AWS Certified Cloud Practitioner

Comprehensive Study Guide

Exam Code: CLF-C02

AWS Certification Preparation

December 23, 2025

Contents

Chapter 1

Introduction

1.1 About This Guide

This comprehensive study guide is designed to help you prepare for and pass the **AWS Certified Cloud Practitioner (CLF-C02)** exam. This certification validates your overall understanding of the AWS Cloud, independent of specific technical roles.

1.2 Certification Overview

- **Exam Code:** CLF-C02
- **Duration:** 90 minutes
- **Question Format:** 65 questions (multiple choice and multiple response)
- **Passing Score:** 700 out of 1000
- **Cost:** $100 USD
- **Validity:** 3 years
- **Delivery:** Pearson VUE testing center or online proctored

1.3 Exam Domain Breakdown

Domain	Percentage
Domain 1: Cloud Concepts	24%
Domain 2: Security and Compliance	30%
Domain 3: Cloud Technology and Services	34%
Domain 4: Billing, Pricing, and Support	12%

Table 1.1: AWS Cloud Practitioner Exam Domains

1.4 Target Audience

This certification is ideal for:

- Individuals new to AWS Cloud
- Sales and marketing professionals
- Business analysts and project managers
- IT professionals transitioning to cloud
- Students and recent graduates
- Anyone seeking foundational AWS knowledge

Exam Tip

No technical prerequisites are required, but 6 months of exposure to AWS Cloud is recommended for success.

Chapter 2

Domain 1: Cloud Concepts (24%)

2.1 What is Cloud Computing?

2.1.1 Definition

Cloud computing is the **on-demand delivery** of IT resources over the Internet with **pay-as-you-go pricing**. Instead of buying, owning, and maintaining physical data centers and servers, you access technology services on an as-needed basis.

2.1.2 Six Advantages of Cloud Computing

1. **Trade capital expense for variable expense**
    - Pay only for what you consume
    - No upfront infrastructure costs
    - Lower Total Cost of Ownership (TCO)
2. **Benefit from massive economies of scale**
    - AWS achieves higher economies of scale
    - Lower pay-as-you-go prices
    - Prices decrease over time
3. **Stop guessing capacity**
    - Scale up or down based on demand
    - No over or under provisioning
    - Elastic resources
4. **Increase speed and agility**
    - Resources available in minutes
    - Faster experimentation and innovation
    - Reduced time to market
5. **Stop spending money running and maintaining data centers**

- Focus on business differentiators
- AWS manages infrastructure
- Reduce operational burden

6. **Go global in minutes**
   - Deploy applications globally
   - Low latency for users worldwide
   - Multiple AWS Regions available

2.2 Cloud Computing Models

2.2.1 Infrastructure as a Service (IaaS)

- Basic building blocks for cloud IT
- Highest level of flexibility and control
- Example: Amazon EC2, Amazon S3
- You manage: OS, applications, data
- AWS manages: Hardware, networking, facilities

2.2.2 Platform as a Service (PaaS)

- Removes need to manage underlying infrastructure
- Focus on deployment and management of applications
- Example: AWS Elastic Beanstalk, AWS Lambda
- You manage: Applications, data
- AWS manages: Runtime, middleware, OS, servers

2.2.3 Software as a Service (SaaS)

- Completed product run and managed by service provider
- End-user applications
- Example: Amazon WorkMail, Amazon Chime
- You manage: User access, data input
- AWS manages: Everything else

2.3 Cloud Deployment Models

2.3.1 Cloud (Public Cloud)

- Fully deployed in the cloud
- All parts of application run in cloud
- Applications built on cloud or migrated
- Can be built on low-level infrastructure or higher-level services

2.3.2 Hybrid

- Connects cloud resources to on-premises infrastructure
- Integrates cloud with existing infrastructure
- Useful for legacy applications
- Common deployment model for many enterprises
- Uses AWS Direct Connect, VPN

2.3.3 On-Premises (Private Cloud)

- Resources deployed using virtualization and resource management tools
- Sometimes called "private cloud"
- Uses AWS Outposts for on-premises AWS infrastructure
- Increased resource utilization

2.4 AWS Well-Architected Framework

The AWS Well-Architected Framework describes key concepts, design principles, and architectural best practices for designing and running workloads in the cloud.

2.4.1 Six Pillars

1. Operational Excellence

- Run and monitor systems to deliver business value
- Continually improve supporting processes and procedures
- Key principles: Perform operations as code, annotate documentation, make frequent small reversible changes
- Services: AWS CloudFormation, AWS Config, AWS CloudTrail, Amazon CloudWatch

2. Security

- Protect information, systems, and assets
- Key principles: Implement strong identity foundation, enable traceability, apply security at all layers
- Services: AWS IAM, AWS Organizations, AWS KMS, AWS Shield, Amazon GuardDuty

3. Reliability

- Ensure workload performs its intended function correctly and consistently
- Recover from failures and dynamically acquire computing resources
- Key principles: Automatically recover from failure, test recovery procedures, scale horizontally
- Services: Amazon RDS Multi-AZ, AWS Auto Scaling, Amazon CloudWatch

4. Performance Efficiency

- Use computing resources efficiently to meet requirements
- Maintain efficiency as demand changes and technologies evolve
- Key principles: Democratize advanced technologies, go global in minutes, use serverless architectures
- Services: AWS Lambda, Amazon EBS, Amazon RDS, AWS Auto Scaling

5. Cost Optimization

- Run systems to deliver business value at lowest price point
- Key principles: Implement cloud financial management, adopt consumption model, measure overall efficiency
- Services: AWS Cost Explorer, AWS Budgets, Reserved Instances, Savings Plans

6. Sustainability

- Minimize environmental impacts of running cloud workloads
- Key principles: Understand your impact, establish sustainability goals, maximize utilization
- Services: Amazon EC2 Auto Scaling, AWS Lambda, Amazon S3 Intelligent-Tiering

Key Point

The Well-Architected Framework is frequently tested on the exam. Understand each pillar's purpose and key services.

2.5 Cloud Economics

2.5.1 Total Cost of Ownership (TCO)

- Financial estimate to identify direct and indirect costs
- Compare on-premises vs. cloud costs
- Includes: Server costs, storage costs, network costs, IT labor costs
- AWS TCO Calculator helps estimate savings

2.5.2 Capital Expenditure (CapEx) vs. Operational Expenditure (OpEx)

CapEx (On-Premises):

- Upfront purchase of physical infrastructure
- Fixed, sunk cost
- Depreciates over time
- Requires capacity planning

OpEx (Cloud):

- Pay for what you use
- Variable cost based on consumption
- No upfront commitment
- Scales with business needs

2.5.3 Migration Strategies (6 R's)

1. **Rehosting (Lift and Shift)**
    - Move applications without changes
    - Fastest migration approach
    - Optimize after migration
2. **Replatforming (Lift, Tinker, and Shift)**
    - Make a few cloud optimizations
    - Don't change core architecture
    - Example: Migrate database to RDS
3. **Repurchasing**
    - Move to a different product

- Often SaaS platforms
- Example: CRM to Salesforce

4. **Refactoring/Re-architecting**
    - Reimagine how application is architected
    - Use cloud-native features
    - Most expensive but highest benefit
5. **Retire**
    - Identify IT assets that are no longer useful
    - Shut down and remove from portfolio
6. **Retain**
    - Keep applications on-premises
    - Not ready to migrate
    - Hybrid deployment

Chapter 3

Domain 2: Security and Compliance

3.0.1 AWS Shared Responsibility Model

Exam Tip

This is one of the most critical concepts for the exam. Understand what AWS manages versus what the customer manages.

The Shared Responsibility Model divides security responsibilities between AWS and the customer. Think of it as "Security OF the Cloud" (AWS) vs "Security IN the Cloud" (Customer).

AWS Responsibility: Security OF the Cloud

AWS is responsible for protecting the infrastructure that runs all services:

- **Physical security of data centers**
- Building access controls
- Security personnel
- Environmental safeguards
- **Hardware and networking components**
- Physical servers
- Storage devices
- Network equipment
- **Compute, storage, database, and networking infrastructure**
- Hypervisor layer
- Managed service infrastructure
- **AWS global infrastructure**
- Regions

- Availability Zones
- Edge Locations
- **Managed services**
- RDS, DynamoDB, Lambda, etc.
- AWS handles OS patching and maintenance for these services

Customer Responsibility: Security IN the Cloud

Customers are responsible for:

- **Customer data**
- All data you store in AWS
- Classification and protection
- **Platform, applications, Identity and Access Management (IAM)**
- Application code
- IAM users, groups, roles, and policies
- **Operating system, network, and firewall configuration**
- OS patches and updates (for EC2)
- Security group rules
- Network ACLs
- **Client-side data encryption and data integrity authentication**
- Encrypting data before upload
- Data validation
- **Server-side encryption (file system and/or data)**
- Encryption at rest
- Key management choices
- **Network traffic protection**
- Encryption in transit (HTTPS, TLS)
- Network security
- **Security group configuration**
- Firewall rules
- Access controls

- **User access management**
- Creating and managing users
- Password policies
- MFA enforcement

Shared Controls

Both AWS and customers have responsibilities for:

Control	AWS Responsibility	Customer Responsibility
Patch Management	Patches infrastructure components	Patches guest OS and applicat
Configuration Management	Configures infrastructure devices	Configures databases and appl
Awareness and Training	Trains AWS employees	Trains their own staff

Exam Tip

For any security question, ask: "Who is responsible?" AWS handles the infrastructure; you handle what you put in the cloud.

—

3.0.2 AWS Identity and Access Management (IAM)

IAM enables you to securely control access to AWS services and resources. It's a **global service** (not region-specific) and is **free** to use.

