write throughput and cost-effective performance.
    - **DynamoDB Streams**: Records item-level changes in near real-time, facilitating event-driven architectures.
    - **Global Tables**: Supports multi-region, multi-active data replication for fast local access, automatically scaling to accommodate workloads.
2. **Built-in Security and Reliability**:
    - **Data Encryption**: Automatically encrypts all customer data at rest.
    - **Point-in-Time Recovery (PITR)**: Protects data from accidental operations with continuous backups for up to 35 days.
    - **Fine-Grained Access Control**: Uses IAM for authentication and allows access control at the item and attribute level.

Data Structure in DynamoDB

- **Tables**: A table is a collection of data containing items, uniquely identifiable by their attributes.
- **Items**: An item consists of one or more attributes, similar to a row in a relational database.
- **Attributes**: Fundamental data elements consisting of key-value pairs.
- **Primary Keys**:
    - **Simple Primary Key**: Composed of a single attribute known as the partition key.
    - **Composite Primary Key**: Combines the partition key and an optional sort key for richer query capabilities.

Examples of Data Representation

- **Base Table**: Represents IoT sensor data with attributes like temperature and error status, indexed by device ID and timestamp.
- **Global Secondary Index (GSI)**: Provides alternate schemas to query data using different partition and sort keys while ensuring eventual consistency.
- **Local Secondary Index (LSI)**: Allows strong consistency reads and must be created with the table; supports alternate sort keys based on the same partition key.

Multi-Region Replication

- **Global Tables**: Enable multi-region, multi-active databases that replicate data automatically across selected AWS regions, enhancing availability and performance.

Security Best Practices

- **Preventative Measures**:
  - Use IAM roles for authentication.
  - Apply IAM policies for resource access and fine-grained control.
  - Implement VPC endpoints to limit access to DynamoDB.
- **Detective Measures**:
  - Utilize AWS CloudTrail for monitoring and logging usage.
  - Employ AWS Config to track configuration changes and compliance with rules.

By leveraging these features, DynamoDB provides a scalable, secure, and highly available database solution suitable for various applications.

Section 4

1. Amazon DocumentDB

- **Suitable Workloads:** Flexible schema, dynamic data storage, online customer profiles.
- **Data Model:** Document data model using JSON-like documents.
- **Key Features:** MongoDB-compatible, high performance for complex documents.
- **Common Use Cases:** Content management systems, customer profiles, operational data storage.

2. Amazon Keyspaces

- **Suitable Workloads:** Fast querying of high-volume data, scalability, heavy write loads.
- **Data Model:** Wide column data model, flexible columns.
- **Key Features:** Managed Apache Cassandra-compatible service, scalability, and high availability.

- **Common Use Cases:** Industrial equipment maintenance, trade monitoring, route optimization.

3. Amazon MemoryDB

- **Suitable Workloads:** Latency-sensitive applications, high request rates, high throughput.
- **Data Model:** In-memory database service.
- **Key Features:** In-memory speed, data durability, compatible with Redis.
- **Common Use Cases:** Caching, game leaderboards, banking transactions.

4. Amazon Neptune

- **Suitable Workloads:** Finding connections in data, complex relationships, highly connected datasets.
- **Data Model:** Graph data model with nodes and edges.
- **Key Features:** High throughput and low latency for graph queries, supports multiple graph query languages.
- **Common Use Cases:** Recommendation engines, fraud detection, knowledge graphs.

5. Amazon Timestream

- **Suitable Workloads:** Identifying trends over time, efficient data processing, ease of data management.
- **Data Model:** Time series data model.
- **Key Features:** Serverless, built-in timeseries functions, automatic data replication.
- **Common Use Cases:** IoT applications, web traffic analysis.

6. Amazon QLDB

- **Suitable Workloads:** Accurate history of application data, financial transaction tracking, data lineage verification.
- **Data Model:** Ledger database with immutable transaction logs.
- **Key Features:** Cryptographically verifiable transaction log, built-in data integrity, flexible querying with PartiQL.
- **Common Use Cases:** Financial transactions, claims history tracking, supply chain data management.

Key Takeaways

- Each AWS database service is purpose-built to address specific application needs:
    - **Amazon Redshift** for data warehousing.
    - **Amazon DocumentDB** for JSON document storage.
    - **Amazon Keyspaces** for wide-column data.

    - **Amazon MemoryDB** for in-memory applications.
    - **Amazon Neptune** for graph databases.
    - **Amazon Timestream** for timeseries data.
    - **Amazon QLDB** for ledger functionality.

This comprehensive overview serves to guide organizations in selecting the appropriate database solutions based on their unique requirements

Section 5

Overview of AWS Database Migration Service (AWS DMS)

- **Managed Service**: AWS DMS is a managed service that facilitates the migration and replication of existing databases and analytics workloads to and within AWS.
- **Supported Databases**: It supports a wide range of commercial and open-source databases, including Oracle, Microsoft SQL Server, MySQL, PostgreSQL, and more. It can replicate data on demand or on a schedule.
- **Endpoints**: AWS DMS allows migration between homogeneous (same database engine) and heterogeneous (different database engines) endpoints. One endpoint must be an AWS service.
- **Continuous Replication**: It can continuously replicate data with low latency, supporting various use cases such as building data lakes in Amazon S3 or consolidating databases into a data warehouse using Amazon Redshift.

Homogeneous Migration

- **Simplification**: Homogeneous migrations simplify moving databases with the same engine (e.g., from on-premises PostgreSQL to Amazon RDS for PostgreSQL).
- **Serverless**: These migrations are serverless, automatically scaling resources as needed.
- **Instance Profiles**: Migrations use instance profiles for network and security settings, ensuring a managed environment for the migration project.

Tools for Heterogeneous Migrations

- **Database Discovery Tool**: AWS DMS Fleet Advisor automatically inventories and assesses on-premises databases, identifying migration paths.
- **Schema Conversion Tools**:
    - **AWS Schema Conversion Tool (SCT)**: Converts source schema and SQL code into compatible formats for the target database.

    - **AWS DMS Schema Conversion**: A managed service for schema assessment and conversion integrated into AWS DMS workflows.

Example Use Case

- **Data Lake Replication**: AWS DMS can replicate data from an on-premises database (like a Student Information System) into an Amazon S3 data lake for analytics, allowing easy access to data for analysis and visualization.

Key Takeaways

- AWS DMS is efficient for migrating data quickly and securely to AWS, supporting both homogeneous and heterogeneous migrations.
- Tools like AWS SCT and AWS DMS Schema Conversion help streamline schema and code conversion processes.

This information provides a solid foundation for understanding AWS DMS and its role in database migration and replication. If you have specific questions or need further details on any aspect, feel free to ask!

Section 6

AWS Well-Architected Framework Database Pillars

The framework consists of six pillars, and this section highlights best practices relevant to database management:

1. **Performance Efficiency**:
    - **Architecture Selection**:
        - Use a data-driven approach to evaluate database options.
        - Consider the trade-offs that architectural choices have on customer experience and system performance.
        - Understand data characteristics and access patterns to select optimal data services.
2. **Security**:
    - **Data Protection**:
        - Implement secure key management and enforce encryption at rest.
        - Utilize AWS Key Management Service (KMS) to manage encryption keys effectively.
        - Ensure data confidentiality through encryption to mitigate risks of unauthorized access.
3. **Cost Optimization**:
    - **Cost-effective Resources**:

    - Select the correct database type, size, and number based on workload characteristics to minimize costs.
    - Engage in right-sizing practices to balance performance and cost effectively.
    - Consider the use of serverless options like Aurora Serverless for on-demand scaling.

Key Takeaways

- **Performance Efficiency**: Make selection choices based on data characteristics and access patterns to optimize workloads.
- **Security**: Implement secure key management and data protection measures to ensure data durability and safety.
- **Cost Optimization**: Assess and select resource types, sizes, and numbers based on workload requirements to achieve the lowest possible costs while meeting technical needs.

By adhering to these principles, cloud architects can create a robust database layer that enhances performance, security, and cost-effectiveness in AWS environments.

Module Checks

1. Which definition describes a virtual private cloud (VPC)?

- ○ A virtual private network (VPN) in the AWS Cloud
- ○ An extension of an on-premises network into Amazon Web Services (AWS)
- ◉ A logically isolated virtual network that you define in the AWS Cloud
- ○ A fully managed service that extends the AWS Cloud to customer premises

2. Which actions are best practices for designing a virtual private cloud (VPC)? (Select THREE.)

- ☐ Match the size of the VPC Classless Inter-Domain Routing (CIDR) block to the number of hosts that are required for a workload.
- ☐ Use the same Classless Inter-Domain Routing (CIDR) block as your on-premises network.
- ☑ Divide the VPC network range evenly across all Availability Zones available.
- ☑ Create one subnet per Availability Zone for each group of hosts that have unique routing requirements.
- ☑ Reserve some address space for future use.

3. A company wants to run a highly available web tier by using two EC2 instances and a load balancer. Which design is valid and provides the highest availability?

- ○ One subnet in one Availability Zone. The subnet contains two EC2 instances.
- ○ One subnet, which spans two Availability Zones. Each Availability Zone contains one EC2 instance.
- ○ Two different subnets in the same Availability Zone. Each subnet contains one EC2 instance.
- ◉ Two different subnets, one per Availability Zone. Each subnet contains one EC2 instance.

4. A company's VPC has the CIDR block 172.16.0.0/21 (2048 addresses). It has two subnets (A and B). Each subnet must support 100 usable addresses now, but this number is expected to rise to at most 254 usable addresses soon. Which subnet addressing scheme meets the requirements and follows AWS best practices?

- ○ Subnet A: 172.16.0.0/25 (128 addresses) Subnet B: 172.16.0.128/25 (1024 addresses)
- ○ Subnet A: 172.16.0.0/25 (128 addresses) Subnet B: 172.16.0.128/25 (128 addresses)
- ◉ Subnet A: 172.16.0.0/23 (512 addresses) Subnet B: 172.16.2.0/23 (512 addresses)
- ○ Subnet A: 172.16.0.0/22 (1024 addresses) Subnet B: 172.16.4.0/22 (128

5. Which combination of actions enables direct internet access for IPv4 hosts in a virtual private cloud (VPC)? (Select THREE.)

- ☐ Creating a default route that points to the virtual private gateway
- ☑ Configuring hosts to have or obtain an internet-routable address
- ☑ Configuring security groups and network access control lists (network ACLs) to permit internet traffic
- ☑ Creating a route for 0.0.0.0/0 that points to the internet gateway
- ☐ Configuring the VPC domain name in a Dynamic Host Configuration Protocol (DHCP) options set

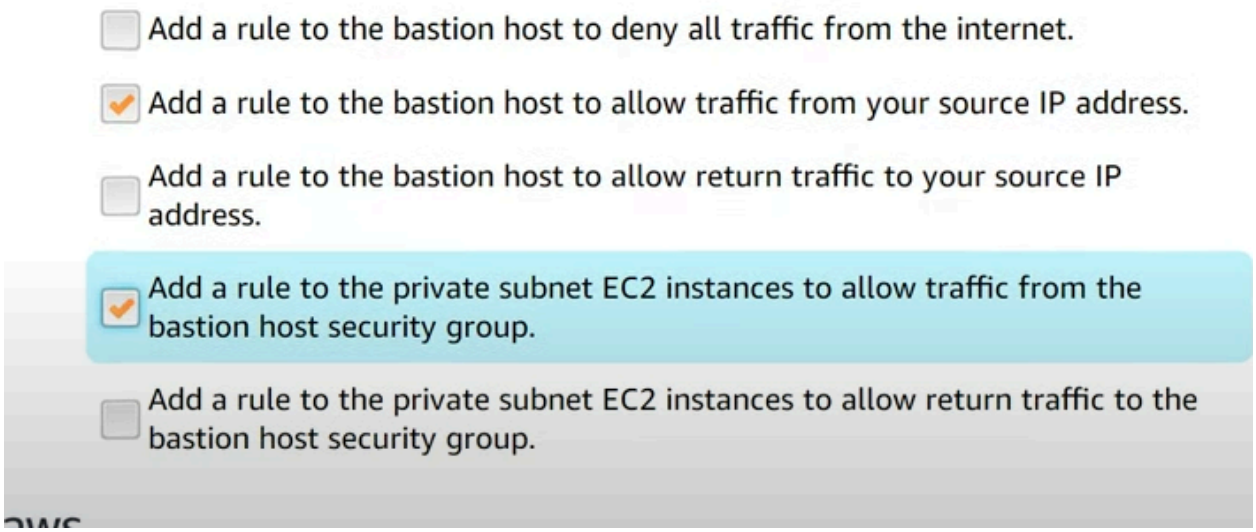
6. A group of consultants requires access to an EC2 instance from the internet, for 3 consecutive days each week. The instance is shut down the rest of the week. The virtual private cloud (VPC) has internet access. How should you assign an IPv4 address to the instance to give the consultants access?

- (●) Associate an Elastic IP address with the EC2 instance
- ( ) Enable automatic address assignment for the subnet
- ( ) Enable automatic address assignment for the EC2 instance
- ( ) Assign the IP address in the operating system (OS) boot configuration

7. Several EC2 instances launch in a virtual private cloud (VPC) that has internet access. These instances should not be accessible from the internet, but they must be able to download updates from the internet. How should the instances launch?

- ( ) With Elastic IP addresses, in a subnet with a default route to an internet gateway
- ( ) With public IP addresses, in a subnet with a default route to an internet gateway
- ( ) Without public IP addresses, in a subnet with a default route to an internet gateway
- (●) Without public IP addresses, in a subnet with a default route to a network address translation (NAT) gateway

8. You are configuring a bastion host to access EC2 instances in a virtual private cloud (VPC). What must you do to the security groups? (Select TWO.)



9. You have a virtual private cloud (VPC) with a public subnet and a secure subnet. All EC2 instances in the secure subnet must be able to communicate with specific internet addresses. How can you control traffic with a network access control list (network ACL)?

- Add rules to the default network ACL to allow traffic from and to allowed internet addresses.
- Add rules to the default network ACL to allow traffic from and to allowed internet addresses. Deny all other traffic.
- **Add rules to the subnet custom network ACL to allow traffic from and to allowed internet addresses.**
- Add rules to the subnet custom network ACL to allow traffic from and to

10. All of the EC2 instances in a subnet can communicate with a certain IPv4 network on the internet. How should you modify the security groups or current custom network access control list (network ACL) to deny traffic to and from several restricted addresses in that network?

- **In the network ACL, deny traffic to and from the restricted addresses.**
- In the security groups, deny traffic to and from the restricted addresses.
- In the network ACL, allow traffic only to and from address ranges that exclude the restricted addresses.
- In the security groups, allow traffic only to and from address ranges that exclude the restricted addresses

Section 1

Here's a summary of the key points regarding Amazon VPC (Virtual Private Cloud) networking, focusing on public and private subnets, Elastic IP addresses, and NAT devices:

Subnets in Amazon VPC

1. **Public Subnets**:
    - Have direct access to the internet through an Internet Gateway.
    - Require instances to have both private and public IP addresses for internet connectivity.
    - A public subnet route table includes a route for `0.0.0.0/0` that points to the Internet Gateway.
2. **Private Subnets**:
    - Do not have direct access to the internet.
    - Instances in private subnets cannot be reached from the internet, enhancing security.
    - The private subnet route table typically mirrors the main VPC route table.
3. **Elastic IP Addresses**:
    - Static public IP addresses associated with an EC2 instance.
    - Can be transferred between instances if needed.
    - There is no cost for the first Elastic IP associated with a running instance; additional Elastic IPs incur charges.

NAT Devices

1. **NAT (Network Address Translation) Devices**:
    - Allow instances in a private subnet to initiate outbound traffic to the internet while remaining unreachable from the internet.
    - Two types of NAT devices:
        - **NAT Gateway**: A managed AWS service that incurs an hourly cost, providing better availability and bandwidth.
        - **NAT Instance**: An EC2 instance configured to perform NAT, which incurs EC2 usage costs.
2. **Connecting Private Subnets to the Internet**:
    - To allow instances in private subnets to access the internet, route traffic through a NAT device.
    - NAT gateways and instances are placed in public subnets to access the Internet Gateway.

Use Cases for Subnets

- **Database Instances**: Recommended in private subnets for security.
- **Batch-Processing Instances**: Should also reside in private subnets.
- **Web Application Instances**: Can be placed in public or private subnets, but AWS recommends placing them in private subnets behind a load balancer for enhanced security.

Key Takeaways

- Amazon VPC creates a logically isolated virtual network.
- Public subnets allow direct internet access; private subnets do not.
- NAT gateways enable outbound internet connectivity for private subnet resources.
- Elastic IPs can be reassigned to different instances as needed.

This summary encapsulates the main points from the training module on creating a networking environment in AWS.

Section2

This text provides a detailed overview of network security mechanisms in AWS, focusing on **Security Groups**, **Network ACLs (Access Control Lists)**, **AWS Network Firewall**, and **Bastion Hosts**. Here's a breakdown of the key concepts and differences discussed in the content:

1. Security Groups vs. Network ACLs

- **Scope**:
    - **Security Groups**: Operate at the resource level (e.g., EC2 instances).
    - **Network ACLs**: Operate at the subnet level.
- **Traffic Rules**:
    - **Security Groups**: Only allow rules (stateful). Inbound rules are not allowed by default, but all outbound traffic is allowed by default.
    - **Network ACLs**: Can specify both allow and deny rules (stateless). By default, they allow all inbound and outbound traffic.
- **Statefulness**:
    - **Security Groups**: Stateful, meaning return traffic is automatically allowed.
    - **Network ACLs**: Stateless, so return traffic must be explicitly allowed.
- **Rule Evaluation**:
    - **Security Groups**: All rules are evaluated.
    - **Network ACLs**: Rules are evaluated in number order, and evaluation stops at the first match.

2. AWS Network Firewall

- Acts as an additional layer of security for VPCs, providing intrusion detection and prevention.
- It is stateful and managed, inspecting incoming traffic to protect subnet resources.
- Requires modification of VPC route tables to route external traffic through the firewall.

3. Bastion Hosts

- A bastion host provides secure access to private subnets from an external network, minimizing direct access to instances.
- Security groups are used to control access; the bastion host connects to resources in the private subnet through SSH (port 22).
- Ideally, only the bastion host should be the source of SSH traffic to the private instances.

Key Takeaways

- Implement multiple layers of defense for securing AWS infrastructure.
- Use security groups for resource-level traffic control and network ACLs for subnet-level control.
- Route external VPC traffic through AWS Network Firewall for enhanced security.
- Utilize bastion hosts to securely administer resources in private subnets from external environments.

This overview encapsulates the content, highlighting the key differences and functionalities of AWS networking security features.

Section 3

1. Connecting EC2 Instances to Managed Services

- **Use Case**: After deploying a workload on an EC2 instance in a private subnet, you may need to access an Amazon S3 bucket in the same AWS Region. Since Amazon S3 operates outside your VPC, direct connectivity is not possible.
- **Challenges**: Accessing S3 through the public internet incurs data transfer costs and may expose your traffic.

2. VPC Endpoints

There are two main types of VPC endpoints for secure, private connectivity to AWS managed services:

A. Interface VPC Endpoints

- **Description**: These use AWS PrivateLink and allow private connections to AWS services.
- **Components**: AWS creates an Elastic Network Interface (ENI) with a private IP address in each specified subnet.
- **IAM Policies**: You can control access to the endpoint using IAM resource policies.
- **Cost**: Charged for hourly usage and data processed.

B. Gateway VPC Endpoints

- **Description**: These provide direct connections to Amazon S3 and DynamoDB using route tables, without AWS PrivateLink.
- **Components**: No additional charge for usage, with no throughput limitations.
- **Route Table**: Configured in the private subnet's route table to direct traffic to S3 and DynamoDB.

3. Steps to Set Up an Interface VPC Endpoint

1. Specify the name of the AWS service you want to connect to.
2. Choose the VPC and specify subnets in different Availability Zones for redundancy.
3. Select a subnet for the interface endpoint, creating a network interface.
4. Specify security groups to control traffic to the network interface.

4. Gateway Load Balancer Endpoint

- **Description**: This type of endpoint connects security appliances in one VPC to application instances in another, allowing for traffic inspection.
- **Traffic Flow**: Incoming and outgoing traffic is routed through the Gateway Load Balancer for inspection before reaching the EC2 application instance.

5. Key Takeaways

- **VPC Resources**: Can access AWS managed services using VPC endpoints.
- **Cost Considerations**: Interface endpoints incur costs while gateway endpoints do not.
- **Throughput**: Interface endpoints have limitations, whereas gateway endpoints do not.
- **Security**: The use of private IP addresses enhances security and eliminates internet exposure.

6. Factors for Choosing Between Endpoint Types

- **Access Method**: Can S3 objects be accessed via public IP or only through a private IP?
- **On-Premises Access**: Do you need on-premises connectivity?
- **Region Access**: Is access needed from another AWS Region?
- **Cost**: Is the budget sufficient for interface endpoint costs?
- **Bandwidth and Packet Size**: What are the throughput requirements and maximum packet size?

This information should provide a solid understanding of how to effectively connect to AWS managed services from a VPC, along with considerations for cost and security.

Section 4

Network Troubleshooting Scenarios

1. **Common Issues**:
    - Slow EC2 instance response times
    - Inability to access EC2 instances via SSH
    - EC2 database instances not applying patches
2. **Investigation**:
    - Monitor network traffic to identify issues like unnecessary traffic (e.g., DDoS attacks).
    - Verify security group rules for allowed traffic (e.g., port 22 for SSH).
    - Check configurations, such as NAT gateways and route tables, for private subnets.

Amazon VPC Flow Logs

- **Purpose**: To capture and log network traffic for troubleshooting.
- **Log Types**:
    - **VPC flow logs**: General traffic monitoring.
    - **Elastic network interface flow logs**: Specific to network interfaces.
    - **Subnet flow logs**: Focused on specific subnets.
- **Log Delivery**:
    - Logs can be sent to Amazon CloudWatch, Amazon S3, or Amazon Kinesis Data Firehose.
    - Can be queried using Amazon Athena or visualized in Amazon OpenSearch Service.

IAM Access for Flow Logs

- Users must have appropriate IAM permissions to create, describe, and delete flow logs.
- An example IAM policy allows these actions across all AWS resources.

Flow Log Record Structure

- Flow logs consist of multiple fields, including:
    - Version, account ID, interface ID, source and destination addresses, ports, and protocol.
    - Also tracks packets transferred, bytes transferred, and action (accept/reject).
- **Log Status**:

- `OK`: Normal logging.
- `NODATA`: No traffic during the interval.
- `SKIPDATA`: Some records skipped due to constraints or errors.

Additional VPC Troubleshooting Tools

1. **Reachability Analyzer**:
   - Tests connectivity between source and destination resources.
   - Identifies any blocking components, such as security groups or network ACLs.
2. **Network Access Analyzer**:
   - Identifies unintended network access and helps improve security posture.
   - Useful for compliance verification (e.g., isolating networks processing sensitive information).
3. **Traffic Mirroring**:
   - Creates copies of network traffic for analysis.
   - Enables deep inspection of actual packet content for troubleshooting performance issues or detecting network attacks.