Core Components

Users

- **Individual people or services**
- Permanent named operators
- Can have long-term credentials:
- Password (for console access)
- Access keys (for programmatic access)
- Should represent a physical person or application
- By default, new users have NO permissions

Groups

- **Collection of users**
- Simplifies permission management
- Key characteristics:
- Groups cannot be nested
- Users can belong to multiple groups
- Apply policies to groups for easier management
- No default groups

Roles

- **Temporary credentials** for users, applications, or services
- No username/password or access keys
- Can be assumed by anyone who needs it
- **Best practice** for EC2 instances accessing AWS services
- Can be used for cross-account access
- Temporary security credentials are automatically rotated

Policies

- **JSON documents** defining permissions
- Attached to users, groups, or roles
- Define what actions are allowed or denied on which resources
- Follow **principle of least privilege**
- Two main types:
- **AWS Managed Policies:** Created and maintained by AWS
- **Customer Managed Policies:** Created and maintained by you

Example Policy Structure:

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Allow",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-bucket/*"
    \}
  ]
\}
```

Detailed IAM Policy Examples

Understanding IAM policies is critical for the exam. Here are common policy scenarios you should know.

Example 1: S3 Read-Only Access to Specific Bucket This policy grants read-only access to objects in a specific S3 bucket.

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Sid": "S3ReadOnlyAccess",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:GetObjectVersion",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::company-reports/*",
        "arn:aws:s3:::company-reports"
      ]
    \}
  ]
\}
```

Key Points:

- `Sid`: Statement ID (optional, for documentation)
- `Effect`: Can be "Allow" or "Deny"
- `Action`: What operations are permitted
- `Resource`: Which AWS resources the policy applies to

Example 2: EC2 Instance Management with Conditions This policy allows starting and stopping EC2 instances only during business hours.

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances",
        "ec2:DescribeInstances"
      ],
      "Resource": "*",
      "Condition": \{
        "DateGreaterThan": \{
```

```
          "aws:CurrentTime": "2024-01-01T08:00:00Z"
        \},
        "DateLessThan": \{
          "aws:CurrentTime": "2024-12-31T18:00:00Z"
        \},
        "IpAddress": \{
          "aws:SourceIp": [
            "203.0.113.0/24",
            "198.51.100.0/24"
          ]
        \}
      \}
    \}
  ]
\}
```

Condition Elements:

- Time-based restrictions
- IP address restrictions
- MFA requirements
- Source VPC restrictions

Example 3: Deny Policy for Sensitive Actions This policy explicitly denies deletion of production resources (deny always overrides allow).

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Deny",
      "Action": [
        "rds:DeleteDBInstance",
        "ec2:TerminateInstances",
        "s3:DeleteBucket"
      ],
      "Resource": "*",
      "Condition": \{
        "StringEquals": \{
          "aws:ResourceTag/Environment": "Production"
        \}
      \}
    \}
  ]
\}
```

Important: Explicit Deny always wins over Allow in IAM policy evaluation.

Example 4: MFA-Required Policy This policy requires MFA for sensitive operations.

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Allow",
      "Action": [
        "ec2:TerminateInstances",
        "rds:DeleteDBInstance"
      ],
      "Resource": "*",
      "Condition": \{
        "BoolIfExists": \{
          "aws:MultiFactorAuthPresent": "true"
        \}
      \}
    \}
  ]
\}
```

Example 5: Cross-Account Access Policy This policy allows assuming a role from another AWS account.

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Allow",
      "Principal": \{
        "AWS": "arn:aws:iam::123456789012:root"
      \},
      "Action": "sts:AssumeRole",
      "Condition": \{
        "StringEquals": \{
          "sts:ExternalId": "UniqueExternalId123"
        \}
      \}
    \}
  ]
\}
```

Use Case: Allow users from Account A to access resources in Account B.

Example 6: Full Administrator Access This policy grants full access to all AWS services (use with extreme caution).

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
```

```
      "Effect": "Allow",
      "Action": "*",
      "Resource": "*"
    \}
  ]
\}
```

Warning: Only assign to trusted administrators. This is equivalent to root access.

Example 7: Read-Only Access Across All Services This policy provides read-only access for auditing purposes.

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Allow",
      "Action": [
        "ec2:Describe*",
        "s3:Get*",
        "s3:List*",
        "rds:Describe*",
        "cloudwatch:Get*",
        "cloudwatch:List*",
        "cloudtrail:LookupEvents"
      ],
      "Resource": "*"
    \}
  ]
\}
```

Example 8: S3 Bucket Policy for Public Read Access This is a resource-based policy attached to an S3 bucket.

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Sid": "PublicReadGetObject",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::my-public-website/*"
    \}
  ]
\}
```

Use Case: Hosting a static website or public content.

Example 9: Service Control Policy (SCP) This SCP prevents anyone in an OU from leaving the organization.

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Deny",
      "Action": [
        "organizations:LeaveOrganization"
      ],
      "Resource": "*"
    \}
  ]
\}
```

Important: SCPs affect all users and roles, including the account root user.

Example 10: Tag-Based Access Control This policy allows actions only on resources with specific tags.

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Allow",
      "Action": [
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": "arn:aws:ec2:*:*:instance/*",
      "Condition": \{
        "StringEquals": \{
          "aws:ResourceTag/Owner": "\$\{aws:username\}",
          "aws:ResourceTag/Department": "Engineering"
        \}
      \}
    \}
  ]
\}
```

Use Case: Users can only manage their own resources in their department.

IAM Best Practices

1. **Root account protection**

- Use only for initial setup, then lock it away
- Enable MFA on root account
- Do not create access keys for root
- Create individual IAM users instead

1. **Principle of Least Privilege**

- Grant only the permissions required to perform a task
- Start with minimum permissions and add as needed
- Regularly review and remove unnecessary permissions

1. **Use Groups for permission management**

- Assign permissions to groups, not individual users
- Add users to appropriate groups
- Easier to manage and audit

1. **Enable MFA (Multi-Factor Authentication)**

- Especially for privileged users
- Required for root account
- Add extra layer of security

1. **Use Roles for applications**

- For applications running on EC2
- Better than embedding credentials
- Automatic credential rotation

1. **Rotate Credentials regularly**

- Change passwords periodically
- Rotate access keys
- Set password expiration policies

1. **Remove Unnecessary Credentials**

- Delete unused users
- Remove unused roles
- Deactivate old access keys

1. **Use Policy Conditions for extra security**

- IP address restrictions
- Time-based access
- MFA requirements
- Source VPC restrictions

—

3.0.3 Security Best Practices - Deep Dive

Comprehensive security best practices you must know for the exam and real-world AWS usage.

1. Identity and Access Management Security

Implement Least Privilege Access **What it means:** Grant only the permissions necessary to perform required tasks, nothing more.

How to implement:

- Start with zero permissions and add what's needed
- Use AWS managed policies as a starting point, then customize
- Regularly review and audit permissions
- Use IAM Access Analyzer to identify unused permissions
- Remove permissions that haven't been used in 90+ days

Example Scenario: A developer needs to read application logs from S3. Give them `s3:GetObject` on the specific bucket, not full S3 access or administrator rights.

Exam Tip: Questions will test whether you can identify overly permissive policies.

Implement Strong Password Policies **Requirements:**

- Minimum length: 14+ characters (AWS allows 8-128)
- Require uppercase, lowercase, numbers, and symbols
- Prevent password reuse (remember at least 24 previous passwords)
- Enforce password expiration (60-90 days recommended)
- Prevent users from changing their password too frequently

AWS Password Policy Settings:

```
- Minimum password length: 14 characters
- Require at least one uppercase letter: Yes
- Require at least one lowercase letter: Yes
- Require at least one number: Yes
- Require at least one non-alphanumeric character: Yes
- Allow users to change their own password: Yes
- Enable password expiration: Yes (90 days)
- Password expiration requires administrator reset: No
- Number of passwords to remember: 24
```

Credential Management Best Practices Never do this:

- Hard-code credentials in application code
- Store credentials in version control (Git)
- Share credentials between users or applications
- Email or message credentials
- Use long-term credentials when temporary ones are available

Always do this:

- Use IAM roles for EC2 instances and Lambda functions
- Use AWS Secrets Manager or Systems Manager Parameter Store for secrets
- Rotate credentials regularly (access keys every 90 days)
- Use temporary credentials via AWS STS
- Delete unused credentials immediately

Enable AWS CloudTrail in All Regions Why it's critical:

- Provides audit trail of all API calls
- Helps with compliance requirements
- Enables security analysis and troubleshooting
- Detects unauthorized access attempts
- Required for incident response

Configuration:

- Enable in all regions (even ones you don't use)
- Enable log file validation for integrity
- Store logs in a separate, secured S3 bucket
- Enable S3 bucket versioning for log storage
- Restrict access to CloudTrail logs
- Set up CloudWatch Logs integration for real-time monitoring

2. Network Security Best Practices

Use Security Groups Properly Key Principles:

- Security groups are stateful (return traffic automatically allowed)
- Default deny: Allow only what's needed
- Use descriptive names and tags
- Reference other security groups instead of IP addresses when possible
- Separate security groups by tier (web, application, database)

Common Patterns:
Web Tier Security Group:

```
Inbound:
- Port 80 (HTTP) from 0.0.0.0/0
- Port 443 (HTTPS) from 0.0.0.0/0
- Port 22 (SSH) from Bastion-SG only

Outbound:
- All traffic (default)
```

Application Tier Security Group:

```
Inbound:
- Port 8080 from Web-Tier-SG
- Port 22 from Bastion-SG only

Outbound:
- Port 3306 to Database-SG
- Port 443 to 0.0.0.0/0 (for API calls)
```

Database Tier Security Group:

```
Inbound:
- Port 3306 (MySQL) from App-Tier-SG only
- Port 22 from Bastion-SG only

Outbound:
- None (most restrictive)
```

Network ACL (NACL) Best Practices Differences from Security Groups:

- Stateless (must allow return traffic explicitly)
- Applies at subnet level
- Rules are processed in numerical order
- Can have explicit DENY rules

When to use NACLs:

- Block specific IP addresses (security groups can't deny)
- Add an additional layer of defense
- Comply with regulatory requirements for network segmentation

Best Practice:

- Use security groups as primary defense
- Use NACLs for additional subnet-level protection
- Leave room between rule numbers (100, 200, 300) for insertions
- Document all custom NACL rules

Implement VPC Flow Logs What they capture:

- Accepted and rejected traffic
- Source and destination IP addresses
- Ports and protocols
- Packet and byte counts

Use cases:

- Troubleshoot connectivity issues
- Monitor traffic patterns
- Detect anomalous behavior
- Meet compliance requirements
- Security forensics

Configuration:

- Enable at VPC, subnet, or ENI level
- Publish to CloudWatch Logs or S3
- Use for analysis with Amazon Athena
- Integrate with security tools

3. Data Protection Best Practices

Encrypt Data at Rest Services with encryption:

- S3: SSE-S3, SSE-KMS, SSE-C
- EBS: Encrypted volumes
- RDS: Encrypted databases
- DynamoDB: Encryption at rest
- Redshift: Encrypted clusters

Best practices:

- Enable encryption by default for all new resources
- Use AWS KMS for key management
- Implement automatic key rotation
- Separate keys for different data classifications
- Grant minimal permissions to decrypt

Encrypt Data in Transit How to implement:

- Use HTTPS/TLS for all web traffic
- Use SSL/TLS for database connections
- Use VPN or Direct Connect for hybrid connectivity
- Enable encryption for all data transfers
- Use AWS Certificate Manager for SSL/TLS certificates

Services that enforce encryption in transit:

- CloudFront (can require HTTPS)
- API Gateway (HTTPS only)
- Application Load Balancer (SSL/TLS termination)
- S3 Transfer Acceleration (HTTPS)

Implement Backup and Recovery Best practices:

- Enable automated backups for databases
- Use AWS Backup for centralized backup management
- Store backups in different region for disaster recovery
- Test restore procedures regularly
- Implement versioning for S3 objects
- Use lifecycle policies to manage backup retention

4. Monitoring and Logging Best Practices

Implement Comprehensive Logging Essential logs to enable:

- CloudTrail: API activity
- VPC Flow Logs: Network traffic
- S3 Server Access Logs: S3 bucket access
- ELB Access Logs: Load balancer requests
- CloudFront Access Logs: CDN requests
- RDS Logs: Database queries and errors

Log management:

- Centralize logs in a dedicated account
- Enable log file integrity validation
- Implement log retention policies
- Protect logs from deletion or modification
- Use CloudWatch Logs Insights for analysis

Set Up Security Alerts Critical alerts to configure:

- Root account usage
- IAM policy changes
- Security group changes
- Network ACL changes
- Failed login attempts (multiple)
- Unauthorized API calls
- Changes to CloudTrail configuration
- S3 bucket policy changes
- Encryption key deletions

Alerting mechanisms:

- CloudWatch Alarms
- SNS notifications
- EventBridge rules
- GuardDuty findings
- Security Hub alerts

5. Compliance and Governance Best Practices

Implement Automated Compliance Checking Tools to use:

- AWS Config Rules for continuous compliance
- AWS Security Hub for centralized security view
- AWS Systems Manager for patch compliance
- Trusted Advisor for best practice checks

Common compliance rules:

- Ensure S3 buckets are not publicly accessible
- Ensure encryption is enabled on all volumes
- Ensure MFA is enabled for root account
- Ensure CloudTrail is enabled in all regions
- Ensure unused IAM credentials are removed

Tag Everything for Governance Essential tags:

- Environment (Production, Staging, Dev)
- Owner (team or individual)
- Cost Center (for billing)
- Project (application or project name)
- Compliance (required compliance programs)
- Data Classification (Public, Internal, Confidential)

Benefits:

- Cost allocation and tracking
- Automated policy enforcement
- Resource organization
- Compliance reporting
- Lifecycle management

6. Incident Response Best Practices

Prepare for Security Incidents Have a plan:

- Document incident response procedures
- Define roles and responsibilities
- Maintain contact lists
- Establish communication channels
- Practice with simulation exercises

AWS tools for incident response:

- CloudWatch for monitoring and alerts
- CloudTrail for forensic analysis
- VPC Flow Logs for network analysis
- AWS Systems Manager for automated remediation
- EC2 snapshot for forensic investigation

Isolation procedures:

- Change security group to deny all traffic
- Snapshot affected resources before investigation
- Isolate in separate VPC or subnet
- Preserve logs and evidence
- Follow chain of custody procedures

7. Application Security Best Practices

Implement Defense in Depth Multiple layers of security:

1. Edge security: CloudFront, AWS WAF, Shield
2. Network security: VPC, Security Groups, NACLs
3. Application security: IAM roles, encryption
4. Data security: Encryption at rest, access controls
5. Monitoring: CloudTrail, GuardDuty, CloudWatch

Benefits:

- No single point of failure
- Multiple chances to detect and stop attacks
- Reduces blast radius of breaches
- Compliance requirement for many frameworks

Secure API Endpoints API Gateway security:

- Use API keys for identification
- Implement throttling and rate limiting
- Enable AWS WAF for protection
- Use Lambda authorizers for custom authentication
- Implement request validation
- Enable CloudWatch Logs for monitoring

Best practices:

- Use HTTPS only
- Implement proper authentication and authorization
- Validate all inputs
- Use least privilege for Lambda execution roles
- Enable CORS correctly (don't use *)
- Implement API versioning

8. Third-Party Security Best Practices

Manage Third-Party Access Securely Use external IDs for cross-account access:

- Prevents confused deputy problem
- Unique identifier per customer
- Include in AssumeRole policy condition

Best practices:

- Use IAM roles instead of sharing credentials
- Implement least privilege access
- Require MFA for sensitive operations
- Monitor third-party access with CloudTrail
- Regularly audit and review access
- Remove access when no longer needed

Secure Container and Serverless Workloads Container security:

- Scan images for vulnerabilities (Amazon ECR scanning)
- Use minimal base images
- Don't run containers as root
- Implement least privilege for task roles
- Use secrets management for credentials
- Enable CloudTrail logging for ECR

Lambda security:

- Use separate execution roles per function
- Store secrets in Secrets Manager or Parameter Store
- Enable VPC access only when needed
- Implement function-level encryption
- Use environment variables for configuration
- Monitor with CloudWatch and X-Ray

Multi-Factor Authentication (MFA)

MFA adds an extra layer of protection beyond username and password.

Authentication Factors:

- **Something you know:** Password
- **Something you have:** MFA device

MFA Device Options:

Type	Description	Use Case
Virtual MFA Device	Mobile app (Google Authenticator, Authy)	Most common, convenient
Hardware MFA Device	Physical token (YubiKey)	High security environments
SMS Text Message	Code sent via SMS	Not recommended for root

Key Point

Best Practice: Always enable MFA on the root account and for all users with console access, especially those with administrative privileges.

—

3.0.4 Data Encryption Best Practices

Encryption at Rest

Encryption at rest protects data stored on disk from unauthorized access.

Amazon S3 Encryption Options **Server-Side Encryption with S3-Managed Keys (SSE-S3):**

- AWS manages encryption keys
- AES-256 encryption
- Enabled with one click
- No additional cost
- Each object encrypted with unique key
- Best for: Simple encryption requirements

Server-Side Encryption with KMS (SSE-KMS):

- AWS KMS manages encryption keys
- You control key policies and rotation
- Audit trail via CloudTrail
- Additional cost per request
- Envelope encryption for large files
- Best for: Compliance requirements, audit needs

Server-Side Encryption with Customer-Provided Keys (SSE-C):

- You manage encryption keys outside AWS
- AWS performs encryption but doesn't store keys
- You must provide key with each request
- Best for: When you must control keys outside AWS

Client-Side Encryption:

- Encrypt data before uploading to S3
- You manage entire encryption process
- AWS stores encrypted data
- Best for: Maximum control over encryption

Configuration Example:

```
\{
  "Rules": [
    \{
      "ApplyServerSideEncryptionByDefault": \{
        "SSEAlgorithm": "aws:kms",
        "KMSMasterKeyID": "arn:aws:kms:region:account:key/key-id"
      \},
      "BucketKeyEnabled": true
    \}
  ]
\}
```

EBS Encryption Features:

- Encrypts data at rest inside volume
- Encrypts data in transit between instance and volume
- Encrypts all snapshots created from volume
- Uses AWS KMS for key management
- Minimal performance impact
- Can't encrypt root volume of existing instance (must create AMI)

How to enable:

- Enable during volume creation
- Enable account-level encryption by default
- Copy unencrypted snapshot and enable encryption
- Create encrypted AMI from unencrypted instance

RDS Encryption What gets encrypted:

- Database storage
- Automated backups
- Read replicas
- Snapshots
- Logs

Important notes:

- Must enable at database creation time
- Cannot encrypt existing unencrypted database
- Workaround: Create snapshot, copy with encryption, restore
- Same key used for instance and snapshots in same region
- Cross-region snapshots use different key

DynamoDB Encryption Features:

- Encryption at rest enabled by default
- Uses AWS owned keys (default, no cost)
- Can use AWS managed key (aws/dynamodb)
- Can use customer managed KMS key
- Encrypts tables, indexes, streams, backups

Encryption types:

- AWS owned CMK: Default, no cost, no CloudTrail logs
- AWS managed CMK: Free, CloudTrail logs available
- Customer managed CMK: You control, costs apply, full audit trail

Encryption in Transit

Encryption in transit protects data moving between systems.

TLS/SSL Best Practices Use TLS 1.2 or higher:

- TLS 1.0 and 1.1 are deprecated
- Configure minimum TLS version
- Use strong cipher suites
- Regularly update SSL/TLS certificates

AWS Certificate Manager (ACM):

- Free SSL/TLS certificates
- Automatic renewal
- Integration with CloudFront, ALB, API Gateway
- Easy deployment
- No certificate management overhead

Use cases:

- HTTPS for websites (CloudFront, ALB)
- Secure API endpoints (API Gateway)
- Database connections (RDS with SSL)
- Email encryption (SES)

VPN Encryption AWS Site-to-Site VPN:

- IPsec VPN connection
- Encrypted tunnel over internet
- Uses Internet Key Exchange (IKE)
- Supports multiple encryption algorithms
- Dead Peer Detection for availability

AWS Client VPN:

- Managed client-based VPN
- OpenVPN-based
- TLS encryption
- Integration with Active Directory
- Multi-factor authentication support

Direct Connect Encryption MACsec for Direct Connect:

- Layer 2 encryption
- Point-to-point encryption
- 10 Gbps and 100 Gbps connections
- Minimal latency impact

VPN over Direct Connect:

- IPsec VPN over DX connection
- End-to-end encryption
- Combines DX reliability with VPN security
- Industry-standard encryption

Key Management with AWS KMS

KMS Key Types Symmetric Keys (default):

- Same key for encryption and decryption
- 256-bit keys
- Never leaves KMS unencrypted
- Used for most AWS services
- Envelope encryption for large data

Asymmetric Keys:

- Public and private key pair
- RSA or Elliptic Curve keys
- Public key can be downloaded
- Private key never leaves KMS
- Used for signing and verification

Customer Master Keys (CMKs) **AWS Managed CMK:**

- Created and managed by AWS
- Used by AWS services
- Automatic rotation every year
- Cannot delete
- No cost for the key (only usage)
- Key alias: aws/service-name

Customer Managed CMK:

- You create and manage
- Full control over key policies
- Optional automatic rotation (annual)
- Can enable/disable
- Can delete (with 7-30 day waiting period)
- Cost: $1/month plus usage

AWS Owned CMK:

- AWS owns and manages
- Used across multiple accounts
- No visibility or control
- No cost
- No CloudTrail logs

KMS Key Policies Default key policy:

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Sid": "Enable IAM policies",
      "Effect": "Allow",
      "Principal": \{
        "AWS": "arn:aws:iam::123456789012:root"
      \},
      "Action": "kms:*",
      "Resource": "*"
    \}
  ]
\}
```

Custom key policy with specific permissions:

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Sid": "Allow encryption",
      "Effect": "Allow",
      "Principal": \{
        "AWS": "arn:aws:iam::123456789012:role/EncryptionRole"
      \},
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:GenerateDataKey"
      ],
      "Resource": "*"
    \},
    \{
      "Sid": "Allow key management",
      "Effect": "Allow",
      "Principal": \{
        "AWS": "arn:aws:iam::123456789012:user/KeyAdmin"
      \},
      "Action": [
        "kms:Create*",
        "kms:Describe*",
        "kms:Enable*",
        "kms:List*",
        "kms:Put*",
        "kms:Update*",
        "kms:Revoke*",
        "kms:Disable*",
        "kms:Get*",
        "kms:Delete*",
        "kms:ScheduleKeyDeletion",
```

```
        "kms:CancelKeyDeletion"
      ],
      "Resource": "*"
    \}
  ]
\}
```

Key Rotation Best Practices Automatic rotation:

- Enable for customer managed keys
- Rotates every 365 days
- Old key versions retained for decryption
- Transparent to applications
- No need to re-encrypt data

Manual rotation:

- Create new CMK
- Update applications to use new key
- Re-encrypt data with new key
- Maintain old key for decrypting old data
- More control but more complex

—

3.0.5 Network Security Deep Dive

VPC Security Architecture

Multi-Tier Architecture Example Public Subnet (DMZ):

- Internet Gateway attached
- Public IP addresses
- Bastion hosts / Jump boxes
- NAT Gateways
- Load balancers
- Route to Internet Gateway

Private Subnet (Application Tier):

- No direct internet access

- EC2 instances for applications
- Auto Scaling groups
- Route to NAT Gateway for outbound
- Access via load balancer only

Private Subnet (Database Tier):

- Most restrictive security
- RDS, DynamoDB endpoints
- No internet access (inbound or outbound)
- Access from application tier only
- Multi-AZ for high availability

Network Segmentation Best Practices Subnet Strategy:

- Separate subnets by tier (web, app, data)
- Separate subnets by environment (prod, staging, dev)
- Separate subnets by compliance requirements
- Use at least 2 AZs for high availability
- Plan IP address ranges carefully

Example CIDR allocation:

```
VPC: 10.0.0.0/16

Availability Zone A:
- Public Subnet:  10.0.1.0/24
- Private Subnet: 10.0.2.0/24
- Data Subnet:    10.0.3.0/24

Availability Zone B:
- Public Subnet:  10.0.11.0/24
- Private Subnet: 10.0.12.0/24
- Data Subnet:    10.0.13.0/24
```

VPC Endpoints for Security Interface Endpoints (PrivateLink):

- Private IP addresses in your VPC
- Elastic Network Interface (ENI)
- Supports many AWS services
- No internet gateway needed

- Charged per hour + data processed

Gateway Endpoints:

- Route table entry
- Free of charge
- Supports S3 and DynamoDB
- No ENI required
- Scalable

Benefits:

- Keep traffic within AWS network
- No internet exposure
- Better performance
- Lower data transfer costs
- Enhanced security

Example use case:

```
S3 Gateway Endpoint:
- Application accesses S3 privately
- No internet gateway required
- No NAT gateway charges
- Traffic stays on AWS network
```

Network Access Control Security Group Best Practices:
Layered security groups:

```
ALB Security Group:
- Inbound: 80, 443 from 0.0.0.0/0
- Outbound: 8080 to App-SG

App Security Group:
- Inbound: 8080 from ALB-SG
- Outbound: 3306 to DB-SG, 443 to 0.0.0.0/0

DB Security Group:
- Inbound: 3306 from App-SG
- Outbound: None
```

NACL Configuration Example:
Public Subnet NACL:

```
Inbound Rules:
100 - HTTP (80) - 0.0.0.0/0 - ALLOW
110 - HTTPS (443) - 0.0.0.0/0 - ALLOW
120 - SSH (22) - YOUR\_IP/32 - ALLOW
130 - Ephemeral (1024-65535) - 0.0.0.0/0 - ALLOW
* - ALL - 0.0.0.0/0 - DENY

Outbound Rules:
100 - HTTP (80) - 0.0.0.0/0 - ALLOW
110 - HTTPS (443) - 0.0.0.0/0 - ALLOW
120 - Ephemeral (1024-65535) - 0.0.0.0/0 - ALLOW
* - ALL - 0.0.0.0/0 - DENY
```

Private Subnet NACL:

```
Inbound Rules:
100 - Custom (8080) - 10.0.1.0/24 - ALLOW
110 - SSH (22) - 10.0.1.0/24 - ALLOW
120 - Ephemeral (1024-65535) - 0.0.0.0/0 - ALLOW
* - ALL - 0.0.0.0/0 - DENY

Outbound Rules:
100 - HTTPS (443) - 0.0.0.0/0 - ALLOW
110 - MySQL (3306) - 10.0.3.0/24 - ALLOW
120 - Ephemeral (1024-65535) - 10.0.1.0/24 - ALLOW
* - ALL - 0.0.0.0/0 - DENY
```

AWS PrivateLink What it is:

- Private connectivity to services
- No internet gateway, NAT, VPN
- Traffic stays on AWS network
- Powered by interface VPC endpoints

Use cases:

- Access SaaS applications privately
- Share services across VPCs
- Hybrid cloud connectivity
- Compliance requirements

Architecture:

```
Service Provider VPC (Your Service)
    ↓
Network Load Balancer
    ↓
```

```
VPC Endpoint Service
    ↓
Interface Endpoint (Consumer VPC)
    ↓
Consumer Application
```

VPN and Direct Connect Security Site-to-Site VPN Security:

- IPsec encryption
- Pre-shared keys or certificates
- Perfect Forward Secrecy (PFS)
- Dead Peer Detection
- IKEv2 support

VPN Configuration Best Practices:

- Use strong encryption (AES-256)
- Enable Perfect Forward Secrecy
- Configure health checks
- Use BGP for dynamic routing
- Monitor tunnel status

Direct Connect Security:

- Dedicated network connection
- Not encrypted by default
- Options for encryption:
- MACsec (Layer 2)
- VPN over DX (Layer 3)
- Application-level encryption
- Physical security at co-location
- Redundancy with multiple connections

—

3.0.6 Identity Federation and SSO

AWS IAM Identity Center (formerly AWS SSO)

What it provides:

- Single sign-on to multiple AWS accounts
- Single sign-on to business applications
- Centralized user management
- Multi-factor authentication
- Integration with external identity providers

Key Features:

- One set of credentials for all accounts
- Temporary credentials for AWS access
- Built-in MFA support
- Integration with AWS Organizations
- Permission sets for access control

Setup Process:

1. Enable IAM Identity Center
2. Connect identity source (built-in directory or external)
3. Create permission sets
4. Assign users to AWS accounts
5. Users access via SSO portal

Permission Set Example:

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Allow",
      "Action": [
        "ec2:Describe*",
        "s3:List*",
        "cloudwatch:Get*"
      ],
      "Resource": "*"
    \}
  ]
\}
```