These tools and practices help maintain a secure and efficient networking environment in AWS, ensuring that issues can be quickly identified and resolved.

Section 5

AWS Well-Architected Framework Network Pillars

When designing a VPC network, it's essential to consider future workloads to ensure they are resilient, secure, performant, and cost-effective. The network design should support these standards, as the AWS Well-Architected Framework provides best practices for workload design, operation, and maintenance, helping you make informed architectural decisions.

Key Best Practices for Network Design

1. **Plan Your Network Topology**:
   - **Resiliency**: The network should perform reliably, anticipating failures and accommodating future traffic growth.
   - **IP Address Allocation**: Ensure sufficient IP subnet allocation for expansion and future needs:
     - Use CIDR blocks that allow multiple subnets across Availability Zones (AZs).
     - Reserve CIDR space for future growth and be mindful of reserved IP addresses.
     - Deploy large CIDR blocks as they cannot be changed or deleted later.
2. **Infrastructure Protection**:

    - **Network Layers**: Group components based on sensitivity. For example, databases with no internet access should reside in isolated subnets.
    - **Control Traffic**: Implement security controls at all layers using security groups, network ACLs, subnets, and route tables.
    - **Inspection and Protection**: Use tools like the VPC Network Access Analyzer to inspect and filter traffic.
3. **Network Architecture Selection**:
    - **Performance Impact**: Analyze how network-related decisions affect workload performance (e.g., consider latency between AZs versus regions).
    - **Networking Features**: Continuously benchmark and evaluate workload performance metrics.
    - **Network Protocols**: Select appropriate protocols (e.g., using TCP for critical data and UDP for real-time data) to enhance performance.
4. **Select the Best Pricing Model**:
    - **Region Selection**: Choose AWS Regions based on cost and proximity to users to minimize latency and meet data privacy requirements.

Identifying Network Design Issues

In the provided scenario for Company A, the following design mistakes were identified:

- **Small VPC/Subnets**: A small VPC with limited IP addresses hinders future growth.
- **Permissive Security Groups**: Security groups allowed broad internet access to both web and database servers.
- **Direct Database Access**: The design permitted direct internet access to database servers, which is insecure.
- **Poor Region Choice**: Deploying resources in a European region while the customer base is in the US leads to higher latency.

Recommendations for Improvement

1. **Use Large VPCs**: Implement larger VPCs with sufficient IP addresses for anticipated growth.
2. **Separate Security Groups**: Configure distinct security groups for web and database servers to restrict access appropriately.
3. **Private Subnets for Databases**: Place databases in private subnets without direct internet access, allowing maintenance through secure channels.
4. **Choose a Closer AWS Region**: Deploy in an AWS Region located near the customer base to reduce latency and ensure compliance with data sovereignty.

Key Takeaways

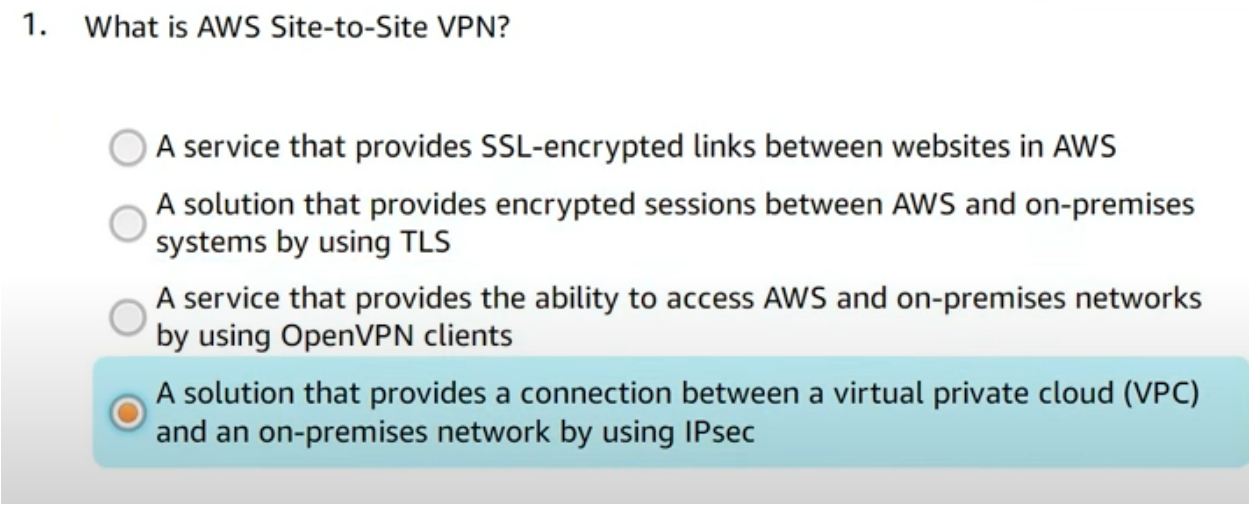
- Plan for IP subnet allocation that accommodates growth.
- Establish network layers and control traffic effectively.

- Understand the impact of networking on performance.
- Optimize performance through suitable network protocols.
- Select AWS Regions based on cost and proximity to users for efficient service delivery.

These principles will guide the design of a robust, secure, and efficient VPC network that aligns with AWS best practices.

Module Checks

1. What is AWS Site-to-Site VPN?



2. What does AWS Direct Connect provide?

- [x] A dedicated network connection from an on-premises network to AWS that uses 802.1q
- [ ] A private telecommunications circuit from an on-premises network direct into AWS that uses Point-to-Point Protocol (PPP)
- [ ] An encrypted tunnel that connects an on-premises network to AWS over the internet
- [ ] An extension of the AWS Cloud into customer data centers that uses AWS hardware installed on premises

3. A company has two virtual private clouds (VPCs). VPC A has a Classless Inter-Domain Routing (CIDR) block of 10.1.0.0/16. VPC B has CIDR block of 10.2.0.0/16. Both VPCs belong to the same Amazon Web Services (AWS) account. What is the simplest way to connect the two VPCs so that they can route all traffic between them?

- [ ] AWS Site-to-Site VPN
- [ ] AWS Direct Connect
- [x] VPC peering
- [ ] VPC endpoints

4. A company is implementing a system to back up on-premises systems to Amazon Web Services (AWS). Which network connectivity method will provide a solution with the most consistent performance?

- [ ] AWS Site-to-Site VPN
- [x] AWS Direct Connect
- [ ] VPC peering
- [ ] VPC endpoints

5. Systems in a secure subnet in a virtual private cloud (VPC) must access a bucket in Amazon Simple Storage Service (Amazon S3). Which solution stops traffic from crossing the internet?

- (●) Create a VPC gateway endpoint for Amazon S3
- ( ) Use a private IP address for the system
- ( ) Use the private IP address of Amazon S3
- ( ) Create a VPC peering connection to Amazon S3

6. A company has three virtual private clouds (VPCs). VPCs A, B, and C have Classless Inter-Domain Routing (CIDR) blocks that do not overlap. Both A and C have separate VPC peering connections with B. However, A cannot communicate with C. What is the simplest and most cost-effective way to enable full communication between A and C?

- ( ) Add routes to B to enable traffic between A and C through B.
- (●) Add a peering connection between A and C, and route traffic between A and C through the peering connection.
- ( ) Link all three VPCs through a transit VPC, and route all traffic through the transit VPC.
- ( ) Create VPC endpoints in A and C for the individual hosts that need to

7. Because of a natural disaster, a company moved a secondary data center to a temporary facility with internet connectivity. It needs a secure connection to the company's virtual private cloud (VPC) that must be operational as soon as possible. The data center will move again in 2 weeks. Which option meets the requirements?

- ( ) VPC peering
- ( ) VPC endpoints
- ( ) AWS Site-to-Site VPN
- ( ) AWS Direct Connect

KEYBOARD NAVIGATION

8. What is the simplest way to connect 100 virtual private clouds (VPCs) together?

- Create a hub-and-spoke network by using AWS VPN CloudHub
- Chain VPCs together by using VPC peering
- Connect each VPC to all the other VPCs by using VPC peering
- **Connect the VPCs to AWS Transit Gateway**

9. A company's security administrator requires that EC2 instances in a specific subnet must connect to Amazon DynamoDB through a VPC endpoint. The company's network standards require that the infrastructure support high availability. Which action meets these architecture requirements without adding another subnet?

- **Associate a single VPC endpoint with the subnet**
- Associate two VPC endpoints with the subnet
- Associate two VPC endpoints with the subnet and use Elastic Load Balancing
- Associate VPC endpoints through an Auto Scaling group that is connected to Elastic Load Balancing

KEYBOARD NAVIGATION

10. A company uses a single AWS Direct Connect connection between their on-premises network and their virtual private cloud (VPC). They want to ensure that the network connectivity is highly available by adding a backup connection. Which network connectivity method provides most cost-effective solution for the backup connection?

- Another AWS Direct Connect connection through the same Direct Connect location
- Another AWS Direct Connect connection through a different Direct Connect location
- An on-demand AWS Client VPN connection across the internet
- An on-demand AWS Site-to-Site VPN connection across the internet

S

Section 1

key concepts related to scaling VPC networks with AWS Transit Gateway, focusing on centralized routing and peering strategies. Here's a summary:

Centralized Outbound Routing Pattern:

- **Purpose**: Centralizing egress internet traffic for enhanced security and cost efficiency by routing outbound internet traffic from multiple VPCs through a dedicated egress VPC containing a NAT gateway.
- **Benefits**: Simplifies monitoring and control, reduces NAT gateway costs by centralizing the function to one VPC, and enhances security. A NAT gateway is recommended for each Availability Zone for redundancy.

Transit Gateway Peering:

- **Scenario**: When VPCs in different AWS Regions or accounts need communication, transit gateway peering is used. This allows network traffic to flow between VPCs across different accounts and regions without traversing the public internet, enhancing security.
- **Configuration**: To enable this, both transit gateways must create and accept peering connections, and the route tables should be updated to point to the transit gateway attachments.

Company Scenario - Connecting Multiple Departments:

- **Problem**: A company with multiple VPCs (in the same or different AWS accounts) needs full resource sharing between departments.
- **Solution**: The simplest and most scalable solution is to connect all departmental VPCs to a transit gateway, ensuring full connectivity and simplifying future expansion.

Configuration Activity:

- **VPC Route Tables**: Configure the VPC route tables by setting each VPC’s route destination as the CIDR range of all connected VPCs and routing it through the transit gateway.
- **Transit Gateway Route Tables**: The transit gateway route table must have entries for each VPC CIDR block, with the respective transit gateway VPC attachment as the target.

Key Takeaways:

- **Transit Gateway**: Acts as a centralized regional router connecting multiple VPCs, providing a scalable solution for managing network traffic across regions and AWS accounts.
- **Costs**: Charges are based on the number of connections and the amount of traffic passing through the transit gateway.

This pattern simplifies VPC management, reduces costs, and improves security for larger organizations or those needing cross-region or cross-account connectivity.

Section 2

This section focuses on connecting multiple VPCs in AWS using the **VPC Peering** feature.

- **VPC Peering** establishes a one-to-one, point-to-point connection between two VPCs, allowing **Amazon EC2 instances** in those VPCs to communicate over private IP addresses, similar to being on the same network. It is ideal for smaller environments or when the budget is constrained, as it does not incur costs (except for inter-Region or inter-Availability Zone data transfers).
- The feature allows connections between VPCs owned by the same account, different accounts, or even across different AWS Regions, enabling resource sharing or geographic redundancy. Inter-Region traffic is encrypted, and VPC peering ensures that all traffic remains on the AWS backbone, reducing exposure to common internet threats.

VPC Peering Architecture and Use

- **Mesh Architecture**: When a small number of VPCs need to be connected without the need for transit gateways, you can use peering for each VPC pair.
- **No Transitive Peering**: Traffic between two VPCs is isolated to the peering connection, meaning traffic from VPC A to VPC B does not automatically pass through to VPC C unless direct peering connections are established.

Establishing VPC Peering

1. **Requesting and Accepting**: One VPC sends a request to peer, and the other accepts. CIDR blocks of the VPCs cannot overlap.
2. **Updating Route Tables**: Each VPC owner updates their route table to add a route to the other's CIDR block, specifying the peering connection as the target.
3. **Security Groups**: The VPC owners may also need to update security group rules to allow traffic between the peered VPCs.

Limitations

- **Transitive Peering**: Not supported. Peering connections are direct and isolated.

- **CIDR Block Restrictions**: Peering is not possible with overlapping CIDR blocks.
- **Internet/NAT Gateway**: VPCs in a peering connection cannot use each other's internet or NAT gateway.

PrivateLink Architecture

- For **application-level connections** or **overlapping CIDR blocks**, **AWS PrivateLink** can be used with a **Network Load Balancer**. This allows consumer VPCs to connect to service provider VPCs within the same AWS Region, without needing a peering connection.

Use Cases

1. **File Sharing**: Peering VPCs to share data without exposing traffic to the internet.
2. **Customer Access**: Providing limited access to customers by peering their VPCs with your central VPC.
3. **Active Directory Integration**: Using VPC peering for centralized services like Active Directory while restricting traffic flow to other VPCs.

This helps create secure, low-latency, private connections between VPCs for resource sharing and communication.

Section 3

This section covers how to connect on-premises environments to an Amazon Virtual Private Cloud (VPC) using a Site-to-Site VPN, AWS VPN CloudHub, or AWS Global Accelerator. The primary method is through AWS **Site-to-Site VPN**, which creates encrypted IPsec VPN tunnels between the on-premises customer gateway and the AWS virtual private gateway (or transit gateway). This setup provides a secure connection to your VPC and ensures high availability by creating two VPN tunnels, one for primary traffic and the other for redundancy.

The process involves:

1. Creating a **customer gateway** for the on-premises device.
2. Setting up a **virtual private gateway** with different Autonomous System Numbers (ASN).
3. Configuring routing tables and **updating security groups** for protocols like SSH or RDP.
4. Establishing the **VPN connection** and downloading configuration details.

For larger organizations with multiple on-premises networks, **AWS VPN CloudHub** enables centralized connectivity, allowing communication between multiple customer gateways using unique BGP ASNs. AWS **Global Accelerator** can also be used to improve VPN connection performance by routing traffic over AWS's infrastructure.

Lastly, by using **Transit Gateway**, you can isolate VPCs from each other while still providing full VPN access to the on-premises network. This setup ensures VPC isolation while routing traffic efficiently between on-premises and VPC environments.

Key takeaways:

- **Site-to-Site VPN** ensures secure, encrypted communication between on-premises networks and AWS.
- **Global Accelerator** improves performance for transit gateway-attached VPN connections.
- **Transit Gateway** can be configured to isolate VPCs, preventing cross-communication while maintaining VPN connectivity to on-premises networks.

Section 4

This section discusses how to connect an on-premises network to an Amazon Virtual Private Cloud (VPC) using AWS Direct Connect. AWS Direct Connect is a dedicated, private, virtual local area network (VLAN) connection that links your on-premises network to AWS resources. It offers a consistent network experience with predictable performance, high bandwidth, and low latency compared to Site-to-Site VPN, which uses encrypted tunnels over the public internet.

Key Points on AWS Direct Connect:

1. **Dedicated Private Network**: Direct Connect establishes a private connection from your on-premises network to AWS, avoiding public internet use. It utilizes VLANs to extend the private network into AWS resources.
2. **Use Cases**:
    - **Hybrid Environments**: Allows seamless integration with on-premises infrastructure, enabling applications that require access to on-premises data.
    - **Large Datasets**: Ideal for applications requiring the transfer of large datasets, offering high throughput and reducing internet bandwidth costs.
    - **Predictable Performance**: Beneficial for real-time applications (e.g., audio, video streaming) where network consistency is critical.
    - **Security and Compliance**: Direct Connect meets enterprise security requirements by ensuring that traffic flows only through private circuits.
3. **Types of Virtual Interfaces**:
    - **Public Virtual Interface**: Provides access to public AWS services like Amazon S3.
    - **Private Virtual Interface**: Connects to a VPC via a virtual private gateway.
    - **Transit Virtual Interface**: Connects to multiple VPCs through a transit gateway using a Direct Connect gateway.
4. **High Availability and Resiliency**:
    - **High Availability**: Direct Connect can be paired with a Site-to-Site VPN connection as a backup to ensure continuous network availability.

    - **High Resiliency**: For critical workloads, it is recommended to use multiple Direct Connect locations and diverse hardware/telecommunications providers to avoid single points of failure.

By following these strategies, organizations can ensure high performance, availability, and secure connections between their on-premises networks and AWS

Section 4

The AWS Well-Architected Framework provides a set of best practices to ensure that workloads deployed over multiple networks, both on-premises and in the cloud, are resilient, secure, high-performing, and cost-effective. Here are the key principles and best practices from the framework's pillars that are relevant for network design:

Key Best Practices for Network Architecture:

1. **Resilient Network Topology**:
    - **Provision Redundant Connectivity**: Ensure failover mechanisms are in place for network connectivity between on-premises environments and AWS. For instance, use redundant Direct Connect connections or combine Direct Connect with a VPN backup to handle network interruptions.
    - **Hub-and-Spoke Topology**: Use hub-and-spoke designs, like AWS Transit Gateway, instead of many-to-many mesh configurations (e.g., VPC peering). This reduces complexity and enhances network scalability.
2. **Infrastructure Protection**:
    - **Control Traffic at All Layers**: Adopt a zero-trust model, securing traffic at every layer using AWS features like Direct Connect and Site-to-Site VPN for private network communications.
    - **Secure Communication**: Protect sensitive data in transit by using protocols such as TLS for authentication and IPsec VPN tunnels for encryption.
3. **Data Protection**:
    - **Authentication and Encryption**: Verify the identity of network communications and enforce encryption protocols like TLS 1.3 to protect data in transit between workloads, services, and end-users.
4. **Network Architecture Selection**:
    - **Choose Appropriate Connectivity**: Select appropriately sized dedicated connections or VPNs based on your bandwidth, latency, and security needs. Direct Connect is often preferred for predictable performance and lower latency, while VPN can be useful for lower-priority workloads or as a backup.
    - **Workload Placement**: Choose locations that minimize latency and optimize network performance, using services like Global Accelerator for improved traffic routing.
5. **Cost Optimization**:

- **Optimize Data Transfer**: Minimize network data transfer costs by using services like content delivery networks (CDNs), Direct Connect, and WAN optimization strategies. This reduces the overall data flow and leverages AWS's pricing models for better cost efficiency.

Conclusion:

When designing networks that span both on-premises and cloud environments, it's essential to balance resiliency, security, performance, and cost efficiency. The AWS Well-Architected Framework provides a structured approach, ensuring your workload is robust and scalable across networks while adhering to best practices.

Module Checks

1. Which statement describes AWS Identity and Access Management (IAM) users?

- ○ IAM users are used to control access to a specific AWS resource.
- ○ IAM user names can represent a collection of individuals.
- ◉ Every IAM user for an account must have a unique name.
- ○ Every IAM user name is unique across all AWS accounts.

2. How can you grant the same level of permissions to multiple users within an account?

- ◉ Apply an AWS Identity and Access Management (IAM) policy to an IAM group.
- ○ Apply an AWS Identity and Access Management (IAM) policy to an IAM role.
- ○ Create a resource-based policy.
- ○ Create an organization in AWS Organizations.

3. Which statements describe AWS Identity and Access Management (IAM) roles? (Select TWO.)

- ☐ They are uniquely associated to an individual.
- ☐ They can only be used by accounts associated to the person who creates the role.
- ☑ They can be assumed by individuals, applications, and services.
- ☑ They provide temporary security credentials.
- ☐ They provide permanent security credentials.

4. Which statement describes a resource-based policy?

- ○ It can be applied to any AWS resource.
- ○ It can be an AWS managed policy.
- ○ It is attached to a user or group.
- ◉ It is always an inline policy.

5. How does AWS Identity and Access Management (IAM) evaluate a policy?

- ○ It checks for explicit allow statements before it checks for explicit deny statements.
- ◉ It checks for explicit deny statements before it checks for explicit allow statements.
- ○ If there is no explicit deny statement or explicit allow statement, users will have access by default.
- ○ An explicit deny statement does not override an explicit allow statement.

6. A team of developers needs access to several services and resources in a virtual private cloud (VPC) for 9 months. How can you use AWS Identity and Access Management (IAM) to enable access for them?

- ○ Create a single IAM user for the developer team and attach the required IAM policies.
- ○ Create an IAM user for each developer, and attach the required IAM policies to each IAM user.
- ◉ Create an IAM user for each developer, put them all in an IAM group, and attach the required IAM policies to the IAM group.
- ○ Create a single IAM user for the developer team, place it in an IAM group, and attach the required IAM policies to the IAM group.

KEYBOARD NAVIGATION

7. How does identity federation increase security for an application that is built in Amazon Web Services (AWS)?

- (●) Users can use single sign-on (SSO) to access the application through an existing authenticated identity.
- ( ) The application can synchronize users' user names and passwords in AWS Identity and Access Management (IAM) with their social media accounts.
- ( ) The browser can establish a trust relationship with the application to bypass the need for multi-factor authentication (MFA).
- ( ) Users can use their AWS Identity and Access Management (IAM) accounts to log in to on-premises systems.

KEYBOARD NAVIG

8. Which services can you use to enable identity federation for your applications that are built in Amazon Web Services (AWS)? (Select TWO.)

- [ ] AWS WAF
- [ ] AWS Key Management Service (AWS KMS)
- [x] AWS Security Token Service (AWS STS)
- [ ] AWS CloudHSM
- [x] Amazon Cognito

KEYBOARD NAVIGATION

9. What service helps you centrally manage billing; control access, compliance and security; and share resources across multiple Amazon Web Services (AWS) accounts?

- ( ) AWS Identity and Access Management (IAM)
- ( ) AWS Control Tower
- ( ) AWS Organizations
- ( ) Amazon Virtual Private Cloud (Amazon VPC) peering

10. A technology company's employees log in to their Amazon Web Services (AWS) accounts through AWS Identity and Access Management (IAM) users. They have administrator access and access to the root users. Which resource can prevent them from deleting the AWS CloudTrail logs?

- An IAM policy that is attached to each IAM user
- A service control policy (SCP) that is attached to the organizational unit (OU)
- An Amazon S3 bucket policy that is attached to logging buckets
- IAM users with administrative access can override the S3 bucket policies.

Section 1

This text explains key concepts about managing user permissions in AWS, focusing on IAM (Identity and Access Management) groups, policies, and access control models like RBAC (Role-Based Access Control) and ABAC (Attribute-Based Access Control).

Key Concepts:

1. **IAM Groups and Policies:**
    - Groups reflect job functions, such as admins, developers, and testers.
    - Adding users to groups automatically grants them the same permissions as other members.
    - Permissions are managed via group policies, which can be updated to modify access for all group members.
    - User policies can override group policies if they are more restrictive, as shown in Zhang's example.
2. **Role-Based Access Control (RBAC):**
    - Permissions are assigned based on job roles.
    - The challenge is scaling because policies must be updated whenever new resources are added, leading to multiple policy changes.
3. **Attribute-Based Access Control (ABAC):**
    - ABAC uses attributes (key-value pairs) to grant access based on tags applied to users, roles, or resources.
    - This method is more flexible and scalable than RBAC because permissions are defined based on tags, not individual resources.
    - ABAC allows a single policy to manage permissions for various resources, reducing the need to update multiple policies.
4. **Tags in AWS:**
    - Tags (key/value pairs) can be applied to both AWS resources and identities (e.g., users and roles).
    - Tags help manage access control, billing, filtering, and resource management.
    - Up to 50 tags can be applied to each AWS resource, with practical uses in organizing technical, business, and security data.