Federation with SAML 2.0

SAML Federation Architecture:

```
User → Identity Provider (IdP) → AWS STS → Temporary Credentials → AWS Resources
```

Identity Providers:

- Microsoft Active Directory Federation Services (ADFS)
- Okta
- Azure AD
- Google Workspace
- Auth0
- OneLogin

How it works:

1. User authenticates with corporate IdP
2. IdP returns SAML assertion
3. User presents SAML assertion to AWS STS
4. STS returns temporary security credentials
5. User accesses AWS resources

Trust Relationship Policy:

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Allow",
      "Principal": \{
        "Federated": "arn:aws:iam::123456789012:saml-provider/
           MyIdP"
      \},
      "Action": "sts:AssumeRoleWithSAML",
      "Condition": \{
        "StringEquals": \{
          "SAML:aud": "https://signin.aws.amazon.com/saml"
        \}
      \}
    \}
  ]
\}
```

Web Identity Federation

For mobile and web applications:

- Users authenticate with Web IdP (Google, Facebook, Amazon)
- Application receives ID token
- Token exchanged for AWS credentials via STS
- Used with Amazon Cognito

Amazon Cognito:

- User pools for authentication
- Identity pools for AWS credentials
- Support for social identity providers
- Support for SAML providers
- Custom authentication flows

Cognito Architecture:

```
Mobile App → Cognito User Pool → Cognito Identity Pool → AWS STS → Temporary Credenti
```

Benefits:

- No AWS credentials in application
- Fine-grained access control
- Scales automatically
- Built-in security features

Active Directory Integration

AWS Directory Service Options:
AWS Managed Microsoft AD:

- Full Microsoft AD in AWS cloud
- Multi-AZ deployment
- Patch and monitoring by AWS
- Trust relationships with on-premises AD
- Best for: Lift-and-shift scenarios

AD Connector:

- Proxy to on-premises AD
- No caching, always redirects to AD

- Users authenticate against on-premises AD
- No data stored in AWS
- Best for: Using existing on-premises AD

Simple AD:

- Standalone directory powered by Samba 4
- Basic AD features
- Small and large sizes
- Cannot join to on-premises AD
- Best for: Simple LDAP needs

Cross-Account Access Strategies

Method 1: IAM Roles (Recommended):

```
Account A (Trusting) creates role
Account B (Trusted) assumes role
No credentials to manage
Temporary credentials only
```

Trust Policy Example:

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Allow",
      "Principal": \{
        "AWS": "arn:aws:iam::111122223333:root"
      \},
      "Action": "sts:AssumeRole",
      "Condition": \{
        "StringEquals": \{
          "sts:ExternalId": "UniqueSecretString"
        \}
      \}
    \}
  ]
\}
```

Method 2: Resource-based Policies:

- S3 bucket policies
- SNS topic policies
- SQS queue policies

- Lambda function policies

Best Practices:

- Always use IAM roles over shared credentials
- Use external IDs for third-party access
- Implement MFA for sensitive cross-account access
- Monitor with CloudTrail
- Use least privilege permissions

—

3.0.7 Security Incident Response Procedures

Incident Response Framework

1. Preparation Phase Before an incident occurs:
Document procedures:

- Create incident response plan
- Define severity levels
- Establish communication protocols
- Document escalation paths
- Identify team members and roles

Setup tools and access:

- Configure CloudTrail in all regions
- Enable VPC Flow Logs
- Setup GuardDuty
- Configure Security Hub
- Prepare forensics tools

Establish baselines:

- Normal traffic patterns
- Typical API usage
- Standard configurations
- Regular user behavior

Training:

- Regular tabletop exercises
- Simulate attack scenarios
- Test response procedures
- Update runbooks

2. Detection and Analysis Detection methods:

- GuardDuty findings
- CloudWatch alarms
- Security Hub alerts
- Config rule violations
- Unusual CloudTrail activity
- VPC Flow Log anomalies

Initial analysis:

- Confirm the incident is real (not false positive)
- Determine scope and severity
- Identify affected resources
- Document timeline
- Collect evidence

Severity Classification:
Critical (P1):

- Data breach confirmed
- Production systems compromised
- Ongoing active attack
- Wide-scale service disruption
- Response time: Immediate

High (P2):

- Suspected data access
- System compromise detected
- Compliance violation
- Response time: 1 hour

Medium (P3):

- Policy violations
- Suspicious activity detected
- Non-production compromise
- Response time: 4 hours

Low (P4):

- Security alerts to investigate
- Anomalous but benign activity
- Response time: 24 hours

3. Containment Strategies Short-term containment:
Isolate compromised instances:

```
\# Change security group to deny all traffic
aws ec2 modify-instance-attribute \textbackslash\{\}
  --instance-id i-1234567890abcdef0 \textbackslash\{\}
  --groups sg-isolation-group
```

Revoke compromised credentials:

```
\# Deactivate access key
aws iam update-access-key \textbackslash\{\}
  --access-key-id AKIAIOSFODNN7EXAMPLE \textbackslash\{\}
  --status Inactive \textbackslash\{\}
  --user-name CompromisedUser
```

Block malicious IP addresses:

```
\# Add NACL deny rule
aws ec2 create-network-acl-entry \textbackslash\{\}
  --network-acl-id acl-12345678 \textbackslash\{\}
  --ingress \textbackslash\{\}
  --rule-number 50 \textbackslash\{\}
  --protocol -1 \textbackslash\{\}
  --port-range From=0,To=65535 \textbackslash\{\}
  --cidr-block 198.51.100.5/32 \textbackslash\{\}
  --rule-action deny
```

Snapshot for forensics:

```
\# Create snapshot of compromised instance
aws ec2 create-snapshot \textbackslash\{\}
  --volume-id vol-1234567890abcdef0 \textbackslash\{\}
  --description "Forensic␣snapshot␣-␣Incident␣2024-001"
```

Long-term containment:

- Patch vulnerabilities
- Change all passwords
- Rotate all access keys
- Update security group rules
- Apply least privilege policies
- Enable additional monitoring

4. **Eradication Remove the threat:**
   - Delete malware
   - Close backdoors
   - Remove unauthorized access
   - Patch vulnerabilities
   - Update configurations

   Rebuild compromised systems:
   - Launch from known-good AMIs
   - Apply all security patches
   - Harden configurations
   - Implement additional controls

   Verify clean state:
   - Scan for malware
   - Review configurations
   - Check for persistence mechanisms
   - Validate logs show no malicious activity

5. **Recovery Restore operations:**
   - Restore from clean backups
   - Gradually bring systems online
   - Monitor closely for reinfection
   - Validate functionality

   Enhanced monitoring:
   - Increased logging verbosity
   - More frequent reviews
   - Additional alerting
   - Closer scrutiny of anomalies

   Communication:
   - Update stakeholders
   - Provide status reports
   - Document changes made
   - Coordinate with teams

6. Post-Incident Activity Lessons learned meeting:

- What happened?
- What was done?
- What worked well?
- What needs improvement?
- How to prevent recurrence?

Update documentation:

- Incident report
- Timeline of events
- Actions taken
- Evidence collected
- Lessons learned

Improve defenses:

- Implement preventive controls
- Update detection mechanisms
- Enhance response procedures
- Additional training
- Technology improvements

AWS Tools for Incident Response

Amazon GuardDuty:

- Automated threat detection
- ML-powered analysis
- Continuous monitoring
- Integration with EventBridge for automated response

AWS CloudTrail:

- Complete audit log of API calls
- Who did what and when
- Source IP addresses
- Request parameters

- Essential for forensics

VPC Flow Logs:

- Network traffic analysis
- Source and destination IPs
- Identify scanning attempts
- Detect data exfiltration

AWS Config:

- Configuration history
- Compliance checking
- Resource relationships
- Change tracking

Amazon Detective:

- Analyze and investigate security findings
- Visualize relationships
- Identify root cause
- Integrated with GuardDuty

AWS Systems Manager:

- Automated remediation
- Patch management
- Run commands across fleet
- Session Manager for secure access

Automated Response Examples

Lambda function for isolation:

```
import boto3

def lambda\_handler(event, context):
    ec2 = boto3.client('ec2')

    \# Extract instance ID from GuardDuty finding
    instance\_id = event['detail']['resource']['instanceDetails']
       ['instanceId']

    \# Change to isolation security group
    ec2.modify\_instance\_attribute(
```

```
        InstanceId=instance\_id,
        Groups=['sg-isolation']
    )

    \# Create forensic snapshot
    instance\_details = ec2.describe\_instances(InstanceIds=[
       instance\_id])
    volume\_id = instance\_details['Reservations'][0]['Instances'
       ][0]['BlockDeviceMappings'][0]['Ebs']['VolumeId']

    ec2.create\_snapshot(
        VolumeId=volume\_id,
        Description=f'Forensic␣snapshot␣for␣\{instance\_id\}'
    )

    \# Tag instance as compromised
    ec2.create\_tags(
        Resources=[instance\_id],
        Tags=[\{'Key': 'SecurityStatus', 'Value': 'Isolated'\}]
    )

    return \{'statusCode': 200, 'body': f'Instance␣\{instance\_id
       \}␣isolated'\}
```

EventBridge rule for GuardDuty findings:

```
\{
  "source": ["aws.guardduty"],
  "detail-type": ["GuardDuty Finding"],
  "detail": \{
    "severity": [7, 8, 9]
  \}
\}
```

Communication Plan

Notification hierarchy:

1. Security team (immediate)
2. System administrators (immediate for high severity)
3. Management (within 1 hour for critical incidents)
4. Legal/compliance (for data breaches)
5. Customers (if required by regulations)

Communication channels:

- PagerDuty / Opsgenie for alerting
- Slack / Teams for coordination

- Email for formal notifications
- Status page for customer communication

—

3.0.8 Security Services

AWS Organizations - Detailed

Centrally manage and govern multiple AWS accounts.