Example Use Case:

- **ABAC for Project Management:** Tags like "Env=Dev" or "Project=NewDev" can be applied to roles and resources to grant the correct level of access. For instance, developers tagged with "Project=NewDev" will only have access to resources also tagged for the "NewDev" project.

Key Takeaways:

- Use IAM groups to assign common permissions across users based on job functions.
- ABAC is more scalable than RBAC and simplifies permission management by using tags as attributes for access control.

Section 2

The slides you provided offer a detailed overview of **identity federation** and its application in AWS, particularly using **AWS STS**, **SAML**, and **Amazon Cognito**. Below is a summary of the key points:

1. Identity Federation to AWS

- **Identity Providers (IdP)**: Users sign in using existing credentials from external IdPs (e.g., corporate login, social accounts like Amazon, Google).
- **Identity Broker**: Acts as an intermediary between the IdP and the Service Provider (AWS in this case), requesting temporary credentials from AWS STS on behalf of the user.
- **AWS Security Token Service (STS)**: Dynamically generates temporary credentials for the application. These credentials have a limited lifespan (minutes to hours).
- **Application Access**: The temporary credentials passed from the identity broker allow users to access AWS services without needing to sign into AWS directly.

2. Identity Federation for AWS Management Console Access

- **OIDC-Based Flow**:
    1. User signs in through a corporate IdP (e.g., Microsoft AD or LDAP).
    2. The identity broker authenticates and requests temporary credentials from AWS STS.
    3. The application redirects the user to AWS Management Console using these credentials.
    4. The user can access AWS services based on permissions granted.
- **SAML-Based Flow**:
    1. A corporate portal functions as the IdP and authenticates the user.
    2. The portal issues a **SAML assertion**, which is sent to AWS.
    3. AWS STS generates temporary credentials using the `AssumeRoleWithSAML` operation.
    4. The user is redirected to AWS Management Console with these credentials.

3. Amazon Cognito for Identity Federation

- **Amazon Cognito** is a fully managed service that provides:
    - Authentication, authorization, and user management.

    - **User pools** for creating a directory of user profiles, managing sign-ups, sign-ins, and third-party logins (social providers or SAML).
    - **Identity pools** for creating unique identities and obtaining temporary AWS credentials to access AWS services.
- **Application Access**:
    - After signing in via a user pool, users can access AWS services through temporary credentials obtained from an identity pool.

4. Cognito User Pools

- **User Pools**: Acts as a standalone user directory and can handle federated identities from third-party IdPs (e.g., Facebook, Google).
- **User Pool Features**:
    - Hosted UI for sign-up/sign-in.
    - Supports **JWTs** for accessing resources or exchanging for AWS credentials.
    - User group management for organizing permissions.

5. Key Takeaways

- **Federation** enables the use of external identities (corporate or social) to access AWS services.
- **AWS IAM Identity Center** offers unified permission management.
- **AWS STS** facilitates temporary credential generation for IAM users and applications.
- **Amazon Cognito** simplifies authentication and authorization for web and mobile apps, supporting both local and federated identities.

Section 3

his content provides a comprehensive guide on managing access and permissions in AWS environments, especially within multi-account structures and organizations. Here's a breakdown of the key points:

1. **Service Control Policies (SCPs)**:
    - SCPs are used in AWS Organizations to enforce guardrails on what actions users and accounts can perform.
    - Example: Preventing member accounts from leaving the organization with a policy that explicitly denies the `organizations:LeaveOrganization` action.
2. **Attaching SCPs**:
    - SCPs can be attached to roots, Organizational Units (OUs), or individual accounts, and their effects cascade downward through the hierarchy.
    - SCPs only grant access if both explicitly allowed and not explicitly denied by any other SCP or IAM policy that applies.
3. **Combining SCPs with IAM Policies**:

    - Permissions granted must be allowed by both SCPs and IAM identity-based policies.
    - An explicit deny in either type of policy overrides any allow permissions.
4. **Permissions Boundaries**:
    - Permissions boundaries are set at the IAM entity level (user or role) and define the maximum permissions an identity-based policy can grant.
    - Example: A user can access only Amazon S3, CloudWatch, and EC2, as allowed by their permissions boundary.
5. **Evaluating Permissions Across Policies**:
    - The final permissions are the intersection of all policy types (SCPs, identity-based, resource-based, permission boundaries).
    - An explicit deny from any policy type will prevent access, regardless of other permissions.
6. **Comparing SCPs and Permissions Boundaries**:
    - SCPs apply at an organizational level, while permission boundaries apply to IAM entities (users or roles).
    - Both are used to limit permissions, but neither grants access on its own.
7. **AWS Control Tower**:
    - AWS Control Tower simplifies the setup and governance of multi-account AWS environments with built-in best practice blueprints and guardrails for security, operations, and compliance.

Section 4

his content provides a detailed overview of **AWS Key Management Service (KMS)**, its features, and its integration with other AWS services like **Amazon S3** and **Amazon Elastic Block Store (EBS)**. Here's a summary of the key points:

AWS KMS Overview:

- **AWS KMS** is a managed service for creating and controlling cryptographic keys to encrypt and decrypt data. It uses **FIPS 140-2 validated hardware security modules (HSMs)** to secure keys and integrates with various AWS services.
- Key features include the ability to create **data keys**, rotate them automatically, set usage policies, and log operations through **AWS CloudTrail**.

Key Types:

1. **Customer Managed Keys (CMKs)**: Keys created and managed by you in your AWS account.
2. **AWS Managed Keys**: Keys created and managed by AWS services.
3. **Data Keys**: Symmetric keys used for encrypting large data outside AWS KMS.

4. **Asymmetric Keys**: Consisting of public and private key pairs, where private keys never leave AWS KMS.

Cryptographic Operations:

- **Encrypt**: Uses KMS keys to encrypt data (up to 4,096 bytes).
- **Decrypt**: Decrypts ciphertext encrypted by a KMS key.
- **GenerateDataKey**: Returns symmetric keys for use outside AWS KMS.
- **GenerateDataKeyPair**: Returns asymmetric key pairs for external use.

Integration with AWS Services:

- **Amazon S3**: Server-side encryption (SSE-KMS) to encrypt objects in S3 buckets using KMS keys. The encrypted key is stored in the object metadata.
- **Amazon EBS**: Encrypted EBS volumes use AES-256-XTS encryption, where KMS keys encrypt data at rest transparently for EC2 instances.

Encryption Concepts:

- **Symmetric Encryption**: Same key is used to encrypt and decrypt data.
- **Asymmetric Encryption**: Uses a public key for encryption and a private key for decryption.
- **Envelope Encryption**: Data is encrypted with a data key, which is further encrypted under a master key

Section 5

AWS Services for Security, Identity, and Compliance

Category	Category description	Examples
Identity and access management	Securely manage identities, resources, and permissions at scale.	AWS Identity and Access Management (IAM) AWS IAM Identity Center Amazon Cognito AWS Organizations
Detection and response	Enhance security posture and streamline security operations across an entire AWS environment.	AWS CloudTrail Amazon Detective Amazon Inspector AWS Security Hub
Network and application protection	Enforce fine-grained security policies at network control points across an organization.	AWS Network Firewall AWS Shield AWS WAF
Data protection	Protect data, accounts, and workloads from unauthorized access.	AWS Key Management System (AWS KMS) AWS Secrets Manager Amazon Macie
Compliance	Get a comprehensive view of compliance status and continuously monitor using automated checks based on AWS best practices and industry standards.	AWS Artifact AWS Audit Manager

The provided text details various AWS services designed to enhance security, identity management, and compliance for applications hosted on the cloud. Here's a concise summary of the key AWS services mentioned across different categories:

1. **Identity and Access Management**:
    - **AWS IAM**: Manage permissions and control access to AWS resources.
    - **AWS IAM Identity Center**: Centralized identity management for large organizations.
    - **Amazon Cognito**: Authentication and authorization for mobile and web apps.
    - **AWS Organizations**: Manage multiple AWS accounts securely.
2. **Detection and Response**:
    - **AWS CloudTrail**: Monitor API calls and track user activity.
    - **Amazon Detective**: Investigate and analyze security issues with machine learning.
    - **Amazon Inspector**: Scan AWS workloads for vulnerabilities.
    - **AWS Security Hub**: Centralize security findings across AWS services for enhanced monitoring.
3. **Network and Application Protection**:
    - **AWS Network Firewall**: Secure and monitor network traffic.
    - **AWS Shield**: DDoS protection for your applications.
    - **AWS WAF**: Monitor and protect against web application attacks (HTTP/HTTPS).
4. **Data Protection**:
    - **AWS Key Management Service (KMS)**: Manage encryption keys for secure data storage.
    - **AWS Secrets Manager**: Securely store and manage sensitive information.
    - **Amazon Macie**: Detect and protect sensitive data using machine learning.
5. **Compliance**:
    - **AWS Artifact**: Access AWS compliance documentation.
    - **AWS Audit Manager**: Automate audits and track compliance processes.

Additional services like **AWS Trusted Advisor** provide recommendations for optimizing AWS resources, including enhancing security, improving performance, and reducing costs. **AWS Security Hub** integrates findings from other services and provides a comprehensive view of security trends across AWS accounts.

These services collectively help you implement a **defense in depth** strategy, ensuring multi-layered security for your cloud infrastructure.

Module Checks

KEYBOARD NAVIGA

1. Which statement about Amazon EC2 Auto Scaling is accurate?

- It requires the customer to purchase Reserved Instances.
- **It can launch Amazon Elastic Compute Cloud (Amazon EC2) instances in multiple Availability Zones.**
- It can launch Amazon Elastic Compute Cloud (Amazon EC2) instances, but customers must terminate instances after they are no longer needed.
- It can only launch new Amazon Elastic Compute Cloud (Amazon EC2) instances based on a schedule.

2. You detected that the demand on a fleet of Amazon Elastic Compute Cloud (Amazon EC2) instances in an Auto Scaling group increases by a set amount each day. Which type of scaling is the most appropriate for this scenario?

- **Scheduled**
- Dynamic
- Predictive
- Manual

3. A fleet of Amazon Elastic Compute Cloud (Amazon EC2) instances are launched in an Auto Scaling group behind an Elastic Load Balancing load balancer. The EC2 instances must maintain 50 percent average CPU utilization. Which type of scaling provides the simplest way to achieve this requirement?

- ○ Step scaling
- ○ Simple scaling
- ◉ Target tracking scaling
- ○ Manual scaling

4. How do you vertically scale an Amazon Relational Database Service (Amazon RDS) database?

- ○ By adding read replicas
- ○ By creating dedicated read and write nodes
- ○ By sharding the database
- ◉ By changing the instance class

5. How do you horizontally scale an Amazon Aurora database?

- ◉ By adding Aurora Replica instances
- ○ By increasing the size of the buffer cache configuration
- ○ By creating Amazon CloudWatch alarms
- ○ By changing the instance type

6. How does Amazon DynamoDB perform automatic scaling?

- It adds and removes database instances in response to changes in traffic.
- It adds read replicas in response to increased read demand.
- It adjusts the provisioned throughput capacity in response to traffic patter
- It changes the instance type in response to changes in processing load.

7. A fleet of Amazon Elastic Compute Cloud (Amazon EC2) instances launch in an Auto Scaling group. The instances run an application that uses a custom protocol on TCP port 42000. Connections from client systems on the internet must balance across the instances. Which load balancing solution ensures the highest availability?

- Round-robin DNS
- Application Load Balancer
- Network Load Balancer
- Instance-based load balancer

KEYBOARD NAVIGATION

8. Users in location A connect to an application in Region A. Users in location B connect to the same application in Region B. If the application in Region A becomes unhealthy, clients in location A must be redirected to the application in Region B. Which solution can meet this requirement?

- Use an Application Load Balancer with Amazon CloudWatch alarms.
- Use geolocation routing with failover records in Amazon Route 53.
- Use latency-based routing in Amazon Route 53 with Amazon CloudWatch alarms.
- Use geoproximity routing and a Network Load Balancer that is attached to both Regions.

KEYBOARD NAVIGATION

9. A company must build a highly available website that uses server-side scripts to serve dynamic HTML. Which solution provides the highest availability for the least cost and complexity?

- (●) An Auto Scaling group launches Amazon EC2 instances, which are served by an Application Load Balancer. DNS name resolution points to the load balancer.
- ( ) Amazon S3 hosts the website. DNS name resolution points to the S3 bucket.
- ( ) An Auto Scaling group launches Amazon EC2 instances, which are served by a Network Load Balancer. Amazon Route 53 uses latency-based routing.
- ( ) A second web server is deployed in another Region. Amazon Route 53 uses failover routing for disaster recovery (DR).

KEYBOARD NAVIGATION

10. You have created an Amazon Web Services (AWS) account for your own personal development and testing. You want your account to stay within the AWS Free Tier and to not generate unexpected costs. Which approach will work and requires the least effort?

- ( ) Log in to the AWS Management Console each month and check your billing dashboard.
- ( ) Create a service control policy (SCP) to restrict all services that are not included in the AWS Free Tier.
- (●) Create an Amazon CloudWatch alarm to send you an email message when the account billing exceeds $0.

MODULE-10

Monitoring your resources:

Importance of Monitoring

- Ensures **operational health**, **resource utilization**, and **application performance**.
- Timely responses are essential under heavy loads, focusing on minimizing latency.
- A low-latency error-handling mechanism is crucial for quick issue identification and resolution.
- Debugging distributed applications can be challenging due to state reproduction difficulties.

Solutions for Monitoring

1. **Centralized Logging**:
    - Consolidate logs from individual services into a central logging system for easier access and analysis.
2. **Key Metrics**:
    - **Operational Health Metrics**: Ensure the runtime environment functions correctly.
    - **Resource Utilization Metrics**: Monitor resource usage (CPU, memory, network).
    - **Application Performance Metrics**: Measure response times and transaction success rates.

AWS Monitoring with CloudWatch

- **Log Collection**: AWS services send logs to CloudWatch for centralized monitoring.
- **Metric Monitoring**: Metrics are organized by namespaces (e.g., AWS/EC2) and can include dimensions (e.g., InstanceId).
- **Alarms**:
    - Trigger actions when metrics breach thresholds over a specified time.
    - Enable automated responses (e.g., Auto Scaling).
- **Dashboards**:
    - Create customizable views for comparing metrics across regions.
    - Visualize application health in real time.

EventBridge Integration

- Routes events when CloudWatch alarms are triggered.
- Supports serverless, event-driven architectures to enable scaling and resilience.

Cost Monitoring

- **AWS Cost Explorer**: Visualizes and manages AWS costs and usage over time.
- **AWS Budgets**: Sets custom budgets and alerts for spending and usage limits.
- **AWS Cost and Usage Reports**: Provides detailed cost and usage data, including metadata about AWS services.

Scaling your compute resources:

The Need for Reactive Architectures

Modern applications must handle large volumes of data with sub-second response times and near-100% uptime. Reactive architectures help meet these demands by ensuring applications are elastic, resilient, responsive, and message-driven:

- **Elasticity**: The infrastructure can dynamically expand or contract based on capacity needs, adding resources during traffic spikes and reducing them during quiet periods.
- **Responsiveness**: Applications remain timely and low-latency under varying workloads.
- **Resilience**: They recover quickly from failures, maintaining responsiveness despite load or component failures.
- **Message-driven**: Asynchronous message-passing ensures loose coupling and location transparency.

Achieving Elasticity through Scaling

Elasticity allows infrastructure to adjust resources based on demand:

- **Vertical Scaling**: Involves upgrading a resource (e.g., larger server). This can lead to downtime.
- **Horizontal Scaling**: Adds or removes resources (e.g., servers) to support applications without downtime.

Amazon EC2 Auto Scaling

Amazon EC2 Auto Scaling manages groups of EC2 instances across Availability Zones, automatically adjusting capacity based on demand:

- **Better Fault Tolerance**: Detects unhealthy instances and replaces them.
- **Better Availability**: Ensures the right capacity is always available.
- **Better Cost Management**: Dynamically adjusts capacity to save costs.

EC2 Auto Scaling Group Components

- **Minimum Capacity**: The smallest number of instances needed.
- **Maximum Capacity**: The largest number permitted.
- **Desired Capacity**: The optimal number of instances under normal conditions.

Scaling Policies

- **Dynamic Scaling**: Adjusts based on metrics from services like CloudWatch.
- **Predictive Scaling**: Uses historical data to anticipate future capacity needs.

More AWS Scaling Options

- **AWS Auto Scaling**: Configure auto scaling for multiple resources (e.g., EC2, Aurora).

- **AWS Application Auto Scaling**: Automatically scales resources beyond EC2, including Lambda and DynamoDB.

Scaling your databases:

Scaling AWS Databases

1. Amazon Aurora

- **Cluster Scaling**:
  - **Vertical Scaling**: Change the instance type or size.
  - **Horizontal Scaling**: Use up to 15 Aurora Replicas for read-only access.
  - **Auto Scaling**: Aurora Auto Scaling dynamically adjusts the number of read replicas based on workload.
- **Serverless**:
  - Automatically scales compute capacity with minimum and maximum Aurora Compute Units (ACUs). It handles intermittent workloads by starting and stopping clusters as needed.

2. Amazon RDS

- **Database Scaling**:
  - **Vertical Scaling**: Change instance class and storage type/size; supports Amazon RDS storage auto scaling.
  - **Horizontal Scaling**: Add read replicas for offloading read traffic.
- **Instance Changes**: Modifications can cause temporary unavailability during maintenance.

3. Amazon DynamoDB

- **Table Scaling**:
  - **Horizontal Scaling**: Scale read capacity units (RCUs) and write capacity units (WCUs) separately.
  - **On-Demand Mode**: Automatically scales based on actual workload; adapts to peak traffic.
  - **Provisioned Mode**: Set expected throughput with auto scaling to adjust capacity dynamically.
- **Global Secondary Indexes**: Each has its own provisioned throughput capacity that can be managed independently.

4. Comparison of Scaling Mechanisms

Service	Vertical Scaling	Horizontal Scaling	Storage Scaling
Aurora	Instance class or size	Aurora Auto Scaling for read replicas	Managed by AWS
Aurora Serverless	ACU throughput limits	Not applicable	Managed by AWS
Amazon RDS	Instance class, storage auto scaling	Read replica databases	Amazon RDS storage auto scaling
DynamoDB	N/A	On-demand mode, provisioned mode with auto scaling	Managed by AWS

Highly Available Systems

Using load balancers to create highly available environments

Uptime and Downtime Percentages:

- **90%:** 36.5 days/year; 2.4 hours/day
- **99%:** 3.65 days/year; 14 minutes/day
- **99.9%:** 8.76 hours/year; 86 seconds/day
- **99.99%:** 52.6 minutes/year; 8.6 seconds/day
- **99.999%:** 5.25 minutes/year; 0.86 seconds/day

Design Principles:

- Avoid single points of failure (e.g., multiple database and application instances).
- Implement backup plans for instance, Availability Zone, and Regional failures.
- Enable resiliency and minimal downtime with automated recovery mechanisms.

AWS Elastic Load Balancers (ELB)

Functions:

- Distributes incoming traffic across multiple targets (e.g., EC2 instances) in different Availability Zones.
- Monitors health of registered targets through health checks.
- Routes traffic to healthy targets and scales based on incoming traffic.

Types of Load Balancers:

- **Application Load Balancer (ALB):**
  - Operates at OSI Layer 7 (application layer).
  - Routes HTTP and HTTPS traffic based on request content.
- **Network Load Balancer (NLB):**

        - Operates at OSI Layer 4 (transport layer).
        - Handles millions of requests per second with ultra-low latency.

    - **Gateway Load Balancer:**

        - Operates at OSI Layer 3 (network layer).
        - Used for third-party virtual appliance fleets (e.g., firewalls).

    - **Classic Load Balancer:**
        - Operates at both OSI Layer 3 and 7.
        - Supports older EC2-Classic networks.

Load Balancer Components

- **Listeners:** Check for connection requests and route them based on defined rules.
- **Target Groups:** Route requests to registered targets and manage health checks.
- **Health Checks:** Periodically monitor target health; targets must return HTTP status 200 to be considered healthy.

High Availability Architecture

- **Using Application Load Balancer and Amazon RDS Multi-AZ:**

    - ALB distributes traffic across EC2 instances in multiple Availability Zones.
    - Amazon RDS Multi-AZ provides synchronous data replication for high availability.

- **Using Network Load Balancer:**

    - Centralized service exposure; handles connections over VPC peering and AWS managed VPN.
    - Supports static IP addresses and various connection protocols.

Security with Gateway Load Balancer

- **Traffic Inspection:** Scans incoming and outgoing traffic using virtual appliances.
- **GENEVE Protocol:** Maintains flow stickiness for secure traffic management.

Summary

AWS load balancers are critical components for designing highly available and fault-tolerant architectures, providing essential traffic management, monitoring, and scaling capabilities to ensure application resilience and minimal downtime.

Using Route 53 to create highly available environments:

DNS Overview

Function of DNS:

1. Translates human-readable domain names (e.g., www.myweb.com) into IP addresses (e.g., 192.0.2.1).
2. Acts like a phone book for the Internet, facilitating communication between devices.

DNS Query Process:

1. User enters a domain name in a web browser.
2. Request is sent to a DNS resolver (often managed by the ISP).
3. Resolver queries a DNS root name server for TLD information.
4. Resolver contacts the TLD name server for the specific domain.
5. Resolver queries the authoritative name server for the domain to obtain the IP address.
6. IP address is returned to the web browser, which then connects to the web server.

AWS Route 53

Overview of Route 53:

- A scalable DNS web service that manages domain name registrations and provides DNS routing and health checks.
- Connects user requests to resources in AWS (e.g., EC2 instances, ELB, S3) and can route traffic to external resources.

Core Functions:

- **Domain Registration:** Purchase and manage domain names, automatically configuring DNS settings.
- **DNS Routing:** Create hosted zones that contain records for routing traffic.
- **Health Checks:** Monitor the health of resources, performing health checks against IP addresses or domains.

Application Recovery Controller:

- Supports zonal autoshift to automatically reroute traffic away from unhealthy Availability Zones.

Routing Policies in Route 53

- **Routing Types:**
  - **Simple Routing:** Basic DNS record management.
  - **Weighted Routing:** Distributes traffic among multiple resources based on specified weights.
  - **Latency-Based Routing:** Routes user requests to the AWS Region with the lowest latency.
  - **Failover Routing:** Directs traffic to healthy resources when a primary resource is unhealthy.
  - **Geoproximity Routing:** Routes based on geographical location.
  - **Geolocation Routing:** Routes based on the geographic location of users.
  - **Multivalue Answer Routing:** Returns multiple values for DNS queries, providing failover options.

Multi-Region Failover

-

Active-Passive Configuration:

  - Primary resources handle traffic normally; secondary resources are on standby.
  - Route 53 checks the health of resources and directs traffic accordingly.

Health Checks:

  - Monitor resource health in simple and complex configurations, ensuring only healthy resources respond to DNS queries.

Summary

AWS Route 53 is a powerful DNS service that facilitates domain management, traffic routing, and resource health monitoring. By leveraging various routing policies and health checks, Route 53 helps ensure high availability and fault tolerance in cloud architectures.

Applying AWS Well-Architected Framework principles to highly available systems

The AWS Well-Architected Framework provides a comprehensive set of best practices for building highly available systems. Here's a summary of the key practices from the reliability and performance efficiency pillars that focus on achieving high availability:

Best Practices for Highly Available Systems

1. Failure Management: Use Fault Isolation to Protect Your Workload

- **Deploy to Multiple Locations**:
  - Distribute workload data and resources across multiple Availability Zones (AZs) or AWS Regions to avoid single points of failure. This ensures that if one zone or resource fails, others can continue to operate.
- **Automate Recovery**:

  - Implement automation to recreate infrastructure and redeploy applications if they are constrained to a single location. Use services like Amazon EC2 Auto Scaling to ensure applications can recover automatically from failures.

2. Design Your Workload to Withstand Component Failures

- **Fail Over to Healthy Resources**:

  - Distribute traffic across multiple resources, AZs, or Regions. If a failure occurs, ensure that traffic is rerouted to healthy resources to maintain availability. This can involve using AWS managed services that inherently support multi-AZ deployments.

- **Send Notifications on Availability Impact**:

- Set up monitoring to track performance metrics and send alerts when thresholds are breached. This allows operations teams to respond quickly to issues, minimizing user impact.

3. Compute and Hardware Best Practices

- **Dynamic Scaling of Compute Resources**:
  - Utilize AWS services that allow dynamic scaling of resources to match changing demand. This can include:
    - **Target-tracking Scaling**: Automatically adjusts capacity based on predefined metrics.
    - **Predictive Scaling**: Anticipates demand based on historical trends.
    - **Schedule-based Scaling**: Manually sets scaling schedules based on expected load.
    - **Service Scaling**: Use serverless architectures that inherently scale as demand fluctuates.

Key AWS Services Supporting These Best Practices

- **Amazon EC2 Auto Scaling**: Automatically adjusts the number of EC2 instances based on demand.
- **Amazon RDS Multi-AZ Deployments**: Provides high availability for databases.
- **Elastic Load Balancing (ELB)**: Distributes incoming application traffic across multiple targets, ensuring even load distribution.
- **Amazon CloudWatch**: Monitors resources and applications, providing insights into performance and setting alarms for certain thresholds.

Conclusion

Implementing these best practices in your cloud architecture enhances system reliability and performance, ensuring that your applications remain highly available and resilient against failures. For a more comprehensive guide, refer to the AWS Well-Architected Framework documentation.

1. Which statement about Amazon EC2 Auto Scaling is accurate? (●) It can launch Amazon EC2 instances in multiple Availability Zones. ( ) It requires the customer to use Reserved Instances only. ( ) It can launch Amazon EC2 instances, but customers must terminate instances after they are no longer needed. ( ) It can launch new Amazon EC2 instances based on a schedule.	

2. A devops engineeer detected that the demand on a fleet of Amazon EC2 instances in an Auto Scaling group increases by a set amount on weekend days. Which type of scaling is the MOST appropriate in this case?

- ○ Manual
- ○ Dynamic
- ◉ Scheduled
- ○ Predictive

3. A devops engineer launches a fleet of Amazon EC2 instances in an Auto Scaling group behind an Application Load Balancer. The EC2 instances must maintain 50 percent average CPU utilization. Which type of scaling is appropriate to use based on CPU utilization usage?

- ◉ Target tracking scaling
- ○ Manual scaling
- ○ Step scaling
- ○ Simple scaling

4. How can a user vertically scale an Amazon RDS database?

- ○ By creating dedicated read and write nodes
- ◉ By changing the instance class or size
- ○ By adding read replicas
- ○ By sharding the database

5. How can an AWS customer horizontally scale an Amazon Aurora database

- ○ By creating a scaling policy
- ◉ By adding Aurora Replica instances by using Aurora Auto Scaling
- ○ By changing the instance type
- ○ By creating Amazon CloudWatch alarms

6. How does Amazon DynamoDB perform automatic scaling?

- ◉ It adjusts the provisioned throughput capacity in response to traffic patterns.
- ○ It adds and removes database instances in response to changes in traffic.
- ○ It adds read replicas in response to increased read demand.
- ○ It changes the instance type in response to changes in processing load.

Correct

DynamoDB provisioned mode uses Amazon CloudWatch and AWS Application Auto Scaling to adjust the provisioned throughput capacity of a table within minimum and maximum limits. DynamoDB on-demand mode scales a table based on actual reads and writes without any limits.

Continue

7. A fleet of Amazon EC2 instances is launched in an Amazon EC2 Auto Scaling group. The instances run an application that uses a custom protocol on TCP port 42000. Connections from client systems on the internet must balance across the instances. Which load balancing solution is the best solution?

- ○ Application Load Balancer
- ○ Classic Load Balancer
- ◉ Network Load Balancer
- ○ Gateway Load Balancer

Correct

A Network Load Balancer can handle TCP, UDP, and TLS network traffic protocols. As a feature of Elastic Load Balancing (ELB), a Network Load Balancer is highly available.

8. A company must build a highly available website that uses server-side scripts to serve dynamic HTML. Which solution provides the HIGHEST availability for the LEAST cost and complexity? ○ An Auto Scaling group launches Amazon EC2 instances, which are served by a Network Load Balancer. Amazon Route 53 uses latency-based routing. ◉ An Auto Scaling group launches Amazon EC2 instances, which are served by an Application Load Balancer. DNS name resolution points to the load balancer. ○ Amazon S3 hosts the website. DNS name resolution points to the S3 bucket. ○ The customer deploys a second web server in another Region. Amazon Route 53 uses failover routing for disaster recovery (DR).	The Auto Scaling group can automatically launch instances to handle increased load and can terminate instances when they become unhealthy or when the load decreases. The Application Load Balancer is highly available and distributes the load across the instances.
9. Users in location A connect to an application in Region A. Users in location B connect to the same application in Region B. If the application in Region A becomes unhealthy, traffic for location A must be redirected to the application in Region B. Which solution meets this requirement? ◉ Use geolocation routing with failover records in Amazon Route 53. ○ Use geoproximity routing and a Network Load Balancer that is attached to both Regions. ○ Use latency-based routing in Amazon Route 53 with Amazon CloudWatch alarms. ○ Use an Application Load Balancer with Amazon CloudWatch alarms.	Geolocation routing enables the separation of traffic based on location. A failover record that points to the application in Region B enables failover if the application in Region A becomes unhealthy.
10. A software engineer has created an AWS account for their own personal development and testing. They want the account to stay within the AWS Free Tier and to not genera unexpected costs. Which approach will work and will require the LEAST effort? ○ Sign in to the AWS Management Console each month and check the billing dashboard. ○ Create an Amazon CloudWatch metric to monitor account billing and limit it to $0. ◉ Create an Amazon CloudWatch alarm to send an email message when the account billing exceeds $0. ○ Create a service control policy (SCP) to restrict all services that are not included in the AWS Free Tier.	Create an alarm for the estimated charges and receive a notification when the estimated charges exceed the AWS Free Tier limit.
• CloudWatch alarms can send notifications to Amazon EC2 Auto Scaling and SNS topics. • CloudWatch collects logs and metrics from AWS services across Regions. • You can use CloudWatch dashboards to visualize metrics and alarms. • EventBridge processes and routes event with an event bus or a pipe. • AWS Cost Explorer, AWS Budgets, and AWS Cost and Usage Report can help you understand and manage the cost of your AWS infrastructure.	• With Amazon EC2 Auto Scaling, you can create a group to manage a logical collection of EC2 instances, called an Amazon EC2 Auto Scaling group. • A group has capacity settings that specify the minimum, maximum, and desired number of instances required to run an application. • Group size can be scaled in and out with schedule actions, dynamic policies, and predictive policies. • To scale more services than EC2 instances, use AWS Auto Scaling or Application Auto Scaling.

• With Aurora, you can choose the database instance class size and number of Aurora Replicas. • Aurora Serverless scales resources automatically based on the minimum and maximum capacity specifications. • You can manually vertically scale compute capacity for your RDS DB instance. • You can use read replicas to horizontally scale your RDS DB instance. • DynamoDB On-Demand offers a pricing model based on actual table reads and writes. • DynamoDB auto scaling uses Application Auto Scaling to dynamically adjust provisioned throughput capacity.	• ELB distributes traffic across multiple targets in one or more Availability Zones and monitors the health of registered targets with health checks. • An Application Load Balancer is used for application architectures and operates at the OSI model application layer (layer 7). • A Network Load Balancer is used for millions of concurrent, ultra-low latency requests and operates at the OSI model transport layer (layer 4). • A Gateway Load Balancer is used to improve security, compliance, and policy controls and operates at the OSI model network layer (layer 3).

• Route 53 is a DNS service that does the following:
  • Manages domain name registrations
  • Provides hosted zones and authoritative name servers
  • Performs DNS routing and health checks
• Route 53 supports multiple routing options, including the following:
  • Simple routing
  • Weighted routing
  • Latency routing
  • Failover routing
  • Geoproximity routing
  • Geolocation routing
  • Multivalue answer routing
  • IP-based routing
	• Deploy the workload to multiple locations.
	• Automate recovery for components constrained to a single location.
	• Fail over to healthy resources.
	• Send notifications when events impact availability.
	• Scale your compute resources dynamically.

Sample exam question: Response choices

A web application gives customers the ability to upload orders to an Amazon S3 bucket. The resulting Amazon S3 events initiate an AWS Lambda function that inserts a message into an Amazon Simple Queue Service (Amazon SQS) queue. A single Amazon EC2 instance reads the messages from the queue, processes them, and stores them in an Amazon DynamoDB table partitioned by unique order ID. Next month, traffic is expected to increase by a factor of 10, and a solutions architect is reviewing the architecture for possible scaling problems. Which component is MOST likely to need re-architecting to be able to scale to accommodate the new traffic?

Choice	Response
A	Lambda function
B	SQS queue
C	EC2 instance
D	DynamoDB table

C EC2 instance

MODULE-11

Automating Your Architecture

· **Long Manual Process to Build an Architecture**

- Organizations often begin AWS with manual tasks like creating S3 buckets or launching EC2 instances.
- Manually adding resources over time makes managing them difficult.
- Manual processes are error-prone, unreliable, and limit agility.

· **Key Questions: Manual vs. Automation**

- Should you focus on design or implementation?
- What are the risks of manual implementations?
- How will you update production servers and handle deployments across regions?
- How will you manage rollback when issues occur?
- How will you debug, manage dependencies, and ensure fixes stay in place?
- Is manual configuration realistic for all tasks?

· **Risks from Manual Processes**

- Lacks scalability, version control, and audit trails.
- Inconsistent configurations create risks.
- Difficulty replicating deployments across regions.
- Challenges in rolling back to previous versions during emergencies.
- Lack of tracking for configuration changes.

· **Benefits of Automation**

- Reduces manual intervention and increases reproducibility.
- Improves productivity by automating testing, scaling, and bug fixing.
- Automation ensures consistency and supports rapid setup.
- Allows automatic scaling to meet demand and maintains standardized configurations.

Using infrastructure as code

1. Infrastructure as Code (IaC) Overview

- IaC is the process of writing human-readable and machine-consumable templates to manage cloud resources.
- It enables replicating, redeploying, and repurposing infrastructure.
- IaC is essential to control costs, reduce risks, and respond quickly to business opportunities.

2. IaC Benefits

- Enables rapid deployment of complex environments with consistent configurations.
- Changes made in the template are propagated to all environments (e.g., dev, test, production).

- Improves reusability, repeatability, and maintainability.
- Simplifies cleanup by deleting resources when no longer needed.

3. CloudFormation

- CloudFormation simplifies modeling, creating, and managing AWS resources in a repeatable, orderly manner.
- Allows version control and rollback to previous resource configurations.
- Supports building complex infrastructure stacks and automating the provisioning process.

4. AWS Services Utilizing CloudFormation

- **AWS Elastic Beanstalk**: Deploys and manages applications without handling underlying infrastructure.
- **AWS Quick Starts**: Provides automated reference architectures using CloudFormation templates.
- **AWS SAM**: Extension of CloudFormation optimized for serverless applications.
- **AWS Amplify**: Simplifies building full-stack web and mobile apps with backend integration.
- **AWS CDK**: Uses modern programming languages to define cloud infrastructure, deploying resources via CloudFormation.

Customizing with CloudFormation:

AWS CloudFormation automates the provisioning of resources in AWS. Here's how it works:

Template Creation: You define resources (like EC2 instances, S3 buckets, or load balancers) in a CloudFormation template. This template can be created from scratch or by using prebuilt ones, and can be written in JSON or YAML.

Template Upload: The template is either uploaded to AWS CloudFormation or stored in Amazon S3, with CloudFormation pointing to the S3 location.

Stack Creation: You initiate the creation of a stack, which is a collection of AWS resources defined in your template. CloudFormation reads the template and provisions resources across multiple AWS services in your account.

Stack Management: The stack retains control of the created resources. You can later update the stack, detect changes (drift detection), or delete it, which automatically deletes the related resources.

CloudFormation Template Format

- **JSON**: Compact and widely used by systems and APIs. Easier to generate and parse by machines but harder for humans to debug.
- **YAML**: Human-readable and optimized for simplicity, with fewer symbols than JSON. It supports embedded comments and is easier to debug.

Sections of a CloudFormation Template

A template consists of several key sections, but only the **Resources** section is required. Here's the recommended order for the sections:

1. **AWSTemplateFormatVersion**: Specifies the CloudFormation template version.
2. **Description**: Optional text description of the template.
3. **Metadata**: Optional metadata about the template.
4. **Parameters**: Parameters that can be passed at runtime to customize stack creation.
5. **Mappings**: Conditional values, like a lookup table, for dynamic resource creation.
6. **Conditions**: Optional conditions that control whether certain resources are created.
7. **Transform**: Used for serverless applications to include AWS SAM transformations.
8. **Resources**: Defines the AWS resources to create (required).
9. **Outputs**: Describes the values returned after stack creation, such as an instance ID.

AWS CloudFormation Designer
This is a visual tool for creating, viewing, and editing CloudFormation templates. You can drag-and-drop resources and see their relationships. It supports both JSON and YAML and allows you to convert between formats.

CloudFormation Features

- **Change Sets**: Allows you to preview changes to a stack before applying them.
- **Drift Detection**: Identifies if a stack's resources have deviated from the original template (e.g., manual changes made outside of CloudFormation).
- **Conditions**: You can use the same template for different environments (e.g., production vs. development), with conditional logic controlling resource creation.

CloudFormation is highly versatile, enabling efficient and repeatable resource provisioning across AWS services, while ensuring consistency between environments.

AWS Quick Starts

AWS Quick Starts are pre-built, customizable, and modular deployments designed based on AWS best practices for security and high availability. These Quick Starts can help you deploy complete architectures within an hour using AWS CloudFormation templates, which embody the concept of infrastructure as code. Here's a breakdown of key features and usage of AWS Quick Starts:

Key Features of AWS Quick Starts:

1. **Gold-standard deployments**: Designed by AWS architects and partners to meet the highest standards of security and availability.
2. **Based on AWS Best Practices**: Follow best practices for infrastructure design, ensuring security, scalability, and high availability.
3. **Fast Deployment**: Enable full architecture deployment in less than an hour, ideal for both production and experimentation.
4. **Customizable**: You can adapt the deployment to suit specific needs by modifying the CloudFormation templates.
5. **CloudFormation as the Foundation**: Provides you with predefined infrastructure code, eliminating the need to build templates from scratch.

Usage of AWS Quick Starts:

- **CloudFormation Templates**: Each Quick Start comes with a CloudFormation template and a detailed deployment guide that explains configuration options, security, and cost estimates.
- **Experimentation and Architecture Development**: Quick Starts can be used as a reference for developing your own infrastructure templates by borrowing sections from existing templates.
- **Deployment Example**: The provided templates and guides help create architecture configurations like serverless image processing, complete with features such as caching with Amazon CloudFront, API management with API Gateway, and image handling via AWS Lambda.

Example Quick Start Architecture:

- **Use Case**: A serverless architecture for dynamic image processing with content moderation and smart cropping using Amazon Rekognition.
- **Components**:

    1. **Amazon CloudFront**: Provides a caching layer, improving cost efficiency and reducing latency.
    2. **API Gateway**: Handles API endpoints to initiate Lambda functions.
    3. **AWS Lambda**: Processes images retrieved from an Amazon S3 bucket, modifying them with open-source image processing tools.
    4. **S3 Buckets**: One bucket for image storage and another for logs.
    5. **AWS Secrets Manager**: Handles secret management if URL signatures are used.
    6. **Amazon Rekognition**: Offers smart image analysis, such as content moderation and cropping.

These Quick Starts help streamline deployment, reduce setup time, and ensure alignment with AWS best practices for building secure, scalable, and efficient cloud solutions.

Customizing with Amazon Q Developer

Key Challenges of Writing Infrastructure as Code (IaC)

1. **Human Error**: Mistakes in code are common and can lead to misconfigurations.
2. **Differing Skill Levels**: Developers have varying abilities, leading to inconsistent code quality.
3. **Complexity**: Templates for even basic architectures can become large and difficult to manage.
4. **Security Vulnerabilities**: Overlooking security can introduce risks into infrastructure.

Amazon Q Developer Features

- **Code Generation and Suggestions**: Helps write, modify, or complete CloudFormation templates.
- **Security Scanning**: Identifies vulnerabilities in code early.
- **No-code Solutions**: Generates infrastructure code from natural language descriptions.
- **Debugging and Optimization**: Assists in troubleshooting and performance improvements.

Example of Using Amazon Q Developer with CloudFormation:

- **CloudFormation YAML Templates**: When starting to define resources, Amazon Q Developer suggests relevant snippets that the developer can accept, saving time and effort. It seamlessly integrates these snippets into the existing template in either YAML or JSON format.

Applying AWS Well-Architected Framework Principles to Automation

Operational Excellence

1. **Perform Operations as Code**: Treat infrastructure operations like application code to reduce errors and improve consistency.
2. **Frequent, Small, Reversible Changes**: Apply small, reversible updates to improve agility and reduce the risk of large failures.
3. **Fully Automate Integration and Deployment**: Automate deployments to ensure consistent, secure, and compliant processes.

Security

1. **Automate Security Best Practices**: Use automated security controls to reduce human error and improve the speed of security implementations, such as automating threat detection.

Reliability

1. **Deploy Changes with Automation**: Use automation to reduce manual intervention when deploying changes, minimizing risks and ensuring controlled updates.
2. **Automate Resource Scaling**: Automate the scaling of resources to handle increased load and replace impaired infrastructure efficiently.

Cost Optimization

1. **Automate Operations**: Automate repetitive tasks to minimize manual interventions, reduce operational costs, and enhance productivity.

1. Which are reasons to use automation to provision resources? (Select TWO.) ☑ Lack of version control with manual processes ☐ Greater expense with manual processes ☐ Automation requirement for high availability ☑ Alignment with the reliability design principle ☐ Automation requirement for creating some resources	With automation, you can introduce version control to control updates and track version history. Automation aligns with the reliability design principle, which requires the use of automation to manage change.

2. Which are benefits of using infrastructure as code (IaC) over manual processes? (Select TWO.) ☑ Deploy environments with configuration consistency. ☐ Automate system-wide security scans. ☑ Propagate updates from a single environment to all environments. ☐ Manage all account users. ☐ Protect environments from deletion.	With IaC, you can deploy environments with consistency and propagate updates across environments. IaC aligns with the reliability design principle, which requires the use of automation to manage change.
3. A cloud architect wants to quickly set up a secure implementation of an Amazon FSx for Windows File Server that follows AWS best practices. Which solution should they use? ○ AWS CloudFormation Designer ○ An AWS CloudFormation template that was downloaded from the internet ◉ An AWS Quick Start ○ An Amazon Machine Image (AMI) on AWS Marketplace	AWS solutions architects and AWS Partners build Quick Starts to help you deploy popular technologies on AWS. They are based on AWS best practices for security and high availability.
4. What is Amazon Q Developer? ○ A template for rapid application deployment ○ An integrated development environment (IDE) ◉ An artificial intelligence (AI)-powered coding companion ○ A set of automated reference architectures	Amazon Q Developer is an AI-powered code generator that integrates with your IDE.
5. Which are reasons to use Amazon Q Developer? (Select TWO.) ☑ Enhance application security. ☑ Accelerate coding tasks. ☐ Write compliance tests. ☐ Share open-source code. ☐ Automate for high availability.	With Amazon Q Developer, you can accelerate coding tasks and enhance application security.
6. What is AWS CloudFormation? ○ A package of all the information that is needed to launch an Amazon EC2 ◉ An AWS service that you can use to create, model, and manage AWS resou ○ A template that describes your infrastructure ○ A description of best practices for designing an AWS implementation	AWS CloudFormation provides a simplified way to model, create, and manage a collection of AWS resources. It enables you to treat infrastructure as code.

7. What is AWS CloudFormation Designer? ○ A collection of reusable templates ◉ A graphical design interface for creating AWS CloudFormation templates ○ A source code repository for AWS CloudFormation templates ○ A tool for automating deployments	You can use AWS CloudFormation Designer to author AWS CloudFormation templates in the AWS Management Console.
8. Which option can be used to accomplish deployment-specific differences in an AWS CloudFormation template? ○ Use change sets. ◉ Use conditions. ○ Use AWS CloudFormation Designer. ○ Use drift detection.	You can use conditions to configure different environments from the same template. This allows for deployment-specific environments that are otherwise configured identically.
9. Which option is a good way to preview changes before implementing them in AWS CloudFormation Designer? ○ Visually inspect the template. ○ Run Detect Drift. ◉ Create a change set. ○ Run Update Stack.	By creating a change set, you can see the changes the template before you apply them.
10. Which option is a good way to know which resources in an application environment were manually modified if the environment was created by running an AWS CloudFormation stack? ○ Run a change set on the stack. ○ Run conditions on the stack. ○ Run a comparison in AWS CloudFormation Designer on the stack. ◉ Run drift detection on the stack.	By running drift detection, you can compare the current stack to the expected template configuration to identify where manual changes were made.
• Manual processes are error prone, unreliable, and inadequate to support an agile business. • Manual processes create risks to applications and environments. • Automation helps you eliminate manual process and build rapidly.	• IaC is the process of provisioning and managing your cloud resources by writing a template file. • IaC rapidly deploys complex environments, and you can build or update the same complex environments repeatedly. • Choose an IaC solution based on your use case, relative balance of convenience and control, and the skills of your team. • CloudFormation is an IaC service to help you create, update, and delete services and architectures.