Key Features:

- **Centrally manage multiple AWS accounts**
- Single pane of glass for all accounts
- Organizational hierarchy
- Up to 4 levels of nesting for OUs
- **Consolidated billing across all accounts**
- One bill for entire organization
- Volume discounts apply across all accounts
- Easier cost tracking and allocation
- Shared volume pricing tiers
- **Hierarchical grouping of accounts (Organizational Units)**
- Organize by department, environment, project
- Apply policies at different levels
- Inherit policies from parent OUs
- **Service Control Policies (SCPs) for governance**
- Control maximum available permissions
- Even limits account root user
- Acts as a permission boundary
- **Automate account creation**
- Programmatic account provisioning
- Standardized setup
- Integration with AWS Control Tower
- **Centralize security and compliance**

- Enforce policies across organization
- Consistent security posture
- Delegated administration for AWS services

Consolidated Billing Benefits:

- One bill for all accounts
- Volume pricing discounts (S3, EC2, etc.)
- Reserved Instance sharing across accounts
- Savings Plans sharing
- Free tier applies once per organization
- Combined usage for tiered pricing

Service Control Policies (SCPs):

- Control maximum available permissions
- Do not grant permissions (only limit them)
- Affect all users and roles in accounts
- Do not affect service-linked roles
- Must enable before use
- Evaluation logic: explicit deny always wins

Use Case Examples:
Example 1: Multi-Environment Organization

```
Root
 Production OU
    Prod-App-Account
    Prod-Data-Account
 Development OU
    Dev-Account
    Test-Account
 Sandbox OU
     Sandbox-Account
```

SCP for Production OU (prevents accidental deletions):

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Deny",
      "Action": [
        "ec2:TerminateInstances",
```

```
        "rds:DeleteDBInstance",
        "s3:DeleteBucket"
      ],
      "Resource": "*",
      "Condition": \{
        "StringNotEquals": \{
          "aws:PrincipalArn": "arn:aws:iam::*:role/AdminRole"
        \}
      \}
    \}
  ]
\}
```

Example 2: Restricting Regions

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": \{
        "StringNotEquals": \{
          "aws:RequestedRegion": [
            "us-east-1",
            "us-west-2",
            "eu-west-1"
          ]
        \}
      \}
    \}
  ]
\}
```

AWS Key Management Service (KMS) - Detailed

Create and control cryptographic keys used to encrypt your data.

Key Features:

- **Create and manage cryptographic keys**
- Symmetric and asymmetric keys
- Hardware Security Modules (HSMs) backed
- FIPS 140-2 validated
- **Control use of keys across AWS services**
- Centralized key management
- Integration with CloudTrail

- Key policies for fine-grained control
- **Integrated with most AWS services**
- S3, EBS, RDS, DynamoDB, and more
- Transparent encryption
- Over 100 AWS services integrated
- **Customer Master Keys (CMKs)**
- **AWS Managed CMKs:** Created and managed by AWS, free
- **Customer Managed CMKs:** You create and manage, $1/month
- **AWS Owned CMKs:** Used by AWS services, no visibility
- **Automatic key rotation available**
- Annual rotation for customer managed keys (optional)
- Automatic for AWS managed keys (mandatory)
- Old key material retained for decryption
- **Audit key usage via CloudTrail**
- Who used which key
- When and for what purpose
- Complete audit trail

Use Case Examples:
Use Case 1: Encrypt S3 Bucket with Customer Managed Key

```
Scenario: Healthcare company storing patient records
Requirement: Control encryption keys, audit access, rotate annually
Solution: Create customer managed KMS key with strict key policy

Benefits:
- Full control over key lifecycle
- Audit trail in CloudTrail
- Can disable key if needed
- Automatic rotation
```

Use Case 2: Cross-Account Data Sharing

```
Scenario: Share encrypted data between AWS accounts
Setup:
1. Create KMS key in Account A
2. Update key policy to allow Account B
3. Share encrypted S3 objects
4. Account B can decrypt with permission
```

```
Key Policy Addition:
\{
  "Effect": "Allow",
  "Principal": \{
    "AWS": "arn:aws:iam::222222222222:root"
  \},
  "Action": [
    "kms:Decrypt",
    "kms:DescribeKey"
  ],
  "Resource": "*"
\}
```

Use Case 3: Envelope Encryption

```
Large file encryption process:
1. KMS generates data encryption key (DEK)
2. DEK encrypts the actual data
3. KMS encrypts the DEK with CMK
4. Store encrypted data + encrypted DEK
5. To decrypt: KMS decrypts DEK, DEK decrypts data

Benefits:
- Better performance for large files
- Reduced KMS API calls
- Data doesn't pass through KMS
```

Pricing:

- Customer managed CMK: $1/month per key
- API requests: $0.03 per 10,000 requests
- Free tier: 20,000 requests/month
- AWS managed CMKs: No charge for the key

AWS Shield

DDoS (Distributed Denial of Service) protection service.

AWS Shield Standard

- **Automatic protection** for all AWS customers
- **No additional cost**
- Protects against **common Layer 3/4 attacks**
- SYN/ACK floods
- Reflection attacks

- UDP floods
- Always-on detection
- Automatic inline mitigations

AWS Shield Advanced

- **$3,000/month** per organization
- **Enhanced protection** for:
- Amazon EC2
- Elastic Load Balancing (ELB)
- Amazon CloudFront
- Amazon Route 53
- AWS Global Accelerator
- **24/7 access to DDoS Response Team (DRT)**
- Expert support during attacks
- Attack diagnostics
- **Cost protection**
- Protection against usage spikes during attacks
- Cost reimbursement for scaled resources
- **Real-time attack notifications**
- CloudWatch metrics
- Health-based detection

Amazon GuardDuty - Detailed

Intelligent threat detection service using machine learning.

Key Features:

- **Intelligent threat detection service**
- Continuous monitoring (24/7)
- ML-powered analysis
- Threat intelligence feeds
- **Uses machine learning**
- Anomaly detection

- Known threat patterns
- Behavioral analysis
- **Monitors multiple data sources**
- VPC Flow Logs (network traffic)
- CloudTrail event logs (API activity)
- DNS logs (DNS queries)
- Kubernetes audit logs (EKS protection)
- S3 data events (S3 Protection)
- RDS login activity (RDS Protection)
- EBS volume snapshots (Malware Protection)
- **Identifies unauthorized or malicious activity**
- Compromised instances
- Reconnaissance attempts
- Account compromise
- Cryptocurrency mining
- Data exfiltration attempts
- **No software to deploy**
- Fully managed service
- Enable with a few clicks
- No impact on performance
- **30-day free trial**
- **Integrates with EventBridge**
- Automated responses to findings
- Lambda function triggers
- SNS notifications

Common Threat Findings:

Finding Type	Description	Example
Backdoor:EC2/...	Backdoor on EC2 instance	C&C server communication
Behavior:EC2/...	Unusual instance behavior	Traffic to unusual port

CryptoCurrency:EC2/...	Cryptocurrency mining	Bitcoin mining detected
Trojan:EC2/...	Trojan detected	DNS query to known bad domain
UnauthorizedAccess:EC2/...	Unauthorized access attempt	SSH brute force attack
Recon:IAMUser/...	Reconnaissance by IAM user	Listing resources unusually
Stealth:IAMUser/...	Stealth techniques	CloudTrail logging disabled
CredentialAccess:IAMUser/...	Credential access attempts	Password policies weakened

Use Case Examples:

Use Case 1: Detecting Compromised Instance

```
Scenario: EC2 instance starts communicating with known C\&C server
GuardDuty Detection:
- Finding: Backdoor:EC2/C\&CActivity.B
- Severity: High
- Details: Instance communicating with command and control server

Automated Response:
1. EventBridge rule triggers Lambda
2. Lambda isolates instance (change security group)
3. Lambda creates snapshot for forensics
4. SNS notification to security team
5. Ticket created in ticketing system
```

Use Case 2: Unusual API Call Pattern

```
Scenario: IAM user making unusual API calls
GuardDuty Detection:
- Finding: Recon:IAMUser/NetworkPermissions
- Severity: Medium
- Details: User listing network resources unusually

Response:
1. Alert security team
2. Review CloudTrail logs
3. Interview user about activity
4. If compromised: rotate credentials
```

Use Case 3: Cryptocurrency Mining

```
Scenario: EC2 instance performing DNS queries to mining pools
GuardDuty Detection:
- Finding: CryptoCurrency:EC2/BitcoinTool.B
- Severity: High
- Details: DNS queries to Bitcoin mining pools

Response:
1. Immediately isolate instance
2. Snapshot for investigation
3. Terminate compromised instance
4. Launch replacement from clean AMI
5. Investigate how compromise occurred
```

Pricing:

- Based on volume of data analyzed
- CloudTrail events: $4.00 per million events
- VPC Flow Logs: $1.00 per GB
- DNS logs: $0.40 per million events
- First 30 days free
- No upfront commitment

Amazon Inspector

Automated security assessment service for applications.

Key Features:

- **Automated security assessment service**
- Continuous scanning
- Scheduled assessments
- **Assesses applications for vulnerabilities**
- CVE vulnerabilities
- Network exposure
- **Checks for:**
- Exposure to external threats
- Vulnerabilities in applications
- Deviations from best practices
- **Generates detailed security findings**
- Severity ratings
- Remediation recommendations
- **Prioritized list of security findings**
- Risk-based prioritization
- Context-aware scoring
- **Supports:**
- EC2 instances
- Container images (ECR)

- Lambda functions

Assessment Types:

- Network assessments
- Host assessments
- Package vulnerability scanning

AWS WAF (Web Application Firewall)

Protects web applications from common web exploits.