• CloudFormation is an IaC service that you can use to model, create, and manage a collection of AWS resources. • CloudFormation IaC is defined in templates that are authored in JSON or YAML. • A stack is what you create when you use a template to create AWS resources. • Actions that are available on an existing stack include update stack, detect drift, and delete stack.	• AWS Quick Starts provide CloudFormation templates built by solutions architects and partners that reflect AWS best practices. • AWS Quick Starts consists of a CloudFormation template and a deployment guide, which provide details about deployment options and ways to configure the deployment to match your needs. • AWS Quick Starts can also help you see patterns and practices to accelerate your own template development.
• Amazon Q Developer is a generative AI coding tool that you can use to generate real-time code suggestions. • Amazon Q Developer can offer suggestions for writing CloudFormation templates.	• Best practices related to automating your architecture include the following: • Perform operations as code. • Make frequent, small, reversible changes. • Fully automate integration and deployment. • Automate security best practices. • Deploy changes with automation. • Use automation when obtaining or scaling resources. • Perform automation for operations.

Consider a situation where you want to create a single AWS CloudFormation template that is capable of creating both a production environment that spans two Availability Zones, and a development environment that exists in a single Availability Zone. Which optional section of the CloudFormation template will you want to make use of to configure the logic that will support this?

Choice	Response
A	Conditions
B	Outputs
C	Resources
D	Description

A Conditions

Caching Content:

The primary purpose of a cache is to increase the performance of data retrieval by reducing the need to access the underlying, slower storage layer.

Caching content helps speed up data access by temporarily storing frequently used information in a faster, more efficient way. This reduces the need to repeatedly access slower, primary storage, making your application run faster and more smoothly. It improves performance, saves time, and lowers costs by reusing data that has already been retrieved or processed.

Static and frequently accessed data : content that rarely changes, such as HTML, CSS, JavaScript, images, and video files.

Results of computationally intensive calculations: Results from complex calculations can be stored in a cache to speed up future access.By using an in-memory cache, these time-consuming tasks don't have to be repeated, which improves performance.

Trade-offs to consider when caching

Benefits of caching	Challenges of caching
• Reduces response latency	• Requires additional engineering
• Reduces cost	• Requires someone to determine how to cache data and deal with stale data based on the business use case
• Alleviates the load on the origin (data source)	
• Provides predictable performance	
• Improves availability	

The **two types** of caching covered in this module are:

Static Caching (Edge Caching):

1. Stores content closer to users using a content delivery network (CDN).
2. CDN uses global edge locations to deliver cached content like videos, webpages, and images quickly.
3. Example: AWS CloudFront accelerates content delivery using its network of edge locations.

Database Caching:

1. Acts as a data access layer in front of a database to improve performance.
2. Works with any database type (relational or NoSQL).
3. Common methods include lazy loading and write-through caching.
4. Example: AWS ElastiCache improves web application performance by retrieving data from in-memory stores rather than slower disk-based databases.

CloudFront

1. Content Delivery Network (CDN) Delivery Challenge

- Deliver geographically dispersed content with low latency and low cost for end users.

2. CDN Solution

- A globally distributed system of caching servers.
- Intermediary servers between the client and the application.
- Caches copies of commonly requested files (static content).
- Delivers a local copy of the requested content from a nearby cache edge or point of presence.

3. Challenges in Content Delivery

- Large physical distances between users and applications cause delays.
- Two-way communication increases data transfer times.

4. CDN Caching Capabilities

- Can cache:
    - Images.
    - Videos.
    - Web objects (HTML, CSS, JavaScript).
- Cannot cache:
    - Dynamically generated content.
    - User-generated data.

5. Caching Static Content

- Web objects, image files, and video files can be cached via CDN or edge cache.
- Dynamic and user-generated data cannot be cached but can be delivered via a custom origin, such as an Amazon EC2 instance.

6. AWS CloudFront

- CDN service delivering content globally with low latency and high transfer speeds.
- Improves resiliency from distributed denial of service (DDoS) attacks using AWS Shield and AWS WAF.

7. CloudFront's Performance & Security Features

- Delivers data through edge locations using intelligent routing.
- Supports encryption using HTTPS and the latest TLS versions.
- Provides high performance, security, and content distribution.

8. CloudFront Global Edge Network Components

- **Edge Locations**: Numerous, closer to users, smaller caches.
- **Regional Edge Caches**: Fewer, larger caches, farther from users.

- Caches content at over 550 edge locations and 13 regional edge caches globally.

9. How Caching Works in CloudFront

- User requests content.
- CloudFront checks edge cache and delivers cached content if available.
- If not cached, it fetches content from the origin (e.g., S3 bucket).
- Time to Live (TTL) determines how long content stays in the cache.

10. Steps to Configure CloudFront Distribution

- Specify origin location.
- Create CloudFront distribution.
- CloudFront becomes available with assigned domain name.
- CloudFront sends configuration to edge locations.

11. Controlling Content Caching in CloudFront

- Set maximum TTL value.
- Implement content versioning.
- Specify cache-control headers.
- Use CloudFront invalidation requests to expire content.

12. Using CloudFront for Video Streaming

- Use an encoder to format and package videos.
- Host converted content in an S3 bucket.
- Examples of encoders include AWS Elemental MediaConvert and Amazon Elastic Transcoder.

13. DDoS Mitigation Challenge

- Publicly accessible web content is vulnerable to common web threats.
- DDoS attacks flood networks with traffic to make applications unavailable.

14. CloudFront Solution for DDoS Mitigation

- Monitoring and mitigation built into Amazon Route 53.
- AWS WAF web ACL can block threats before reaching servers.
- AWS Shield protects against DDoS vectors such as UDP reflection attacks.

ElastiCache

Database Cache Benefits:

- **Latency Reduction:** Decrease network and data retrieval latency, improving user experience.
- **Database Offloading:** Relieve pressure on primary databases for better throughput.

- **Cost Efficiency:** In-memory cache can handle high request volumes, reducing the need for costly database scaling.

Caching Trade-offs:

- **Performance vs. Accuracy:** Caching boosts performance but requires strategies to update or invalidate stale data.

ElastiCache Overview:

- **Fully Managed:** Automates setup, monitoring, and failure recovery.
- **Scalable:** Adapts to fluctuating workloads.
- **High Performance:** Sub-millisecond response times using in-memory caching.
- **Cost-effective:** Reduces the frequency of expensive database queries.

ElastiCache Engines:

- **Memcached:**
    - Basic, low-maintenance model.
    - Supports multithreading and horizontal scalability with Auto Discovery.
- **Redis:**
    - Supports advanced data structures, persistence, ranking, and publish/subscribe messaging.
    - Offers high availability with automatic failover.

ElastiCache Components:

- **Nodes:** The smallest unit of an ElastiCache deployment, each with its own DNS name and port.
- **Clusters:** Logical grouping of one or more nodes for Memcached or shards for Redis.
- **Partitioning:**
    - Memcached partitions data across nodes (up to 20).
    - Redis clusters can scale up to 250 nodes, each shard grouping 1-6 nodes.

Time to Live (TTL):

- **TTL Value:** Specifies the expiration time for cached data.
- **Key Expiry:** After TTL, expired keys lead to database queries for refreshed data, preventing stale values.

Caching Strategies:

Lazy Loading:

Process: Loads data into the cache only when it's requested by the application (cache miss).

Advantages:

1. Only caches data that the application actually needs.
2. Avoids unnecessary data in cache, saving storage costs.
3. Reduces load on the primary database for frequently requested data.

Disadvantages:

1. Data can become stale, requiring a strategy for cache invalidation.
2. Cache miss incurs latency, leading to slower response times.
3. Three trips: cache miss -> fetch from database -> update cache -> return to user.

Use Case: Best for data that is read frequently but rarely updated, such as user profiles.

Write-Through:

1 .Process: Updates the cache immediately after data is written to the database.

2. **Advantages**:

    1. Cache is always up-to-date with the latest data.
    2. Reduces the chances of cache misses, improving performance.
    3. Ensures lower latency because the data is already in the cache when needed.

3. **Disadvantages**:

    1. Caching data that might not be used can lead to higher costs (storage of unnecessary data).

4. **Use Case**: Ideal for real-time applications where data needs to be updated instantly and must always be fresh.

Decision Guidelines:

- Use **lazy loading** when you want to minimize cache storage costs and are willing to tolerate occasional cache misses.
- Use **write-through** when you need up-to-date data and cannot afford the penalty of cache misses.

Well-Architected Framework principles to caching

Best Practice Approaches

1. Data Management

Implement Data Access Patterns:

- **Utilize Caching**: Identify databases, APIs, and services that can benefit from caching, especially those with heavy read workloads or high read-to-write ratios.
- **Optimize Query Performance**: Use caching to enhance read latency, throughput, user experience, and overall efficiency.
- **Cache Invalidation Strategy**: Implement time-to-live (TTL) for cached data to balance freshness and reduce load on the backend datastore.
- **Monitor Cache Performance**: Aim for a cache hit rate of 80% or higher; lower rates may indicate inadequate cache size or inefficient access patterns.

Use of Amazon CloudFront:

- Control caching behavior with minimum and maximum TTL settings, content versioning, and cache-control HTTP headers to manage cached content effectively.

In-Memory Databases:

- Use solutions like Amazon ElastiCache for applications needing real-time data access with microsecond latency.

2. Foundations – Plan Your Network Topology

- **Highly Available Network Connectivity**:
  - Use CDNs like Amazon CloudFront to ensure low latency and high availability for public endpoints.
  - Edge locations and Regional edge caches enhance content retrieval speed and improve application resiliency against DDoS attacks.

3. Cost-Effective Resources – Plan for Data Transfer

- **Perform Data Transfer Modeling**:
  - Analyze data transfer requirements and costs to identify optimization opportunities.
- **Optimize Data Transfer Costs**:
  - Use data transfer modeling to pinpoint high-cost areas and implement architectures to minimize data transfer or costs.
- **Implement Services to Reduce Costs**:
  - Leverage CDNs or edge locations for content delivery to users, which minimizes load on servers and improves performance.
  - Utilize CloudFront to cache data at edge locations globally, reducing administrative overhead and increasing efficiency.

Additional Considerations

- **Caching Strategy Selection**:
    - Decide on caching strategies based on when to cache, how to manage stale content, and the potential integration of CDN services.
- **Cost vs. Freshness**:
    - Balance the need for up-to-date data with the costs associated with caching strategies. If the impact of stale data is high, lean towards real-time updating strategies; if lower, lazy loading might suffice.

1. What is caching? ◯ An in-memory database ◯ A global network for content distribution ◯ A way to store passwords ◉ A high-speed data storage layer	A cache is a high-speed data layer for storing frequently accessed data.
2. Which type of data should you cache? ◯ Dynamically generated web content ◯ Specialized data that is needed by a subset of users ◉ Static data that is frequently accessed ◯ Data that can be retrieved quickly with simple queries	Good candidates for caching include data that do not change often and that users access frequently
3. Which is a benefit of caching? ◯ Decreased costs ◯ Load balancing the application ◉ Reduced response latency ◯ Increased application reliability	Caching reduces response latency by reducing or eliminating slow resources between the requester and the content. It can reduce network hops, eliminate the need to communicate over slow links or through slow devices, or reduce load on the origin.
4. Which types of content on a web page can be cached using an edge cache? (Select TWO.) ☑ Web objects, such as hyperlinks ☑ Video files, such as a product demo ☐ Dynamically generated content, such as a user's name ☐ Shopping cart filled with a user's item ☐ User-generated data, such as search terms entered by user	An edge cache can cache static content. Such content might include web objects (for example, HTML documents, CSS style sheets, or JavaScript files), image files, and video files.

5. What does Amazon CloudFront enable?

- ○ Automatic creation of a time to live (TTL) value
- ◉ Multi-tiered and regional caching of content
- ○ Bidirectional caching between users and an origin host
- ○ Transactional processing with an in-memory database

CloudFront provides a multi-tiered cache so that you can have multiple rules for when content expires. It also provides Regional caching, which reduces latency by reducing the number of network hops.

6. How does Amazon CloudFront use edge locations?

- ○ It caches Regional data at Regional edge locations and delivers the content to clients through their Regional edge locations.
- ○ It caches local content at edge locations. It delivers the cached content to clients through the edge location that requires the fewest network hops to reach those clients.
- ◉ It caches frequently accessed content at edge locations. It delivers the cached content to clients through the edge location with the lowest latency to those clients.
- ○ It caches all content from an origin distribution at the edge location and delivers the content to clients through the fastest edge location.

The requested content might be
CloudFront distribution and unav
location. In this case, CloudFront
from the origin and then caches i
location.

7. Which statement describes an efficient way to deliver on-demand video content?

- ◉ Use Amazon S3 to store the content. Then use Amazon CloudFront to deliver the content.
- ○ Launch an Amazon EC2 instance to host your video content. Then use Amazon CloudFront to deliver the content.
- ○ Use Amazon S3 to store and serve the content.
- ○ Launch an Amazon EC2 instance to host and serve your video content.

Storing content in Amazon S3 and then delivering it near the end user by using Amazon CloudFront is an efficient way to deliver on-demand video content.

8. Which role does Amazon CloudFront play in protecting against distributed denial of service (DDoS) attacks?

- ○ Controls traffic by the source IP addresses of requests
- ◉ Routes traffic through edge locations
- ○ Performs deep packet inspection to detect attacks
- ○ Restricts traffic by geography to help block attacks that originate from specific countries

DDoS attacks place a large distributed load on concentrated points. With CloudFront, you can route traffic through edge locations to reduce the load on individual systems.

9. How can an application use Amazon ElastiCache to improve database read performance? (Select TWO.)

- ☐ Direct all read requests to the database, and configure it to read from ElastiCache when a cache miss occurs.
- ☐ Read data from the database first, and write the most frequently read data to ElastiCache.
- ☑ Read data from ElastiCache first, and write to ElastiCache when a cache miss occurs.
- ☑ Write data to ElastiCache whenever the application writes to the database.
- ☐ Replicate the database in ElastiCache, and direct all reads to ElastiCache and all writes to the database.

Writing data to the cache only when a cache miss occurs is called lazy loading. Writing data to the cache every time that the application writes data to the database is called write-through.

10. Which type of caching strategy should be used when there's data that must be updated in real time?

- ○ Cache-control headers
- ○ Lazy loading
- ◉ Write-through
- ○ Time to live (TTL)

A write-through caching strategy is used when you have data that must be updated in real time.

• A cache is a high-speed data storage layer that stores a subset of data so that future requests for that data are served up faster. • Static and frequently accessed data are good candidates for caching. • Caching improves application speed and decreases database cost. • Content is retrieved from the nearest edge location with static caching and retrieved from an in-memory database layer with database caching.	• Static file caching is done by using a CDN. • CloudFront is a CDN service that uses edge locations and Regional edge caches to deliver content securely across the globe. • You can control CloudFront caching behavior by using a combination of actions. • By using CloudFront for distributing content, you are also protecting your systems from DDoS attacks.
• ElastiCache is a key-value, in-memory data store with the purpose of providing millisecond latency and inexpensive access to copies of data. • ElastiCache reduces the complexity of administering your caching systems for Redis and Memcached engines. • Lazy loading and write-through are two strategies to handle keeping the cache as up to date as is appropriate.	• Consider best practices in the performance efficiency pillar such as making selection choices based on data characteristics and available options. • Consider best practices in the reliability pillar such as using highly available network connectivity for your public endpoints. • Consider best practices in the cost optimization pillar such as implementing services to reduce data transfer costs.

Module summary

This module prepared you to do the following:

- Identify how caching content can improve application performance and reduce latency.
- Identify how to use Amazon CloudFront to deliver content by using edge locations protection.
- Create architectures that use CloudFront to cache content.
- Describe how to use ElastiCache for database caching.
- Use the AWS Well-Architected Framework principles when designing caching strategies.

Your company is hosting an ecommerce site and storing the records in an **Amazon RDS database**. After a few days, the Amazon RDS database is experiencing serious performance issues due to website requests. Which option is the most **cost-effective** way to deal with the **performance issues due to many read requests**?

Choice	Response
A	Enable the Multi-AZ feature for the Amazon RDS database.
B	Configure Amazon ElastiCache to cache frequently used content from the database.
C	Place a load balancer in front of the database.
D	Configure Amazon CloudFront to cache content from the database.

B Configure Amazon ElastiCache to cache frequently used content from the database.

MODULE-13:

Building Decoupled Architectures :

Key Concepts

Tight Coupling:

1. Communication is synchronous, leading to direct dependencies between components (e.g., web server, app server).
2. Challenges include system downtime during updates, scaling difficulties, and increased complexity with each new server.

Loose Coupling:

1. Introduces intermediary components (e.g., load balancers) to reduce dependencies.
2. Easier to scale and manage failures since components operate independently.

Microservices Architecture:

1. Breaks down applications into smaller, independent functions (microservices).
2. Each microservice can scale and fail independently, improving resilience and flexibility.

Asynchronous Messaging:

1. Decouples components by using message queues (Amazon SQS) and notification services (Amazon SNS).
2. Components communicate indirectly through messages, allowing for better load management and responsiveness.

Amazon MQ:

1. Facilitates loose coupling between on-premises applications and AWS applications through asynchronous messaging.

Summary

- Loose coupling enhances scalability, reduces complexity, and improves application resilience by isolating components and minimizing direct dependencies. Using AWS services like Amazon SQS, Amazon SNS, and Amazon MQ allows for effective implementation of these principles in cloud architectures.

Decoupling applications with Amazon SQS:

Point-to-Point Messaging

- **Definition**: Asynchronous communication where a producer sends messages to a specific consumer.

- **Producer**: The application that generates and sends messages.
- **Consumer**: The application that receives and processes messages.
- **Message Queue**: A temporary storage for messages waiting to be processed, allowing for decoupled application architecture.

Key Concepts

1. **Message**: Represents data sent by the producer to the consumer (e.g., customer records, orders).
2. **Queue**: Holds messages until consumed; messages remain until deleted or until the retention period expires.
3. **Polling Mechanism**:
    - Consumers poll the queue to check for available messages (pull mechanism).

Amazon SQS Overview

- **Managed Service**: Fully managed message queuing service by AWS.
- **Integration**: Helps integrate and decouple distributed systems and application components.
- **High Availability**: Offers durable message queuing across multiple AWS Availability Zones.

Benefits of Amazon SQS

- **Fully Managed**: Eliminates the need for messaging software management.
- **Reliability**: Ensures message delivery without loss.
- **Security**: Supports secure data transfer with encryption.
- **Scalability**: Automatically scales based on usage; processes billions of messages daily.

Basic Components of Amazon SQS

1. **Messages**:
    - Max size: 256 KB (can use Extended Client Library for larger messages).
    - Retained for a configurable period (default: 4 days, max: 14 days).
2. **Queues**:
    - **Types**:
        - **Standard Queue**: At-least-once delivery, best-effort ordering.
        - **FIFO Queue**: First-in-first-out delivery, exactly once processing.
3. **Dead-Letter Queues (DLQ)**:
    - Stores messages that cannot be processed after maximum attempts.

Example Use Case

- **Order Processing**:

- Web application captures orders and uses an SQS queue to decouple the order fulfillment process.
- The producer (order capture) sends messages to the queue, and the consumer (order fulfillment) processes them asynchronously, enhancing resilience and scalability.

Queue Configuration

Polling Type:

- **Short Polling**: Returns messages from a subset of servers quickly, may result in empty responses.
- **Long Polling**: Waits for messages, reducing empty responses and associated costs.

Message Visibility Timeout:

- Prevents multiple consumers from processing the same message. Set to the maximum processing time required by the application (default: 30 seconds, max: 12 hours).

Message Lifecycle in SQS

1. Producer sends a message to the queue.
2. Consumer retrieves the message, initiating visibility timeout.
3. Consumer processes the message and deletes it to prevent reprocessing.

Use Cases for Amazon SQS

- **Work Queues**: Decouple components that process workloads at varying rates.
- **Buffering**: Mitigate traffic spikes or manage batch processing.

Decoupling applications with Amazon SNS:

Publish/Subscribe Messaging

- **Decoupling Applications**: Enables asynchronous communication where a **publisher** sends messages to multiple **subscribers** without knowing their details.
- **Topics**: Messages are sent to a topic, which broadcasts them to all subscribers (push mechanism) without queueing.
- **Asynchronous Notifications**: Allows multiple subscribers to perform different actions on the same message in parallel.

Amazon Simple Notification Service (SNS)

- **Overview**: A fully managed pub/sub messaging service that helps decouple applications through notifications.
- **Key Features**:

  - Supports multiple message destinations: email, SMS, HTTP/HTTPS, AWS Lambda, SQS, and Kinesis Data Firehose.
  - Durable storage of messages during processing, but does not persist messages after delivery.
  - Offers a web console and API for managing topics and subscriptions.

Use Cases

- **Application and System Alerts**: Instant notifications for events (e.g., changes in Auto Scaling groups).
- **Push Notifications**: Deliver targeted messages to users (e.g., mobile app updates).
- **Workflow Systems**: Integrate notifications into workflows for event-driven architectures.

Example Use Case: Decoupling with SNS and SQS

- **Fanout Scenario**: A mobile client uploads an image to S3, triggering an SNS notification that pushes messages to multiple SQS queues for processing by different applications (e.g., generating thumbnails).

Considerations

- **Message Delivery**: Use standard topics for non-sequenced delivery and FIFO topics for strict order.
- **Retry Policies**: Customize delivery policies for HTTP/HTTPS endpoints

Comparison: Amazon SNS vs. Amazon SQS

Feature	Amazon SNS	Amazon SQS
Messaging Model	Publisher-Subscriber	Producer-Consumer
Distribution Model	One-to-Many	One-to-One
Delivery Mechanism	Push (passive)	Pull (active)
Message Persistence	No	Yes

Decoupling a hybrid application with Amazon MQ:

Amazon MQ Overview

- **Service Type**: Fully managed message broker service for Apache ActiveMQ and RabbitMQ.
- **Functionality**: Facilitates communication between software applications and components that may use different programming languages and platforms.
- **Decoupling Applications**: Supports queue- and topic-based messaging for loose coupling of applications.

Features

- **Managed Service**: Reduces operational overhead by handling provisioning, setup, and maintenance.
- **Standard Protocols**: Supports various messaging APIs, including:
    - Java Message Service (JMS)
    - .NET Message Service (NMS)
    - Advanced Message Queuing Protocol (AMQP)
    - Streaming Text Oriented Messaging Protocol (STOMP)
    - Message Queuing Telemetry Transport (MQTT)

- WebSocket

- **Migration**: Allows migration from other message brokers without extensive code rewriting; simply update application endpoints.

Use Case: Hybrid Cloud Environment

- **Integration**: Enables communication between on-premises applications and cloud-based applications, facilitating hybrid environments.
- **Example Scenario**:

1. An on-premises producer application sends messages to a consumer application in AWS.
2. An Amazon MQ for ActiveMQ broker is created in an Availability Zone and configured for optimal performance.
3. The producer sends messages to the broker, which then propagates them to the consumer application.