Key Features:

- **Protects web applications** from common exploits
- **Deployed on:**
- Amazon CloudFront
- Application Load Balancer (ALB)
- Amazon API Gateway
- AWS AppSync
- **Create custom rules** to block attack patterns
- Define conditions
- Action on matches (Allow, Block, Count)
- **Protection against:**
- SQL injection
- Cross-site scripting (XSS)
- Size constraints violations
- Geo-blocking
- **IP-based filtering**
- IP sets (allow/deny lists)
- IP rate limiting
- **Geo-blocking capabilities**
- Block traffic from specific countries
- **Rate-based rules**
- Prevent DDoS

- Limit requests per IP

Web ACL (Access Control List):

- Collection of rules
- Applies to CloudFront distribution or ALB
- Rules evaluated in order

Amazon Macie

Data security and privacy service using machine learning.

Key Features:

- **Data security and privacy service**
- Sensitive data discovery
- Data protection
- **Uses machine learning**
- Intelligent pattern matching
- Anomaly detection
- **Discovers and protects sensitive data**
- Personally Identifiable Information (PII)
- Financial data
- Credentials
- **Identifies PII**
- Names, addresses
- Credit card numbers
- Social Security numbers
- Passport numbers
- **Monitors S3 buckets**
- Data inventory
- Security findings
- Bucket policies
- **Provides dashboards and alerts**
- Security findings

- Data classification
- **Helps meet compliance requirements**
- GDPR
- HIPAA
- PCI DSS

Use Cases:

- Discover sensitive data in S3
- Monitor for suspicious access patterns
- Compliance auditing
- Data classification

AWS Artifact

Self-service portal for on-demand access to AWS compliance reports.

Key Features:

- **On-demand access** to AWS compliance reports
- **Self-service portal** for audit artifacts
- **Download AWS security and compliance documents**
- Instant access
- No waiting for support
- **Examples of available reports:**
- ISO certifications (27001, 27017, 27018)
- SOC reports (SOC 1, 2, 3)
- PCI DSS reports
- FedRAMP documentation
- **No cost**
- Free to use
- Available to all AWS customers
- **Support compliance and regulatory requirements**
- Audit evidence
- Third-party attestations

Two Main Sections:

1. **Artifact Reports:** Compliance reports and certifications
2. **Artifact Agreements:** Review and accept agreements (BAA, GDPR DPA)

—

3.0.9 Compliance

AWS Compliance Programs

AWS complies with numerous industry-specific compliance programs and regulations:

Program	Description	Industry
HIPAA	Health Insurance Portability and Accountability Act	Healthcare
PCI DSS	Payment Card Industry Data Security Standard	Payment Processing
SOC 1, 2, 3	Service Organization Controls	Various
ISO 27001	Information Security Management	Various
FedRAMP	Federal Risk and Authorization Management Program	US Government
GDPR	General Data Protection Regulation	EU Data Privacy

HIPAA (Health Insurance Portability and Accountability Act)

- **Healthcare industry** compliance
- Protects **Protected Health Information (PHI)**
- Requires **Business Associate Agreement (BAA)** with AWS
- HIPAA-eligible services include:
- S3, EC2, RDS, DynamoDB
- And many others (check AWS documentation)

PCI DSS (Payment Card Industry Data Security Standard)

- **Payment card processing** compliance
- Protects **cardholder data**
- Multiple compliance levels
- AWS infrastructure is PCI DSS compliant
- Customer applications may need separate certification

SOC (Service Organization Controls)

- **SOC 1:** Financial reporting controls
- **SOC 2:** Security, availability, confidentiality controls
- Type I: Design of controls
- Type II: Operating effectiveness
- **SOC 3:** General use report (public)

ISO 27001

- **International standard** for information security
- Information Security Management System (ISMS)
- Risk management framework
- Demonstrates security commitment

FedRAMP (Federal Risk and Authorization Management Program)

- **US Government** cloud compliance
- Standardized approach to security assessment
- Authorization levels:
- Low Impact
- Moderate Impact
- High Impact

GDPR (General Data Protection Regulation)

- **EU data privacy** regulation
- Applies to processing of EU residents' data
- Key requirements:
- Data protection by design
- Right to erasure
- Data portability
- Breach notification
- AWS provides GDPR-compliant services and features

Exam Tip

You don't need to memorize all compliance programs in detail, but know what they stand for and which industries they apply to.

—

3.0.10 Compliance Programs - Deep Dive

HIPAA Compliance Details

What is HIPAA?

- US legislation protecting patient medical records and PHI
- Enacted in 1996
- Applies to covered entities and business associates
- Requires safeguards for PHI confidentiality, integrity, availability

AWS and HIPAA:

- AWS infrastructure is HIPAA-compliant
- Must sign Business Associate Agreement (BAA) with AWS
- BAA is free, request through AWS Artifact
- Only HIPAA-eligible services can store PHI

HIPAA-Eligible Services (common ones):

- Compute: EC2, Lambda, Elastic Beanstalk
- Storage: S3, EBS, EFS, Glacier
- Database: RDS, DynamoDB, Redshift
- Networking: VPC, Direct Connect, Route 53
- Analytics: EMR, Kinesis, Athena

Customer Responsibilities:

- Execute BAA before processing PHI
- Use only HIPAA-eligible services for PHI
- Implement proper access controls
- Encrypt PHI at rest and in transit
- Maintain audit logs
- Implement breach notification procedures
- Regular risk assessments

Technical Safeguards Required:

- Access controls (IAM, MFA)
- Audit controls (CloudTrail, Config)
- Integrity controls (checksums, versioning)
- Transmission security (TLS, VPN)
- Encryption (KMS, SSL/TLS)

PCI DSS Compliance Details

What is PCI DSS?

- Payment Card Industry Data Security Standard
- Protects cardholder data
- Applies to merchants and service providers
- 12 requirements across 6 control objectives

Six Control Objectives:

1. Build and maintain secure network
2. Protect cardholder data
3. Maintain vulnerability management program
4. Implement strong access control measures
5. Regularly monitor and test networks
6. Maintain information security policy

AWS PCI DSS Compliance:

- AWS infrastructure: PCI DSS Level 1 compliant
- Highest level of compliance
- Applies to compute, storage, network services
- Customer applications may need separate validation

Compliance Levels:

- **Level 1:** 6+ million transactions/year
- **Level 2:** 1-6 million transactions/year
- **Level 3:** 20,000-1 million e-commerce transactions/year
- **Level 4:** ¡20,000 e-commerce transactions/year

AWS Services for PCI DSS:

- Cardholder Data Environment (CDE) can run on EC2
- Segment CDE in separate VPC or subnet
- Use encryption for data at rest (KMS)
- Use TLS for data in transit
- Implement logging (CloudTrail, VPC Flow Logs)

- Use AWS WAF for application protection

Key Requirements:

- Network segmentation (VPC, security groups)
- Access controls (IAM, MFA)
- Encryption (KMS, TLS)
- Logging and monitoring (CloudTrail, CloudWatch)
- Vulnerability management (Inspector)
- Penetration testing (with AWS permission)

SOC Reports Details

SOC 1 (SSAE 18):

- Focus: Financial reporting controls
- Audience: Financial auditors
- Content: Controls relevant to financial statements
- AWS provides: SOC 1 Type II report

SOC 2 (AT-C 105):

- Focus: Security, availability, processing integrity, confidentiality, privacy
- Audience: Management, regulators, stakeholders
- Two types:
- **Type I:** Design of controls at specific point in time
- **Type II:** Operating effectiveness over period (usually 6-12 months)
- AWS provides: SOC 2 Type II report

SOC 3:

- Simplified version of SOC 2
- General use report
- Publicly available
- Does not include detailed testing results
- Good for marketing and general assurance

Five Trust Service Principles:

1. **Security:** Protection against unauthorized access

2. **Availability:** System accessibility as agreed
3. **Processing Integrity:** Complete, valid, accurate processing
4. **Confidentiality:** Confidential information protection
5. **Privacy:** Personal information protection per commitments

How to Access:

- AWS Artifact for SOC 1, 2, 3 reports
- No cost
- Requires AWS account
- NDA acceptance required

ISO 27001 Details

What is ISO 27001?

- International standard for ISMS
- Published by ISO/IEC
- Specifies requirements for establishing, implementing, maintaining ISMS
- Risk-based approach

Key Components:

- 14 control domains
- 114 controls
- Continuous improvement cycle (Plan-Do-Check-Act)

14 Control Domains:

1. Information security policies
2. Organization of information security
3. Human resource security
4. Asset management
5. Access control
6. Cryptography
7. Physical and environmental security
8. Operations security
9. Communications security

10. System acquisition, development, maintenance
11. Supplier relationships
12. Incident management
13. Business continuity
14. Compliance

AWS ISO Certifications:

- ISO 27001 (Information Security Management)
- ISO 27017 (Cloud Security)
- ISO 27018 (Cloud Privacy)
- ISO 27701 (Privacy Information Management)
- ISO 9001 (Quality Management)
- ISO 22301 (Business Continuity)

Access Reports:

- Download from AWS Artifact
- Available to all AWS customers
- Updated annually

FedRAMP Details

What is FedRAMP?