Choosing the Right Solution for Decoupling

Service	Applicability	Messaging Model	Programming API	Pricing Model
Amazon SQS	Cloud-centered applications	Producer-Consumer	Amazon SQS API	Pay per request
Amazon SNS	Cloud-native applications	Publisher-Subscriber	Amazon SNS API	Pay per request
Amazon MQ	Hybrid applications and message broker migration	Producer-Consumer & Publisher-Subscriber	Industry standard APIs	Pay per hour & per GB

Recommendations

- **Amazon MQ**: Best for integrating on-premises applications with cloud applications and for migrating existing message brokers to the cloud.
- **Amazon SQS & SNS**: Recommended for new cloud applications due to ease of use and lack of broker management overhead.

1. Which statement describes the difference between tightly and loosely coupled architectures? (●) Components in a tightly coupled architecture are highly dependent on each other. In a loosely coupled architecture, components aren't highly dependent on each other. ( ) Loose coupling increases the complexity of scaling. Tight coupling decreases it. ( ) Tightly coupled architectures are more likely to fail than loosely coupled architectures. ( ) Loosely coupled architectures must use managed services, and tightly coupled architectures don't have this limitation.	The number of dependencies between components determines the level of coupling. A tightly coupled architecture has more dependencies between components than a loosely coupled architecture.

2. Which statements describe Amazon Simple Queue Service (Amazon SQS)? (Select THREE.) ☐ Sends push notifications to consumers ☑ Requires a consumer to poll the queue for messages ☐ Supports standard queues and topics ☑ Enables you to decouple and scale microservices, distributed systems, and serverless applications ☐ Supports email and text messaging ☑ Stores and optionally encrypts messages until they are processed and deleted	Amazon SQS is a fully managed message queuing service. You can send, store, and receive messages between software components without losing messages or requiring other services to be available. Message consumers retrieve (pull) messages from the queue.
3. Which statements are true when using an Amazon Simple Queue Service (Amazon SQS) standard queue? (Select TWO.) ☐ A message must be processed only once. ☐ Messages can be assigned a priority. ☑ Messages can be sent in any order. ☐ Messages must be processed in the order that they are sent. ☑ A message might be delivered more than once.	Amazon SQS offers two types of message queues. Standard queues offer maximum throughput, best-effort ordering, and at-least-once delivery. SQS first-in-first-out (FIFO) queues ensure that messages are processed exactly once, in the order that they are sent.
4. A fleet of Amazon EC2 instances process videos that users upload. Which function can Amazon Simple Queue Service (Amazon SQS) perform in this application? ○ An SQS queue receives messages from the application and notifies all available EC2 instances that videos are available. ○ EC2 instances put edited video files in an SQS queue. The application retrieves the videos from the queue. ◉ The application places job messages in an SQS queue. EC2 instances with available processing capacity pull the next job message from the queue. ○ The application writes the video files to an SQS queue. EC2 instances with available processing capacity pull the next video from the queue.	This function is like people in line for the next available customer service representative. By using an SQS queue, you can send, store, and receive messages between software components at any volume, without losing messages or requiring other services to be available.
5. What is Amazon Simple Notification Service (Amazon SNS)? ◉ A fully managed messaging service for both system-to-system and application-to-person (A2P) communication, which uses publish/subscribe patterns ○ A cost-effective, flexible, and scalable email service that enables developers to send email from within any application ○ A flexible and scalable outbound and inbound marketing communications service that uses email, short message service (SMS), push, or voice communication channels ○ A serverless event bus that enables easy connection of applications by using data from your own applications, integrated software as a service (SaaS) applications, and AWS services	When you use Amazon SNS, you create topic and set policies that restrict who can publish or subscribe to the topic.
6. What are some use cases for Amazon Simple Notification Service (Amazon SNS)? (Select THREE.) ☐ Trigger a single AWS Lambda function when an object is created in an Amazon S3 bucket. ☑ Send a text message to systems operators when unusual activity has been detected. ☑ Send a push notification to mobile applications when a new software update is available. ☑ Notify multiple systems that user input is ready for processing. ☐ Gather streaming data from multiple systems. ☐ Hold input until it can be processed in the order that it's received.	SNS topics can send messages to email addresses, HTTP(S) URLs, AWS Server Migration Service (AWS SMS) clients, SQS queues, and AWS Lambda functions.

7. What are some features of Amazon Simple Notification Service (Amazon SNS (Select TWO.) ☐ Guaranteed message delivery even when an endpoint isn't accessible ☐ Providing strict message ordering with standard topics. ☑ Message delivery to an Amazon Simple Queue Service (Amazon SQS) que ☐ Recall of sent messages ☑ Message delivery to a URL	SNS topics can send messages to email addresses, HTTP(S) URLs, AWS Server Migration Service (AWS SMS) clients, SQS queues, and AWS Lambda functions.
8. Two different AWS Lambda functions must simultaneously process PDF files that are uploaded to an Amazon S3 bucket. The S3 event notification allows only one action when the PDF files are uploaded. Which solution provides the least complex way to process messages with both Lambda functions efficiently? ○ Send the S3 event to an Amazon Simple Queue Service (Amazon SQS) queue that both Lambda functions poll. ◉ Send the S3 event to an Amazon Simple Notification Service (Amazon SNS) topic that both Lambda functions subscribe to. ○ Send the S3 event to Amazon MQ for distribution to both Lambda functions. ○ Upload two copies of each PDF file by using different object key prefixes.	Amazon SNS is fully integrated with Amazon S3 and AWS Lambda. This design is efficient because the Lambda functions don't run unless they receive notification from the SNS topic.
9. What is Amazon MQ? ◉ Message broker service ○ Data migration service ○ Application monitoring service ○ Identity broker service	Amazon MQ is a managed message broker service that makes it easy to set up and operate message brokers in the cloud.
10. Which is a common use case for Amazon MQ? ◉ Leverage an existing on-premises application that uses Apache ActiveMQ to communicate between microservices. ○ Decouple components in a new cloud-native application. ○ Connect a virtual private cloud (VPC) to an on-premises network. ○ Upload a standalone static website to AWS.	Amazon MQ reduces the operational load by managing the provisioning, setup, and maintenance of Apache ActiveMQ. Connect current applications to Amazon MQ using open standard APIs and protocols.

- Tightly coupled systems are difficult to scale and introduce bottlenecks and single points of failure.
- A loosely coupled architecture removes direct dependencies between related components and permits scalability and resiliency.
- Loose coupling solutions divide infrastructure layers or application functions and typically introduce an intermediary component between them.
- Loose coupling solutions can be synchronous or asynchronous.
  - ELB is an example of a synchronous solution.
  - Amazon SQS, Amazon SNS, and Amazon MQ are examples of an asynchronous solution.

- Amazon SQS is a fully managed message-queuing service that you can use to decoupl application components so that they run independently.
- Amazon SQS supports standard and FIFO queues.
- Messages that cannot be processed can be sent to a dead-letter queue.
- Long polling helps reduce the cost of using Amazon SQS by reducing the number of empty responses to a consumer's receive message request.
- A producer sends a message to a queue. A consumer processes and deletes the messag during the visibility timeout period.

- Amazon SNS is a web service that you can use to set up, operate, and send notifications from the cloud.
- Amazon SNS follows the pub/sub messaging paradigm.
- To use Amazon SNS, you create a topic, create subscribers for the topic, and publish messages to the topic.
- You can use topics to decouple message publishers from subscribers and fanout messages to multiple recipients at one time.
- AWS services can publish messages to an SNS topic to invoke event-driven computing and workflows.

- Amazon MQ is a managed service that you can use to set up and operate Apache ActiveMQ and RabbitMQ message brokers in the cloud.
- Amazon MQ is compatible with open standard messaging APIs and protocols.
- You can use Amazon MQ to integrate on-premises and cloud environments in a loosely coupled manner.
- You can also use Amazon MQ to migrate your on-premises open source message brokers to the AWS Cloud.

A company must perform asynchronous processing and implement Amazon Simple Queue Service (Amazon SQS) as part of a decoupled architecture. The company wants to ensure that the number of empty responses from polling requests is kept to a minimum.
What should a solutions architect do to ensure that empty responses are reduced?

Choice	Response
A	Increase the maximum message retention period for the queue.
B	Increase the maximum receives for the redrive policy for the queue.
C	Increase the default visibility timeout for the queue.
D	Increase the receive message wait time for the queue.

D Increase the receive message wait time for the queue

MODULE-14

Building Serverless Architectures and Microservices

Serverless Architectures in AWS

Three-Tier Design in a VPC

- **Web Tier**: Deployed on multiple Amazon EC2 instances across two Availability Zones (AZs) in an auto-scaling group for high availability.
- **Application Tier**: Managed through an Application Load Balancer; ensures business logic processing with redundancy.
- **Data Tier**: Utilizes Amazon RDS with Multi-AZ deployment for fault tolerance and automatic promotion of the secondary database in case of failure.

Benefits of AWS Serverless

- **No Server Management**: Focus on application logic, not server maintenance.
- **Pay-for-Value**: Cost is based on actual usage, ideal for startups.
- **Continuous Scaling**: Automatically adjusts resources based on demand.
- **High Availability**: Services designed to operate across multiple AZs.
- **Supports Event-Driven Architectures**: Facilitates microservices and event-based communication.

Key AWS Serverless Services

Compute:

- **AWS Lambda**: Event-driven execution without server management.
- **AWS Fargate**: Serverless containers for running applications.

Authentication:

- **Amazon Cognito**: User management and identity integration.

Web Hosting:

- **AWS Amplify**: Build and host web/mobile apps.
- **Amazon CloudFront**: Distributes static content.

Application Integration:

- **Amazon API Gateway**: Manages RESTful APIs.
- **AWS AppSync**: Manages GraphQL APIs.
- **Amazon SQS/SNS**: Decouples components and facilitates notifications.
- **AWS Step Functions/EventBridge**: Orchestrates workflows and event routing.

Data Stores:

- **Amazon S3**: Object storage.
- **Amazon DynamoDB**: Low-latency key-value/document database.
- **Aurora/Redshift Serverless**: Scalable relational and data warehouse solutions.

Example Serverless Three-Tier Architecture

1. **Front-End**: Served via Amazon CloudFront from an S3 bucket.
2. **Authentication**: Handled by Amazon Cognito with JWTs for API access.
3. **API Gateway**: Validates requests and invokes AWS Lambda functions.
4. **Data Access**: Lambda functions retrieve data from DynamoDB.

Conclusion

AWS serverless architectures simplify application development, offering scalable, resilient solutions without infrastructure complexities, making them ideal for modern applications.

Architecting serverless microservices

Microservices Overview

- **Definition**: Microservices architecture breaks applications into independent components, allowing each service to run autonomously and perform a single business function.

Characteristics of Microservices

1. **Autonomous**:
    - Independent development and deployment.
    - Scales independently.
    - Does not share code with other microservices.
    - Communicates via well-defined APIs.
2. **Specialized**:
    - Focuses on a single business capability.
    - Owned by small development teams that select tools.
    - Stateless design for fast scaling and recovery.
    - Owns its data store to maintain ACID transactions.

Monolithic vs. Microservices

- **Monolithic Applications**: Tightly coupled components; scaling or updating one part affects the entire system, increasing risk of failure.
- **Microservice Applications**: Each component (e.g., user service, topic service) is independent, allowing for specific scaling, updates, and feature enhancements without impacting other services.

Benefits of Microservices

1. **Team Agility**: Small teams manage services, enabling quicker development cycles.
2. **Reusable Code**: Functions can be reused across different features.
3. **Flexible Scaling**: Each microservice can be scaled according to its demand.
4. **Technological Freedom**: Teams choose the best tools for their services.
5. **Resilience**: Applications can degrade functionality without crashing entirely.
6. **Simplified Deployment**: Continuous integration and delivery pipelines facilitate experimentation and quick rollbacks.

Microservice Patterns on AWS

1. **RESTful APIs**: Utilize Amazon API Gateway and AWS Lambda for stateless service communication.
2. **Containers**: Use AWS Fargate for longer-running processes (over 15 minutes) with Amazon ECS or EKS as alternatives.
3. **Streaming**: Integrate AWS Lambda with Amazon Kinesis for scalable streaming services.

Microservices in a Web Serverless Three-Tier Architecture

- **App and Data Tier**: Microservices operate here, fulfilling API requests and utilizing:
    - **API Gateway**: Acts as the API.
    - **AWS Lambda**: Handles compute tasks (max 15-minute execution).
    - **DynamoDB**: Serves as the data store.

Conclusion

Microservices offer a flexible and scalable architecture for modern applications, enhancing development efficiency and resilience. AWS provides various tools and patterns to effectively implement serverless microservices.

Building serverless architectures with AWS Lambda :

Connecting a Lambda Function to Your VPC

- **Default Behavior**: Lambda functions are not connected to VPCs by default.
- **VPC Access**: To access resources in a VPC, configure the Lambda function to connect to it, utilizing Hyperplane elastic network interfaces (ENIs) created for this purpose.
- **NAT Gateway**: To enable internet access for the function, route outbound traffic to a NAT gateway in a public subnet.
- **Scaling Considerations**: Be aware that resources like EC2 instances and RDS databases can become bottlenecks if Lambda functions scale too aggressively. Solutions include:
    - Deploying EC2 instances behind an application load balancer.
    - Using RDS proxy for managing connections to databases.

Identifying Lambda Serverless Scenarios

- **Synchronous Processing**: Commonly used in web apps, microservices, and ML inferences. Example: A Lambda function invoked via API Gateway to update a shopping cart.

- **Asynchronous Processing**: Used for events scheduled to run later, like image transformations and status updates from queues.
- **Streaming Processing**: Suitable for handling continuous data streams, such as processing changes in DynamoDB.

Invoking Lambda Functions

- **Synchronous Invocation**:
    - Via API Gateway or function URL, where the caller waits for a response.
    - Responses include results or errors.
- **Asynchronous Invocation**:
    - The caller hands off the event and doesn't wait for a response. Lambda manages the event queue and can handle errors.
    - AWS services like S3 and SNS can trigger Lambda functions asynchronously.

Event Source Mappings

- Event source mappings allow Lambda to read from queues or streams (e.g., DynamoDB, Kinesis) that don't invoke Lambda directly.
- Lambda batches records and invokes the function with the payload, adhering to size limits.

Python Lambda Function Handler

- The handler function processes incoming events and utilizes the event and context objects.
- Best practices include keeping the handler code minimal and separating business logic into different methods.

Lambda Layers

- Layers are .zip archives containing supplementary code, dependencies, or custom runtimes.
- Benefits of using layers:
    - Reduce deployment package size.
    - Separate core logic from dependencies.
    - Share dependencies across multiple functions.
    - Enable usage of the Lambda console code editor for smaller packages.

Building microservice applications with AWS container services:

1. Creating and Deploying Docker Containers

- **Dockerfile**: A text file containing build instructions to create a container image.
- **Build Process**: After creating the Dockerfile, the image is built using the container engine's build command.
- **Container Repository**: The image is uploaded to a repository after local testing.
- **Deployment**: Containers are typically deployed from a single image, often using orchestration services to manage scaling and deployment.

- **Orchestration Services**: Monitor the health of the compute platform and containers, facilitating deployment across different environments per the Open Container Initiative (OCI) standards.

2. AWS Container Services

- **Registry**: Amazon Elastic Container Registry (ECR) is a managed service for storing container images with support for private/public repositories.
- **Orchestration**:
    - **Amazon Elastic Container Service (ECS)**: An AWS-native service for container orchestration with built-in best practices.
    - **Amazon Elastic Kubernetes Service (EKS)**: Uses Kubernetes to manage containers, suitable for those already familiar with Kubernetes.

3. Compute Options

- **Amazon EC2**: Virtual servers for running applications.
- **AWS Fargate**: Serverless compute engine that allows running containers without managing servers.
- **AWS Lambda**: Allows invoking containers, but not managed by orchestration services.

4. Benefits of AWS Fargate

- **No Server Management**: Eliminates the need for provisioning or maintaining servers.
- **Billing**: Per second billing, ensuring users pay only for what they use.
- **Automatic Scaling**: Tasks can be scaled based on demand, suitable for teams new to container technology.

5. Deploying and Invoking Containers

- **Amazon ECS**:
    - Create an ECS cluster with compute nodes (AWS Fargate or EC2).
    - Define an ECS task as a blueprint for your application.
    - Maintain high availability using services that manage the desired number of tasks.

- **Amazon EKS**:
    - Create a cluster with worker nodes.
    - Deploy containers as Pods, with the option for Deployments and Services for better management and scaling.

6. Choosing Between Amazon ECS and Amazon EKS

Topic	Amazon ECS	Amazon EKS
Complexity	Simplifies cluster management	Offers more control but is complex
Scaling	Automated scaling based on demand	Manual configuration needed for scaling
Toolset	ECS toolset	Kubernetes toolset
Team Experience	Ideal for teams new to container architecture	Best for teams familiar with Kubernetes

Summary

- **ECS**: Best for simplicity and integration with AWS services.
- **EKS**: Preferred for flexibility and control over Kubernetes-based environments.

Orchestrating microservices with AWS Step Functions :

Key Concepts of AWS Step Functions

Use Cases:

1. **Microservice Orchestration:** Manage the workflows of multiple microservices.
2. **Data Processing:** Handle large-scale data processing tasks efficiently.
3. **Machine Learning Workflows:** Automate various stages of ML processes, from data preprocessing to model evaluation.
4. **Security Automation:** Streamline repetitive security tasks, such as software updates and vulnerability management.

Workflow Types:

1. **Standard Workflows:** Suitable for long-running processes, can orchestrate applications using AWS Fargate.
2. **Synchronous Express Workflows:** Best for short-duration workflows requiring immediate responses, often used in web and mobile applications.
3. **Asynchronous Express Workflows:** For short-duration workflows that don't require immediate feedback.

Scalability and Fault Tolerance:

1. Step Functions can handle millions of concurrent executions and provides automatic retries and error management, enhancing workflow reliability.

Workflow Coordination:

1. Supports sequential and parallel task execution, error management through try-catch-finally patterns, and conditional branching with choice states.

State Machine State Types:

1. **Work States:**
    1. **Task:** Executes a task integrated with AWS services.
    2. **Activity:** Performs tasks hosted externally.
    3. **Pass:** Transfers input data to the next state without processing.
    4. **Wait:** Delays workflow for a specified duration.
2. **Transition States:**
    1. **Choice:** Controls the workflow based on conditions.
    2. **Parallel:** Executes multiple tasks simultaneously.
    3. **Map:** Processes datasets in parallel.
3. **Stop States:**
    1. **Success:** Ends the workflow successfully.
    2. **Fail:** Marks the workflow as failed.

Defining a State Machine:

1. State machines are defined using Amazon States Language (ASL), a JSON-based syntax that specifies states, transitions, and execution order.

Example Workflow: Stock Trade State Machine:

1. A practical example is provided where a stock trading process involves:
    1. Initiating a Lambda function to fetch trade details.
    2. Conditional checks for trade recommendations.
    3. Human approval through notifications.
    4. Handling transaction outcomes based on approval or rejection.

Conclusion

AWS Step Functions offer a powerful framework for managing workflows, making them ideal for microservices, data processing, and automation. Their flexibility, scalability, and built-in error handling significantly simplify complex workflow management

Extending serverless architectures with Amazon API Gateway :

API Types

REST APIs

1. **Characteristics**: Stateless, collection of routes and methods (e.g., GET, POST, PUT).
2. **Use Case**: Cloud applications that require API management features like usage plans, payload validation, and resource policies.
3. **Features**:

1. Allows private API endpoints for secure access.
2. API proxy functionality for extensive management features.
3. Supports CORS.

HTTP APIs

1. **Characteristics**: Stateless, optimized for serverless workloads.
2. **Use Case**: Microservices needing low latency and cost-effective solutions.
3. **Features**:
    1. Provides better performance and lower costs than REST APIs.
    2. Does not offer extensive API management features.
    3. Supports CORS.

WebSocket APIs

1. **Characteristics**: Collection of WebSocket routes, stateful.
2. **Use Case**: Real-time applications requiring persistent connections.
3. **Features**:
    1. Allows bidirectional communication between client and server.
    2. API management features include payload schema validation and data transformations.

Backend Integrations

- **HTTP Proxy Integration**: Connects API routes to publicly routable HTTP endpoints, passing requests and responses directly.
- **First-Class Integrations**: Connects API routes directly to AWS service APIs (e.g., Amazon SQS, AWS Step Functions).
- **VPC Link**: Connects API Gateway to services within a VPC, such as Load Balancers or EC2 instances.

Decomposing a Monolithic Application

For an online shopping application with components like Shopping Cart, Payments, and Delivery, the following solutions can be implemented:

Shopping Cart Microservice:

- **API Type**: HTTP API.
- **Integration**: AWS Lambda for routing requests.
- **Data Storage**: Amazon DynamoDB for storing shopping cart transactions.
- **Performance Requirement**: Millisecond response times.

Payment Microservice:

- **API Type**: HTTP API.
- **Integration**: Amazon ECS containers for processing payments, using an Application Load Balancer for request distribution.
- **Data Storage**: DynamoDB for payment records.

- **Porting Method**: Minimal code/configuration changes from on-premises.

Delivery Workflow Service:

- **API Type**: HTTP API.
- **Integration**: AWS Step Functions for managing the delivery process as a workflow.
- **Queue Management**: Amazon SQS for handling delivery requests.
- **Notifications**: Amazon SNS for email notifications upon delivery or cancellation.
- **Workflow States**: Manage order status updates and refunds.

Applying AWS Well-Architected Framework principles to microservices and serverless architectures :

The **AWS Well-Architected Framework** provides a structured approach to designing and operating reliable, secure, efficient, and cost-effective systems in the cloud. The **Serverless Applications Lens** focuses specifically on best practices for serverless architectures, emphasizing critical aspects of design, implementation, and maintenance. Here's a summary of the key best practices highlighted in your content:

1. Failure Management

- **Use a Dead-Letter Queue:** Implement mechanisms to retain, investigate, and retry failed transactions using Amazon SQS dead-letter queues.
- **Roll Back Failed Transactions:** Use AWS Step Functions to manage transaction rollbacks for synchronous calls.

2. Identity and Access Management

- **Control Access to Your Serverless API:** Utilize Amazon Cognito, Lambda authorizers, and resource policies to authorize API calls effectively.
- **Manage Security Boundaries:** Implement the principle of least privilege for Lambda functions, ensuring they only have the permissions necessary for their tasks.

3. Data Protection

- **Encrypt Data:** Ensure data is encrypted in transit and at rest. This includes encrypting sensitive information in API requests and storing it securely in services like Amazon DynamoDB or S3.
- **Implement Application Security:** Validate and sanitize incoming events and perform thorough security code reviews.

4. Performance Optimization

- **Optimize Performance:** Conduct performance tests under different conditions (steady and burst rates) to ensure your serverless application meets performance expectations.
- **Test AWS Service Configurations:** Experiment with memory settings in AWS Lambda and evaluate the execution rates for Step Functions to find the optimal configuration.

5. Cost-Effectiveness

- **Optimize Application Costs:** Take advantage of the pay-per-use model of serverless architectures, which can reduce the need for extensive capacity planning and management.

6. Optimizing Over Time

- **Use Direct AWS Service Integrations:** Leverage direct integrations between AWS services (e.g., API Gateway, EventBridge, Lambda) to reduce operational overhead and enhance cost efficiency.

Summary

These best practices aim to enhance the reliability, security, and efficiency of serverless applications while optimizing performance and cost. Following the guidelines in the AWS Well-Architected Framework, particularly the Serverless Applications Lens, can lead to a more robust architecture and a better user experience. For further details, you can explore the complete best practices and recommendations on the AWS Well-Architected Framework website.

1. A solutions architect wants to propose a microservice architecture on AWS to management. Which serverless computing benefits should they include in the proposal (Select THREE.) ☑ Continuous scaling ☐ Lessens amount of server maintenance ☐ Predictable performance ☐ Full control over runtime environment ☑ Pay-for-value services ☑ Built-in high availability	With serverless computing, you pay only for execution duration instead of by server unit, and there is no infrastructure to provision or manage. These lower your total cost of ownership. Serverless computing provides a good solution for microservice applications.
2. A developer needs to create and deploy a simple web form to capture employee commute information for the company internal employee website. Which is a serverle solution for creating the simple web form? ◉ Host static assets in an Amazon S3 bucket, use an Amazon DynamoDB table, and use Amazon API Gateway and AWS Lambda functions to interact with the database ○ Host website assets in a container in an Amazon Elastic Container Service (Amazon ECS) cluster with EC2 instance nodes, use an Amazon DynamoDB table, and use server-side scripts to interact with the database. ○ Host static assets in an Amazon S3 bucket, use an Amazon RDS database, and use Amazon API Gateway and AWS Lambda functions to interact with the database. ○ Host website assets in a container in an Amazon Elastic Container Service (Amazon ECS) cluster with Fargate nodes, use an Amazon RDS database, and use server-side scripts to interact with the database.	All of these services are serverless. You can improve performance by using an Amazon CloudFront distribution to cache static assets for a global audience.
3. Why would a solutions architect recommend to use a microservice architecture? (Select TWO.) ☐ Industry-standard interfaces ☐ HTTP(S) communication ☐ Open-source code ☑ Solves a specific business problem ☑ Independent from other components	In a microservice architecture, business functions of a larger overall application are assigned to specific components. Components are loosely coupled and maintain independence.

4. Which scaling configuration does a team lead need to apply to accomplish Lambda function scaling?

- (●) None because the AWS Lambda service scales functions automatically.
- ( ) Provision enough function instances to meet the maximum predicted load.
- ( ) Configure the auto scaling parameter in the function configuration.
- ( ) Launch functions in Auto Scaling groups.

When multiple Lambda functions are invoked, the AWS Lambda service provisions multiple environments to invoke multiple functions simultaneously.

5. A solutions architect has to use a custom library in an architecture that uses Lambda functions as the compute option. Which Lambda feature should be used for optimal function deployments?

- ( ) Lambda destinations
- ( ) Lambda function URLs
- (●) Lambda layers
- ( ) Lambda triggers

With layers, you can use libraries in your function without needing to include them in your deployment package.

6. Which set of statements is true for software packaged as a container?

- (●) Standardized, portable application code packages that contain code and code dependencies. A container is run by a container engine. A container does not include a guest operating system.
- ( ) Standardized, portable application code packages that contain code and code dependencies. A container is run by a container engine. A container includes a guest operating system.
- ( ) Portable application code packages that contain code and code dependencies. A container is run by a hypervisor. A container does not include a guest operating system.
- ( ) Portable application code packages that contain code and code dependencies. A container is run by a hypervisor. A container includes a guest operating system.

A container enables you to run an application and its dependencies in resource-isolated processes on top of a container engine without a guest operation system. It can be ported to different host operating systems. Virtual machines use hypervisors to host VM packages containing a guest operating system with the application and its dependencies.

7. Which is the most effective container deployment using Amazon Elastic Container Service (Amazon ECS) containers when refactoring a monolithic application to use a microservice architecture?

- ( ) Create services that each provide a distinct function of the application, and run each service in a separate container that Amazon ECS manages.

The first step in moving to a microservice architecture is to break up monolithic applications into services that each provide a more specialized function.

8. An environmental science organization wants to provide HTTPS read-only access to its sensor data Amazon DynamoDB database. The user usage pattern is intermittent with 68 percent idle time per month. Which solution is the most cost efficient and secure with the least amount of operational maintenance?

- ( ) Create web proxy servers on Amazon EC2 instances in an Auto Scaling group connecting to the DynamoDB sensor database, which is served by an Elastic Load Balancing load balancer.
- (●) Create a public interface by using Amazon API Gateway with Lambda functions accessing the DynamoDB sensor database.
- ( ) Create a microservices architecture by using Amazon Elastic Container Service (Amazon ECS) connecting to the DynamoDB sensor database.
- ( ) Create user accounts in the organization's systems to allow direct access to the DynamoDB sensor database.

You can create a serverless API by using Amazon API Gateway and AWS Lambda to provide read-only connections to the sensors and databases.

9. Which workflows are suitable to implement with AWS Step Functions when the goal is to implement the solution to reduce operational overhead? (Select THREE.)

- [ ] Notify a group of email addresses when the costs for an AWS account exceed a threshold.
- [x] Implement manual approval in a security incident response workflow.
- [x] Coordinate multi-step analytics and machine learning workflows.
- [ ] Deploy different kinds of infrastructure that are based on environment variables.
- [ ] Send messages to multiple Amazon EC2 instances when a file is created or modified in an Amazon S3 bucket.
- [x] Update inventory and run a shipment workflow when a customer purchases an item on an e-commerce site.

These workflows involve multi-step operations and decision making, both of which can be automated. These characteristics make them good candidates for AWS Step Functions.

10. A developer wants to implement a manual approval step using an AWS Step Functions state machine. The manual approval state has to pause until a notification of approval or rejection is received from an approver email. Which should be implemented in the Step Functions state machine?

- ○ A task or activity state with automated callback parameters
- ○ A map state with an activated wait for callback parameter
- ○ A map state with automated callback parameters
- ◉ A task or activity state with an activated wait for callback parameter

A wait for callback parameter is an optional feature of a task or activity. It pauses the execution at the state until the execution receives a callback from SendTaskSuccess or SendTaskFailure APIs with the task token.

- AWS serverless services provide you the following benefits:
  - No server management
  - Pay-for-value services
  - Continuous scaling
  - Built-in fault tolerance
- Serverless services are suitable for event-driven and microservice architectures.

- Microservice applications are composed of autonomous, specialized services.
- Microservices have the following benefits:
  - Team agility
  - Reusable code
  - Flexible scaling
  - Technological freedom
  - Resilience
  - Simplified deployment
- Microservices can form part of a three-tier architecture operating in the app and data tiers.

- AWS Lambda is a service to run code functions without provisioning or managing servers.
- A Lambda function can run inside a VPC owned by the AWS Lambda service or as Lambda@Edge in an Amazon CloudFront regional cache.
- A Lambda function can be configured to connect to your VPC to access AWS services inside the VPC.
- A Lambda function can be invoked synchronously, asynchronously, and with event source mappings for queues and streams.
- Use Lambda layers to package code dependencies or custom runtimes to be re-used by all Lambda functions in the Region.

- Choose a container solution instead of Lambda functions when an application requires resources exceeding Lambda service limits.
- Amazon ECR provides a container image repository service.
- Amazon ECS is a container orchestration service with AWS toolkit.
- Amazon EKS is a container orchestration service with Kubernetes toolkit.
- AWS Fargate manages serverless cluster nodes and can be deployed in Amazon ECS and Amazon EKS clusters.

- AWS Step Functions is a serverless orchestration service to manage workflows between multiple AWS services.
- A state machine (workflow) is a series of event-driven states (steps).
- A Step Functions state machine is a collection of states defined in the Amazon States Language.
- States can be grouped into work, transition, and stop states.
- A task state can invoke an AWS service or request an activity hosted on any compute service with an HTTP connection.

- Amazon API Gateway provides the ability to create, publish, and maintain application APIs.
- It provides access to AWS services and publicly accessible endpoints.
- Use REST APIs when full API management and control are required.
- Use HTTP APIs when lower latency and lower cost than REST APIs are required.
- Use WebSocket APIs for real-time applications that require an active session.

• Use a dead-letter queue mechanism to retain, investigate, and retry failed transactions. • Roll back failed transactions. • Control access to your serverless API. • Control access to your serverless application. • Encrypt data in transit and at rest. • Implement application security in your workload. • Optimize the performance of your serverless application. • Optimize application cost. • Use direct AWS service integrations where available.	

What is the most effective use of Amazon Elastic Container Service (Amazon ECS) when refactoring a monolithic application to use a microservice architecture?

Choice	Response
A	Create services that each provide a distinct function of the application, and run multiple services in a single container that Amazon ECS manages.
B	Port the application to a new image, and run it in a container that Amazon ECS manages.
C	Refactor the application and centralize common functions to create a smaller code footprint.
D	Create services that each provide a distinct function of the application, and run each service in a separate container that Amazon ECS manages.

D Create services that each provide a distinct function of the application, and run each service in a separate container that Amazon ECS manages.

MODULE-15:

Data characteristics:

AWS Well-Architected Framework: Serverless Applications Lens

Key Concepts

Five Vs of Data Characteristics:

1. **Value**: Ensuring the data collected provides business value and insights.
2. **Veracity**: Accuracy and trustworthiness of data to support decision-making.
3. **Volume**: Amount of data affects storage, processing, and access methods.
4. **Velocity**: Speed of data generation impacts processing and pipeline design.
5. **Variety**: Diversity of data types and sources influences how data is processed.

Data Infrastructure Design Considerations:

1. Keep the end-user in mind.
2. Value relies on veracity.
3. Consider data retention duration and access frequency.
4. Volume and velocity together dictate throughput and scaling.
5. Combining datasets can enrich analysis but complicates processing.

Modern Data Architecture:

1. **Modernize**: Shift to cloud-based infrastructures to reduce operational overhead.
2. **Unify**: Create a single source of truth to make data available organization-wide.
3. **Innovate**: Leverage AI/ML for new insights and improved processes.

Solution for Complex Data Movement:

1. A modern architecture integrates data lakes, data warehouses, and other data stores to enable unified governance and seamless data movement, allowing for agile decision-making and analytics.

Data pipelines :

Elements of a Data Pipeline

Data Sources:

1. Origin of raw data, which can come from various platforms and formats.

Ingest:

1. **Homogeneous Ingestion**: Moving data from a source to a destination without transformation (e.g., extracting data from a NoSQL database to Amazon RDS MySQL).

2. **Heterogeneous Ingestion**:

    1. **ETL (Extract, Transform, Load)**:
        1. **Steps**:
            1. Extract structured data.
            2. Transform it into the required format.
            3. Load into structured storage for analytics.
    2. **ELT (Extract, Load, Transform)**:

        1. **Steps**:
            1. Extract structured or unstructured data.
            2. Load it into storage in raw form.
            3. Transform as needed for analytics.

Process:

1. Transform data to ensure it's suitable for analysis, often involving schema alignment and data cleansing.

Analyze:

1. Discover insights and build visualizations or predictive models based on the processed data.

Store:

1. Securely and cost-effectively store both raw and processed data, ensuring easy access for analysis.

Data Processing Patterns

Batch Processing:

- Processes complete datasets at once.
- Runs infrequently, often scheduled during off-peak hours.
- Requires high computing power.
- Use Case: Analyzing sales transaction data overnight for morning reports.

Streaming Processing:

- Continuously processes data in real-time as it arrives.
- Requires low computing power and reliable, low-latency network connections.
- Use Case: Providing immediate product recommendations based on current user behavior.

- Comparison of Batch and Streaming Processing Patterns

Feature	Batch Processing	Streaming Processing
Data Processing Cycles	Infrequent, runs during off-peak hours	Continuous, real-time analytics
Compute Requirements	Requires high computing power	Requires low computing power
Use Case	Sales transaction analysis overnight	Immediate product recommendations

AWS tools to ingest data:

Ingestion Layer of the Data Pipeline

The ingestion layer is responsible for bringing raw data into the cloud and preparing it for processing. Key components include:

Data Sources: Various origins of raw data, such as SaaS applications, on-premises databases, and third-party data providers.

Ingestion Services:

Amazon AppFlow:

1. Integrates data between SaaS applications (e.g., Salesforce, Zendesk) and AWS services.
2. Automates data flows, reducing the time and effort needed to create connectors.
3. Offers transformation capabilities like filtering, masking, and aggregating.

AWS DataSync:

1. Transfers large volumes of data between on-premises data centers and AWS (Amazon S3, EFS, FSx).
2. Supports scheduled replication with encryption and integrity validation for secure data transfer.

AWS Data Exchange:

1. Facilitates finding, subscribing to, and using third-party data in the cloud.
2. Bridges the gap between data providers and subscribers.
3. Supports delivery through files, tables, and APIs, allowing quick access to licensed data.

Use Cases:

- **Amazon AppFlow**: Quickly integrate and transform data from multiple SaaS applications.

- **AWS DataSync**: Efficiently sync large datasets between on-premises and AWS.
- **AWS Data Exchange**: Enhance analytics with third-party data, like weather data for inventory planning or location data for restaurant expansion.

Processing batch data :

Batch Ingestion and Processing Overview

Batch ingestion and processing involve collecting and analyzing large volumes of data at scheduled intervals rather than in real-time. Here are the key points:

Use Case Example:

1. **Banking System**: Collects all financial transactions at the end of each day to generate reports for stakeholders. This approach improves efficiency and utilizes processing power more cost-effectively.

Benefits:

1. Efficient for repetitive tasks and compute-intensive operations.
2. Allows processing during off-peak times, optimizing resource usage.

Common Applications:

1. **Medical Research**: Analyzing large datasets in fields like computational chemistry and genomic testing, where real-time analysis is unnecessary.

When to Use Batch Processing:

1. For **reporting** purposes.
2. When handling **large datasets**.
3. For analytics focused on **aggregating** or **transforming** data instead of real-time analysis.

AWS Glue: Batch Processing and ETL

Overview: AWS Glue is a data integration service that automates ETL (Extract, Transform, Load) tasks for batch or streaming data pipelines.

1. **Data Sources**: Integrates with multiple AWS services (e.g., Amazon S3, DynamoDB, Redshift).

Key Components:

1. **ETL Jobs**: Automates data transformation tasks.
2. **Data Catalog**: Stores metadata about datasets, including schema information.
3. **DataBrew**: A no-code tool for data cleaning and preparation.
4. **Data Quality**: Monitors and assesses data quality, detecting issues before they impact business decisions.

Workflow Example:

1. **Gaming Company**: Collects user play data every 6 hours, processed by AWS Glue for analysis. The workflow includes:

    1. Data ingestion into S3.
    2. Crawlers populating the Data Catalog.
    3. Aggregation jobs running to prepare data for analytics.

Data Transformation:

1. AWS Glue supports converting data formats (e.g., from CSV to Apache Parquet) for better storage efficiency and faster analytics.
2. Can detect and mask sensitive information (PII) in datasets to ensure data privacy.

Summary

Batch ingestion and processing are efficient for handling large datasets in scheduled intervals, making it ideal for applications like banking and medical research. AWS Glue simplifies this process through automated ETL tasks, data cataloging, and data quality monitoring, ensuring effective data management and analysis.

Processing real-time data:

Streaming Pipeline Workflow with Kinesis Data Streams

- **Purpose**: Ingests and temporarily stores streaming data for real-time analytics.
- **Use Case**: Collecting clickstream data from social media apps.
- **Workflow**:
    - **Sources**: Mobile applications.
    - **Producers**: Apps delivering data to Kinesis.
    - **Streams**: Kinesis Data Streams for storage.
    - **Consumers**: Apps processing the data.
    - **Destination**: Data moved to permanent storage.

Amazon Managed Service for Apache Flink

- **Functionality**: Real-time data analysis and insights.
- **Use Cases**: Log analytics, IoT, ad tech, gaming.
- **Features**:
    - Aggregation and anomaly detection.
    - Serverless and auto-scaling.

Amazon Kinesis Video Streams

- **Functionality**: Captures live video data from various sources.
- **Features**:

- Streams video for analytics and ML.
- Secure storage and access via API.

Amazon MSK (Managed Streaming for Apache Kafka)

- **Overview**: Fully managed Apache Kafka service.
- **Benefits**:
  - Reduces operational overhead; compatible with existing tools.
  - More setup required compared to Kinesis.

Use Case Examples

1. **Café Clickstream Data**:
   - Analyzes user behavior using Kinesis Data Firehose to send data to Amazon S3.
2. **Medical Devices Monitoring**:
   - Uses Kinesis Data Streams for immediate anomaly detection from IoT sensors.

Comparison of Streaming Ingestion Services

- **Kinesis Data Firehose**: Easiest to use; retains undelivered data for 24 hours.
- **Kinesis Data Streams**: More control and retention (up to 365 days); requires coding.
- **Amazon MSK**: Ideal for open-source solutions; greater complexity.

Key Considerations

- Assess business use case, engineering effort, and data retention needs when selecting a service.

Storage in the data pipeline :

AWS Data Lake Architecture

- **Components**: Amazon S3, Lake Formation, AWS Glue, Athena.
- **Purpose**: Centralizes various data types for analytics and machine learning.

AWS Storage Services for Banking Application

Individual Customer Financial Transactions:

- **OLTP**: Use Amazon Aurora or DynamoDB for real-time processing and small record writes.

Total Daily Financial Transactions:

- **OLAP**: Use Amazon Redshift for long-term analysis and reporting on aggregated data.

User Activity Logs for Fraud Analysis:

- Store in **Amazon S3** (data lake) or **OpenSearch Service** for indexed search capabilities.

Application Error Logs:

- Use **Amazon S3** for simple, scalable storage.

Data Warehouse vs. Data Lake

Data Warehouse:

- Structured data, schema-on-write, curated for fast SQL queries.
- Higher cost, suitable for batch reporting and analytics.

Data Lake:

- Stores both structured and unstructured data, schema-on-read.
- Lower cost, supports diverse analytics (real-time, ML).

Cost and Querying Considerations

- **Amazon S3**: Cost-effective for data lakes, but slower query times.
- **Amazon Redshift**: Higher cost, efficient for large datasets.
- **Redshift Spectrum**: Allows querying S3 data without moving it to Redshift, reducing costs.

Storage Classes in Amazon S3

- **Standard**: For frequently accessed data.
- **Intelligent-Tiering**: For unknown access patterns.
- **Glacier**: For long-term archival storage.

Summary

Select storage options based on data type, structure, access patterns, and cost considerations. Use a combination of services to optimize data storage and analytics in a banking application.

Parallel processing in the data pipeline :

Big Data Workflow for Parallel Processing

1. **Dataset Splitting**: Large datasets are divided into smaller parts.
2. **Parallel Processing**: Parts are processed simultaneously.
3. **Aggregation**: Results are combined.

Amazon EMR Overview

- **Infrastructure Management**: Handles cluster provisioning and setup.

- **Deployment Options**:
    - Amazon EC2, VPC, Multi-AZ with Amazon EKS, or on-premises with AWS Outposts.
- **Serverless Option**: Use EMR Serverless for managing jobs without clusters.

ETL Solution Choices

- **Amazon EMR**: For full control and legacy Hadoop app migration.
- **EMR Serverless & AWS Glue**: For new cloud apps or pay-per-job models.
    - **EMR Serverless**: For teams with Hadoop/Spark experience.
    - **AWS Glue**: For teams new to analytics or focused on Spark.

Data Curation Workflow

1. **Data Ingestion**: Move data to Amazon S3 (drop zone) using Lake Formation.
2. **Data Cleaning**: Amazon EMR processes data and stores it in the analytics zone.
3. **Data Curation**: Another EMR job curates data for user access.

Best Practices

- Organize data lakes into zones based on quality for better processing and access.

Analysis and visualization :

AWS Services for Data Analysis and Visualization

Business Analyst: Interactive Data Dashboard

1. **Service**: **Amazon QuickSight**
    1. **Features**:
        1. Creates interactive dashboards with charts and graphs.
        2. Uses SPICE for fast visualizations.
        3. Allows sharing via published links or embedding in applications.
        4. Integrates with multiple data sources (RDS, S3, Redshift, etc.).
        5. Supports natural language queries with QuickSight Q.

Data Engineer: One-time SQL Queries

1. **Service**: **Amazon Athena**
    1. **Features**:
        1. Serverless SQL query engine for structured and unstructured data.
        2. Queries data stored in Amazon S3 using standard SQL.
        3. Integrates with Data Catalog for metadata management.
        4. Supports various data formats (CSV, JSON, Parquet, etc.).
        5. Pay-per-query model, ideal for one-off analysis.

DevOps Engineer: Real-time Application Monitoring Dashboard

1. **Service**: **OpenSearch Service**
    1. **Features**:
        1. Managed service for interactive log analytics and real-time monitoring.
        2. Provides OpenSearch Dashboards for data visualization.
        3. Supports alerting and anomaly detection using machine learning.
        4. High availability with Multi-AZ deployments for resilience.

Implementing the Principle of Least Privilege

- Ensure users or systems have only the necessary permissions to perform their tasks:
    - For **Amazon QuickSight**, grant read permissions to the specific datasets.
    - For **Amazon Athena**, ensure data access permissions for querying S3 data.
    - For **OpenSearch Service**, limit access based on monitoring and alerting needs.

By focusing on these services and implementing appropriate access controls, organizations can effectively analyze and visualize their data while maintaining security.

Applying the AWS Well-Architected Framework principles to data pipelines:

Review Process for Data Analytics Lens

Overview

The AWS Well-Architected Framework guides workload design on AWS, focusing on four key pillars: Security, Performance Efficiency, Cost Optimization, and Reliability.

1. Security Pillar

- **Best Practice**: Implement the principle of least privilege.
    - **Action**: Grant only necessary permissions to users based on their needs.

2. Performance Efficiency Pillar

- **Best Practice**: Choose appropriate tools for technical challenges.
    - **Action**: Utilize services like Amazon Redshift, Kinesis, and QuickSight as needed.

3. Cost Optimization Pillar

- **Best Practices**:
    - **Remove Unused Resources**: Delete outdated data and automate expirations in Amazon S3.

- **Reduce Overprovisioning**: Adjust infrastructure based on workload utilization.

4. Reliability Pillar

- **Best Practice**: Design resilience into analytics workloads.
  - **Action**: Use appropriate data ingestion patterns (ETL, ELT, ETLT) to meet business requirements.

Conclusion

Adhering to these best practices ensures secure, efficient, cost-effective, and reliable analytics workloads on AWS.

Appendix: Data Characteristics

Value of Data

1. Ensures data collected delivers business insights.
2. Design infrastructure with user needs in mind.
3. **Example**: Clickstream data informs customer behavior and requires real-time analytics.

Veracity of Data

1. Focuses on data accuracy and trustworthiness.
2. Assess data consistency and impact on decisions.
3. **Example**: Healthcare data inconsistencies affect patient care analysis.

Volume of Data

1. Refers to the amount of data processed and stored.
2. Consider dataset size and access frequency.
3. **Example**: Social media platforms must upgrade infrastructure due to increased data volume.

Velocity of Data

1. Speed at which data moves through the pipeline.
2. Evaluate generation frequency and processing speed.
3. **Example**: Smart city projects require infrastructure for real-time traffic data processing.

Variety of Data

1. Different types and sources of data collected.
2. Identify data formats and diversity of sources.
3. **Example**: Retail companies need to integrate unstructured data (e.g., reviews) with structured data.

Data Pipelines

- **Transforming Data During Ingestion**: Ensure data quality by cleaning, formatting, augmenting, and adding metadata for analysis.

1. Which scenario describes a challenge to data velocity? ○ A sales department wants to use a data source but does not have information about its lineage or the quality of the data. ○ Regional offices send data in different file formats to an organization's head office. ◉ A shopping website collects clickstream to make personalized recommendations while a user is shopping. When the website is very busy, there is a delay in returning results to customers. ○ A pipeline ingests data from regional sales sites, and the overnight batch job fails because it runs out of disk space.	This represents a data velocity challenge. The application cannot process the incoming data fas enough to keep up with the pace of requests.
2. Which statement describes the goal of a modern data architecture? ○ Give users the ability to access all of an organization's data through a highly structured data warehouse that provides for fast SQL queries. ◉ Give users the ability to access all of an organization's data by integrating a data lake, a data warehouse, and other purpose-built data stores. ○ Select a single ingestion service that can support the data formats, structures, and velocity requirements of all the data sources that will be collected. ○ Select purpose-built streaming services to make all of the organization's data available for real-time analysis.	Integrating a data lake, a data warehouse, and other purpose-built data stores supports unified governance and data movement among different data stores.
3. Which analytic workload scenario can be a use case for the batch ingestion process? ○ Produce real-time alerts based on log data to identify potential fraud as soon as it occurs. ○ Populate a dashboard with real-time error rates of sensors in a factory. ○ Send small bits of clickstream data at a continuous pace from a retailer's website for immediate analysis. ◉ Send sales transaction data from a retailer's website to a central location periodically. Analyze the data overnight, and deliver reports to branches in the morning.	This is a good use case for batch ingestion because the data is pooled together and processed overnight.
4. A medical research company has a ribonucleic acid (RNA) sequencing machine that stores its private results to the lab's on-premises networked-attached storage. Their data science team wants to ingest these results into their AWS account. How should they ingest this data? ○ Use Amazon AppFlow to connect to the on-premises data and move the data into the pipeline. ○ Use AWS Database Migration Service (AWS DMS) to sync data from the on-premises file store to Amazon S3. ○ Use AWS Data Exchange to subscribe to the RNA-sequencing data. ◉ Use AWS DataSync to transfer data from the on-premises file store to an Amazon S3 bucket in the data lake.	DataSync is a purpose-built service to transfer data from file stores and can write to Amazon S3.
5. A data engineer has ingested a new JSON file into an Amazon S3 bucket in their data lake. An AWS Glue Data Catalog maintains metadata about data in the lake. Which feature of AWS Glue can the data engineer use to discover the JSON data schema with the fewest steps in a code-free way? ◉ Run an AWS Glue crawler on the S3 bucket. ○ Use AWS Glue Studio to write a script that converts the JSON data to Apache Parquet format. ○ Use AWS Glue Studio to transform the data and move it into the data warehouse. ○ Set up an AWS Glue workflow to orchestrate a set of jobs that transforms the data into an open columnar format.	When an AWS Glue crawler runs on an S3 bucket, the crawler automatically generates a schema and stores it with other metadata to the Data Catalog. This makes it discoverable for data lake consumers.

6. A data pipeline will ingest clickstream data from a shopping website. The data engineer must transform data as it arrives to feed a real-time analytics Amazon OpenSearch Service dashboard. They must also generate a monthly report based on the dashboard. Which configuration meets this need?

- (●) Use Amazon Kinesis Data Streams to capture the data. Use Amazon Kinesis Data Analytics to consume and transform data from the stream. Use Amazon Kinesis Data Firehose to deliver transformed data to OpenSearch Service.
- ( ) Use two data pipelines: one to ingest data into Amazon Kinesis Data Analytics and send it to OpenSearch Service and one to send the streaming data directly from Amazon Kinesis Data Streams to Amazon S3.
- ( ) Use Amazon Kinesis Data Streams to capture the data. Use Amazon Kinesis Data Analytics as a consumer to deliver data to OpenSearch Service. Use Amazon Redshift Spectrum to run SQL queries on the data stream for the monthly report.
- ( ) Use Amazon Kinesis Data Firehose to capture the data and send the data to Amazon S3. Run an AWS Glue crawler to support querying the data for the dashboard.

Kinesis Data Streams can ingest the data. Kinesis Data Analytics can also consume the data from the data stream and process it immediately to feed the OpenSearch Service dashboard by using Kinesis Data Streams. Kinesis Data Firehose can deliver data to storage and analytics destinations such as OpenSearch Service, where the report can be produced. Continue

7. Which statement accurately describes a consideration for designing pipeline storage

- ( ) Choose the lowest cost storage option regardless of the intended use case.
- ( ) Choose the storage option that provides the fastest queries regardless of the use case.
- ( ) Store raw data that will be used for analytics in a data warehouse.
- (●) Archive data out of relational databases into a more cost-efficient storage option.

It is common practice to archive data out of a relational database into a more cost-efficient storage option. Amazon S3 storage classes are purpose-built for varying access patterns at corresponding costs.

8. A data engineer is designing a low-cost infrastructure to store data directly from a central repository for both structured and unstructured data. Which option meets the data engineer's needs?

- ( ) Amazon Quantum Ledger Database (Amazon QLDB)
- (●) Amazon S3
- ( ) Amazon Redshift
- ( ) AWS Database Migration Service (AWS DMS)

This scenario describes a data lake. Data lakes are centralized repositories that developers can use to store structured and unstructured data regardless o scale. Amazon S3 is a good choice for this purpose.

9. A DevOps engineer is migrating an on-premises Apache Hadoop cluster to AWS. The cluster runs scheduled jobs by using parallel processing. Which AWS service is the MOST appropriate choice?

- ( ) AWS Glue
- ( ) AWS Glue DataBrew
- (●) Amazon EMR
- ( ) Amazon Kinesis Data Analytics

Amazon EMR supports multiple Apache Hadoop applications, including frameworks such as Hadoop MapReduce and Apache Spark.

10. A marketing manager quickly needs one-time insights about the number of leads closed deals across multiple postal codes. Which service would be the MOST cost-effective method to query daily aggregates of sales data stored in Amazon S3?

- (●) Amazon Athena
- ( ) Amazon Redshift
- ( ) Amazon QuickSight
- ( ) Amazon OpenSearch Service

Athena can be used for one-time querying and quick analysis of data directly on Amazon S3 and uses a pay-as-you-go pricing model.

• Five data characteristics are value, veracity, volume, velocity, and variety. • Consider the data characteristics together to make decisions about the infrastructure design for each business use case. • Value and veracity impact accurate insights that the business needs. Volume and velocity *together* drive the expected throughput and scaling requirements of your pipeline. • The goal of the modern data architecture is to store data in a centralized location and make it available to all consumers to perform analytics and run AI/ML applications.	• A data pipeline includes layers to ingest, store, process, and analyze data. • Data ingestion is the process of extracting data from its source and loading it into a data pipeline to be stored or analyzed. • In homogenous ingestion, data doesn't need to be transformed because the source and destination match. When transformation is needed, heterogenous ingestion can involve ETL or ELT patterns. • If the data needs to be analyzed in near real time, consider streaming ingestion methods instead of batch ingestion.
• To ingest data, consider purpose-built tools that will reduce undifferentiated lifting. • Amazon AppFlow can ingest from SaaS applications, such as Salesforce and Zendesk. • DataSync can ingest from file shares. • AWS Data Exchange provides customers with a way to find, subscribe to, and use third-party data in the cloud.	• Batch ingestion and processing make high-volume, repetitive tasks more efficient to run. • AWS Glue provides functionality for schema identification, data cataloging, data preparation and cleaning, ETL job authoring, and data quality assessment. • Use AWS Glue when an analytics use case doesn't require real-time aggregation or transformation of data.

• The Amazon Kinesis family provides fully managed data ingestion services:
  • Kinesis Data Firehose delivers near real-time streaming data to certain destinations. It provides short-lived retention of delivered data but lacks replay support.
  • Kinesis Data Streams collects and ingests large streams of data records in near real time. It offers the best low-latency data streaming properties but requires some configuration.
  • Kinesis Video Streams securely streams video from connected devices to AWS for analytics and other processing.
• Amazon Managed Service for Apache Flink analyzes streaming data.
• Amazon MSK is a fully managed Apache Kafka service that you can deploy in a virtual private cloud (VPC).
	• A modern data architecture is built on a data lake as a central component with analytics and storage services as its peripheral components.
	• A data lake architecture consists of data storage, a data catalog, data access security, and data transformation tools.
	• A data warehouse has a schema-on-write approach, and a data lake has a schema-on-read approach.
	• Design data storage by considering data origin, data shape, cost, query scope, and data retention periods.

- Big data parallel processing is a computing technique that breaks up the processing of large datasets into smaller parts.
- EMR clusters can use Hadoop MapReduce or Apache Spark frameworks to process data in parallel.
- Choose Amazon EMR to manage clusters.
- Choose EMR Serverless or AWS Glue for serverless solutions.

- QuickSight is a business intelligence tool to visualize data by using publishable dashboards.
- Athena SQL is a data query engine and data schema metadata store with the ability to query data stored in other services by using SQL queries.
- With Athena for Apache Spark, you can use Apache Spark to process parallel workloads of big data.
- OpenSearch Service indexes uploaded data to facilitate search and analysis of data.

A data engineer needs to **analyze and visualize** a lot of **streaming data** from **user activity logs** in real time. Which single service would provide this capability?

Choice	Response
A	Amazon OpenSearch Service
B	Amazon Athena
C	Amazon QuickSight
D	Amazon Redshift

A Amazon OpenSearch Service

MODULE-16

Planning for Disaster :

Key Concepts in Disaster Planning

Types of Failures:

1. **Small-Scale Events**: Affect individual resources (e.g., a server going offline).
2. **Large-Scale Events**: Involve multiple resources across Availability Zones.
3. **Global Events**: Widespread failures affecting numerous users and systems.

Philosophy of Failure:

1. Recognizing that failures are a normal part of system operation, as emphasized by AWS VP and CTO Werner Vogels.

Preparation Strategies:

1. **Fault Tolerance**: Designing systems with redundancy to withstand component failures.
2. **Backup**: Establishing robust backup protocols to protect critical data.
3. **Disaster Recovery**: Implementing comprehensive plans to restore operations after a disaster.

Factors Influencing Disaster Planning:

1. **Time Dependency**: How quickly issues must be resolved to avoid business impact.
2. **Data Loss Tolerance**: Acceptable amounts and types of data loss during recovery.
3. **Geographic Considerations**: Different recovery needs based on the location of resources.
4. **Cost vs. Risk**: Balancing preparation costs with potential risks to the business.

Defining RPO and RTO:

1. **Recovery Point Objective (RPO)**: Maximum acceptable data loss measured in time. For instance, if a disaster occurs 8 hours after the last backup, the RPO is 8 hours.
2. **Recovery Time Objective (RTO)**: Maximum acceptable downtime after a disaster. If recovery takes 2 hours, the RTO is 2 hours.

Business Continuity Plan (BCP):

1. A comprehensive approach that includes:

    1. Business impact analysis
    2. Risk assessment
    3. Disaster recovery planning
    4. Evaluating RPO and RTO

Conclusion

Developing a disaster recovery strategy is critical for organizations to mitigate the impact of failures. It requires understanding the acceptable limits of data loss, recovery times, and the broader context of business operations. A well-structured BCP is vital for maintaining service continuity, emphasizing the importance of regular evaluation and updates to the plan.

AWS disaster recovery planning:

· **EBS Volume Snapshots**: Create incremental backups, save costs, and allow data restoration to new volumes. Automate management with Amazon Data Lifecycle Manager.

· **File System Replication**: Use DataSync for fast transfers between EFS or FSx, with daily backups stored in S3. Implement multi-Region replication for disaster recovery.

· **Compute Recovery**: Automatically recover EC2 instances and use preconfigured AMIs (golden AMIs) for quick restoration.

· **EventBridge Failover**: Utilize global endpoints to reroute events based on service health across Regions.

· **Resiliency Design**: Route 53 for DNS load balancing, ELB for traffic distribution, and Direct Connect/VPN for secure AWS access.

· **Database Recovery**: Amazon RDS for snapshots and read replicas; DynamoDB for point-in-time recovery and global tables.

· **Environment Replication**: CloudFormation for automated infrastructure deployment; OpsWorks for application management with automatic host replacement.

Disaster recovery patterns :

Disaster Recovery Patterns

Backup and Restore

1. **Description**: Regular backups stored in Amazon S3. Suitable for data loss mitigation.
2. **RPO/RTO**: Higher RTO; lower cost.
3. **Implementation**:
    1. **Preparation**: Create backups, store in S3, document restore procedures.
    2. **Disaster Response**: Retrieve backups, restore infrastructure, route traffic to the new system.

Pilot Light

1. **Description**: A minimal backup version of the environment is always running. Faster recovery than backup and restore.
2. **RPO/RTO**: Moderate RTO; moderate cost.
3. **Implementation**:

1. **Preparation**: Configure EC2 instances, maintain AMIs of key servers, regularly test and update.
2. **Disaster Response**: Bring up resources around the core dataset, scale as needed, switch traffic to the new system.

Warm Standby

1. **Description**: A scaled-down version of a fully functional environment running continuously.
2. **RPO/RTO**: Lower RTO than pilot light; higher cost.
3. **Implementation**: Keep resources running, replicate data, and scale quickly during a disaster.

Multi-Site

1. **Description**: Fully functional environments running in multiple regions, providing high availability.
2. **RPO/RTO**: Lowest RTO; highest cost.
3. **Implementation**: Active-active setup in multiple locations, ensuring continuous availability and immediate failover.

Key Concepts

- **Recovery Point Objective (RPO)**: Maximum acceptable data loss time.
- **Recovery Time Objective (RTO)**: Maximum acceptable downtime after a disaster.

Conclusion

Choosing the right pattern depends on the organization's RPO, RTO, and budget. Effective disaster recovery planning minimizes the impact of disasters by ensuring readiness and quick recovery.

Applying AWS Well-Architected Framework principles to disaster planning :

Key Best Practices for Disaster Planning

Failure Management - Planning for Disaster Recovery

1. **Define Recovery Objectives (RTO & RPO)**:
    1. Understand the impact of downtime and data loss on business.
    2. RTO (Recovery Time Objective) and RPO (Recovery Point Objective) are critical for selecting a disaster recovery (DR) strategy.
2. **Use Defined Recovery Strategies**:

    1. Choose strategies like backup and restore, pilot light, warm standby, or multi-site based on the trade-off between downtime/data loss and cost/complexity.

3. **Test Disaster Recovery Implementation**:

    1. Regularly conduct failover tests to ensure RTO and RPO are met.

Operational Excellence - Managing Workload and Operations Events

4. **Define a Customer Communication Plan for Outages**:

    1. Establish a communication plan to inform customers and stakeholders about outages and service restoration.
    2. Test the communication plan biannually to ensure effectiveness.

Identity and Access Management - Permissions Management

5. **Establish an Emergency Access Process**:

    1. Create processes to provide emergency access in case of issues with centralized identity providers.
    2. Document and test emergency access processes to minimize downtime and maintain service availability.

Key Takeaways

- **Recovery Objectives**: Establish and define RTO and RPO based on business needs.
- **Recovery Strategies**: Implement strategies tailored to meet defined objectives.
- **Testing**: Regularly validate the disaster recovery plan to ensure it functions as intended.
- **Communication**: Maintain a clear communication strategy during outages.
- **Emergency Access**: Prepare for emergencies with documented access processes.

These principles are essential for creating resilient cloud architectures and ensuring business continuity in the face of disruptions.

1. What are the definitions for recovery point objective (RPO) and recovery time objective (RTO)? ○ RPO is the maximum acceptable data loss, measured in bytes. RTO is the average amount of time required to recover ○ RPO is the target time until recovery. RTO is the average amount of time to recover. ○ RPO is the maximum acceptable data loss, measured in bytes. RTO is the maximum acceptable data loss, measured in time. ◉ RPO is the maximum acceptable data loss, measured in time. RTO is the maximum acceptable time until recovery.	RPO answers the question: How often must your data be backed up? RTO answers the question: How quickly must your applications and data be recovered?
2. What can you do to quickly replicate or redeploy environments in a disaster? ◉ Use AWS CloudFormation templates to deploy duplicate environments in the same Region. ○ Use AWS CodeBuild to deploy application containers in a new virtual private cloud (VPC). ○ Use AWS Elastic Beanstalk to deploy a new virtual private cloud (VPC) and subnets in a different Region. ○ Use AWS OpsWorks to rebuild Amazon RDS instances.	AWS CloudFormation templates enable you to treat infrastructure as code. By using this method, you can standardize deployments. AWS CloudFormation StackSets enable you to create, update, or delete stacks across multiple accounts and Regions with a single operation.

3. A company stores data in an Amazon S3 bucket. Which solution provides the most efficient way to ensure that all new and existing objects and metadata are copied to another Region for disaster recovery (DR)? ○ Create a workflow with AWS Step Functions and AWS Lambda to synchronize the buckets. ○ Use an AWS Lambda function to copy objects so that all object create events initiate the function. ○ Copy existing objects to the target bucket, and configure clients to write new files to both buckets. ◉ Enable cross-Region replication on the bucket and copy existing objects onto themselves.	Cross-Region replication copies all new objects and overwrites of existing objects to a bucket in another Region and preserves the object metadata.
4. Which strategy is the most efficient for Amazon EC2 disaster recovery (DR)? ◉ Store essential data separately from the instance, and develop rapid rebuild processes for compute instances. ○ Synchronize instances with standby instances on nearly a continuous basis. ○ Rebuild instances by using Amazon Machine Images (AMIs) from AWS Marketplace. ○ Back up instances on a regular schedule.	Storing essential data separately from the instance, such as in Amazon Elastic Block Store (Amazon EBS) volumes and Amazon Elastic File System (Amazon EFS) file systems, is the most efficient strategy. It enables you to rebuild undifferentiated parts of instances from AMIs and code that is stored in repositories.
5. Which service provides automatic failover between multiple endpoints in support of a geographic disaster recovery (DR) strategy? ○ Elastic Load Balancing (ELB) ◉ Amazon Route 53 ○ AWS Direct Connect ○ Amazon Virtual Private Cloud (Amazon VPC)	You can configure Route 53 to fail over between multiple endpoints in different Regions.
6. Which statement about the backup and restore disaster recovery (DR) pattern is true? ○ Most cost-effective, but highest recovery point objective (RPO) ○ Less cost effective, but lowest recovery time objective (RTO) ◉ Most cost-effective, but highest recovery time objective (RTO) ○ Less cost effective, but lowest recovery point objective (RPO)	This pattern is the most cost-effective because you primarily pay for data storage and do not pay to maintain running systems.
7. Which statements accurately describe the infrastructure characteristics of common disaster recover (DR) patterns? (Select TWO.) ☐ Pilot light runs with some infrastructure at full capacity. The rest scales up when a disaster occurs. ☑ Warm standby has a scaled-down version of all infrastructure that scales as necessary and within pre-defined limits to meet the load when a disaster occurs. ☐ Pilot light has a scaled-down version of all infrastructure that runs until a disaster occurs. ☐ Warm standby has a second fully functional set of infrastructure that runs all the time. ☑ Pilot light has minimal infrastructure that always runs. The rest of the infrastructure does not run until a disaster occurs.	Pilot light has minimal infrastructure that runs all the time, and the rest of the infrastructure starts when a disaster occurs. Warm standby has a scaled-down version of all infrastructure that scales as necessary and within pre-defined limits to meet the load when a disaster occurs.

8. What does the multi-site disaster recovery (DR) pattern involve?

- ○ It involves failover to another site that is not running until it is needed.
- ○ The load is distributed across multiple geographically separated sites to reduce the impact of disasters.
- ○ Backups are stored in different sites so that they are protected if a disaster occurs.
- ◉ It involves automatic failover to a second fully functional, constantly operational system that is in another site.

In AWS, you can implement this pattern with infrastructure in a second Region or virtual private cloud (VPC). You can also use infrastructure in AWS as a second site for your on-premises infrastructure.

9. A company requires a disaster recovery (DR) solution for a business-critical application that provides a recovery time objective (RTO) and recovery point objective (RPO) in minutes. However, they do not want to pay for more than what they need. Which DR pattern would most likely meet these requirements?

- ○ Multi-site
- ○ Pilot light
- ○ Backup and restore
- ◉ Warm standby

Warm standby can provide RTO and RPO in minutes. However, it comes with a tradeoff of higher cost compared to pilot light or backup and restore. It is less expensive than a multi-site approach.

10. What does an AWS Storage Gateway enable you to do? (Select THREE.)

- ☐ Provide a fully managed elastic Network File System (NFS) endpoint to systems in a virtual private cloud (VPC) and on-premises.
- ☑ Use Server Message Block (SMB) or Network File System (NFS) to connect to Amazon S3.
- ☐ Give applications in a virtual private cloud (VPC) access to on-premises block storage volumes.
- ☑ Present cloud-based internet Small Computer Systems Interface (iSCSI) block storage volumes to on-premises applications.
- ☑ Transfer backup jobs from tape or Virtual Tape Library (VTL) systems to the cloud.
- ☐ Connect to Amazon S3 through an API.

Amazon S3 File Gateway offers SMB or NFS-based access to data in Amazon S3 with local caching. Volume Gateway presents cloud-based iSCSI block storage volumes to on-premises applications. AWS Storage Gateway can act as a VTL that is backed by Amazon S3.

- Failures can occur at any time and on a small, large, or global scale.
- A disaster recovery plan will help limit business and customer impact when a disaster occurs.
- RPO is the maximum acceptable amount of data loss after an unplanned data-loss incident.
- RTO is the amount of time an application, system, and process can be down without causing significant damage to the business.
- A BCP is a system of prevention and recovery from potential threats to a company that includes RPO and RTO.

- Use features such as CRR, EBS volume snapshots, and Amazon RDS snapshots to protect data.
- Use networking features, such as Route 53 failover and ELB, to improve application availability.
- Use automation services, such as CloudFormation, as part of your DR strategy to quickly deploy duplicate environments when needed.
- Use EventBridge to use global endpoints to automatically restore data and failover to functioning servers.

• Common disaster recovery patterns on AWS include backup and restore, pilot light, warm standby, and multi-site. • Backup and restore is the most cost-effective approach. However, it has the highest RTO. • Multi-site provides the fastest RTO. However, it costs the most because it provides a fully running production-ready duplicate. • Storage Gateway provides three interfaces—File Gateway, Volume Gateway, and Tape Gateway—for data backup and recovery between on-premises and the AWS Cloud.	

-

A solutions architect must create a disaster recovery (DR) solution for a company's **business-critical** applications. The maximum acceptable amount of data loss is 7 minutes. The DR solution also requires the deployment of a completely **functional version of the applications to handle the majority of traffic immediately**, and then scale up to full capacity over time. Which disaster recovery pattern would provide the most **cost-effective** solution?

Choice	Response
A	Backup and restore
B	Pilot light
C	Warm standby
D	Multi-site

C Warm standby