- Federal Risk and Authorization Management Program
- US Government cloud security standard
- Standardizes security assessment and authorization
- Mandatory for federal agencies

Authorization Levels:
Low Impact:

- Data loss: Limited impact
- Examples: Static websites, public information
- Controls: 125 security controls

Moderate Impact:

- Data loss: Serious impact

- Examples: Most federal applications
- Controls: 325 security controls
- Most common baseline

High Impact:

- Data loss: Severe/catastrophic impact
- Examples: National security systems
- Controls: 421 security controls
- Highest security requirements

AWS FedRAMP Compliance:

- FedRAMP Authorized at High impact level
- Covers AWS GovCloud (US) regions
- Covers select services in commercial regions
- Continuous monitoring required

FedRAMP Authorization Process:

1. Preparation (package development)
2. Assessment by 3PAO (Third Party Assessment Organization)
3. Authorization by JAB or Agency
4. Continuous monitoring

AWS Services FedRAMP Authorized:

- 100+ services authorized
- Check FedRAMP Marketplace for current list
- New services regularly added

GDPR Details

What is GDPR?

- General Data Protection Regulation
- EU regulation effective May 2018
- Applies to processing of EU residents' data
- Extraterritorial scope (applies globally)
- Heavy fines for non-compliance (up to 4% of revenue or €20M)

Key Principles:

1. **Lawfulness, fairness, transparency**
2. **Purpose limitation**
3. **Data minimization**
4. **Accuracy**
5. **Storage limitation**
6. **Integrity and confidentiality**
7. **Accountability**

Data Subject Rights:

- Right to access
- Right to rectification
- Right to erasure ("right to be forgotten")
- Right to restrict processing
- Right to data portability
- Right to object
- Rights related to automated decision-making

AWS GDPR Compliance:

- AWS Data Processing Addendum (DPA) available
- Supports customer GDPR compliance
- Data residency options (choose regions)
- Encryption capabilities
- Access controls and logging
- Data portability features

Technical Measures for GDPR:

- **Encryption:** KMS, SSL/TLS for data protection
- **Access Control:** IAM for limiting data access
- **Logging:** CloudTrail for accountability
- **Data Residency:** Region selection for data location
- **Deletion:** S3 lifecycle policies for right to erasure

- **Portability:** Data export capabilities
- **Anonymization:** Services for de-identification

Breach Notification:

- Must notify supervisory authority within 72 hours
- Must notify affected individuals without undue delay
- AWS notifies customers of breaches affecting them
- Customer responsible for notifying authorities/individuals

AWS Tools for GDPR:

- IAM for access control
- KMS for encryption
- CloudTrail for audit logs
- Config for compliance monitoring
- Macie for PII discovery
- S3 versioning and lifecycle for data retention

—

3.0.11 Common Security Mistakes and How to Avoid Them

Mistake 1: Using Root Account for Daily Tasks

Why it's dangerous:

- Root account has unrestricted access
- Cannot limit permissions
- If compromised, entire account at risk
- Difficult to track who did what

How to avoid:

- Create IAM users for daily tasks
- Use root account only for initial setup
- Enable MFA on root account
- Never create access keys for root account
- Lock away root account credentials
- Set up billing alerts on root account

Best practice:

```
1. Create IAM admin user immediately after account creation
2. Enable MFA on root account
3. Store root credentials in secure location (password manager)
4. Use IAM admin user for all tasks
5. Monitor root account usage with CloudWatch alarm
```

Mistake 2: Overly Permissive IAM Policies

Common patterns:

```
\{
  "Effect": "Allow",
  "Action": "*",
  "Resource": "*"
\}
```

Why it's dangerous:

- Grants unlimited access
- Violates least privilege
- Increases blast radius of compromise
- Hard to audit what's actually used

How to avoid:

- Start with minimal permissions
- Add permissions as needed
- Use AWS managed policies as starting point
- Regularly review and remove unused permissions
- Use IAM Access Analyzer
- Implement permission boundaries

Better approach:

```
\{
  "Effect": "Allow",
  "Action": [
    "s3:GetObject",
    "s3:PutObject"
  ],
  "Resource": "arn:aws:s3:::specific-bucket/*"
\}
```

Mistake 3: Hardcoding Credentials in Code

Examples of what NOT to do:

```
\# NEVER DO THIS
aws\_access\_key = "AKIAIOSFODNN7EXAMPLE"
aws\_secret\_key = "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
```

Why it's dangerous:

- Credentials exposed in version control
- Difficult to rotate
- Can be discovered by attackers
- Violates security best practices

How to avoid:

- Use IAM roles for EC2 instances
- Use environment variables
- Use AWS Secrets Manager
- Use Systems Manager Parameter Store
- Use temporary credentials via STS

Better approach:

```
\# Use IAM role (credentials automatically provided)
import boto3
s3 = boto3.client('s3')  \# Credentials from instance role

\# Or use Secrets Manager
import json
secretsmanager = boto3.client('secretsmanager')
secret = secretsmanager.get\_secret\_value(SecretId='MySecret')
credentials = json.loads(secret['SecretString'])
```

Mistake 4: Leaving S3 Buckets Publicly Accessible

Why it's dangerous:

- Data exposed to internet
- Source of many data breaches
- Compliance violations
- Potential for data loss or ransomware

How to avoid:

- Enable S3 Block Public Access (account-level)
- Use bucket policies to restrict access
- Enable S3 server access logging
- Use AWS Macie to find sensitive data
- Regular audits with AWS Config
- Use VPC endpoints for private access

Configuration:

```
Enable S3 Block Public Access Settings:
 Block public access to buckets through new ACLs
 Block public access to buckets through any ACLs
 Block public access to buckets through new public bucket policies
 Block public and cross-account access through any public bucket policies
```

Mistake 5: Not Enabling MFA

Why it's dangerous:

- Password-only authentication is weak
- Vulnerable to phishing
- Credential stuffing attacks
- Account takeover

How to avoid:

- Enable MFA on root account (mandatory)
- Enable MFA for all IAM users
- Require MFA for sensitive operations
- Use hardware MFA for high-privilege users
- Enforce MFA with IAM policies

MFA enforcement policy:

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": \{
        "BoolIfExists": \{
          "aws:MultiFactorAuthPresent": "false"
```

```
          \}
        \}
      \}
  ]
\}
```

Mistake 6: Ignoring CloudTrail Logs

Why it's dangerous:

- No audit trail
- Can't investigate incidents
- Compliance violations
- Unable to detect unauthorized access

How to avoid:

- Enable CloudTrail in all regions
- Send logs to S3 bucket
- Enable log file validation
- Set up CloudWatch Logs integration
- Create alarms for suspicious activity
- Restrict access to CloudTrail logs
- Enable in separate security account

Critical events to monitor:

- Root account usage
- IAM policy changes
- Security group changes
- CloudTrail being disabled
- Unauthorized API calls
- Failed login attempts

Mistake 7: Poor Security Group Configuration

Common mistakes:

- Opening 0.0.0.0/0 on all ports
- Allowing RDP/SSH from anywhere
- Overly permissive outbound rules
- Not using security group references

Why it's dangerous:

- Exposes resources to internet
- Increases attack surface
- Brute force attacks
- Lateral movement if compromised

How to avoid:

- Use principle of least privilege
- Restrict SSH/RDP to specific IPs
- Use security group references
- Regular audits
- Use AWS Config rules
- Implement bastion hosts

Bad configuration:

```
Inbound: 0.0.0.0/0 on port 22 (SSH)
```

Good configuration:

```
Inbound: YOUR\_IP/32 on port 22 (SSH)
Or better: Bastion-SG on port 22
```

Mistake 8: Not Encrypting Data

Why it's dangerous:

- Data exposed if storage compromised
- Compliance violations
- Data breaches
- Regulatory fines

How to avoid:

- Enable encryption by default
- Use KMS for key management
- Encrypt data in transit (TLS/SSL)
- Encrypt data at rest
- Use S3 bucket encryption
- Enable EBS encryption by default
- Use RDS encryption

Enable encryption by default:

```
Account Settings:
 EBS encryption enabled by default
 S3 default encryption enabled
 RDS encryption required

 TLS 1.2+ enforced
 HTTPS required for CloudFront
```

Mistake 9: Sharing IAM Credentials

Examples:

- Multiple people using same IAM user
- Sharing access keys
- Using one "service account" for everything

Why it's dangerous:

- No accountability
- Can't track who did what
- Difficult to rotate
- Violates compliance requirements

How to avoid:

- Create individual IAM users
- Use IAM roles for services
- Implement federation for user access
- No shared credentials ever
- Use temporary credentials
- Monitor and alert on concurrent logins

Mistake 10: Neglecting Security Updates

What's neglected:

- OS patches
- Application updates
- Security patches
- AMI updates

Why it's dangerous:

- Known vulnerabilities exploited
- Malware infections
- Compliance violations
- Security breaches

How to avoid:

- Use AWS Systems Manager Patch Manager
- Enable automatic security updates
- Regularly update AMIs
- Use Amazon Inspector
- Implement patch compliance monitoring
- Schedule regular maintenance windows

Patch management strategy:

```
1. Test patches in dev environment
2. Schedule maintenance windows
3. Use Systems Manager for patching
4. Monitor patch compliance with Config
5. Automate where possible
6. Maintain patch documentation
```

Mistake 11: Not Using Least Privilege

Common patterns:

- Giving admin access to everyone
- Using wildcard (*) in policies
- Not reviewing permissions
- Adding permissions but never removing

How to avoid:

- Start with zero permissions
- Add only what's needed
- Regular access reviews
- Use IAM Access Analyzer
- Remove unused permissions
- Use permission boundaries

Mistake 12: Poor Network Segmentation

Mistakes:

- All resources in public subnet
- No separation between tiers
- Flat network architecture

How to avoid:

- Use multiple subnets
- Separate by tier (web, app, data)
- Use private subnets for databases
- Implement defense in depth
- Use NACLs and security groups
- Follow well-architected principles

Proper architecture:

```
Public Subnet: Load balancers, bastion hosts
Private Subnet: Application servers
Private Subnet: Databases (no internet access)
```

—

3.0.12 Security Checklist for Exam Preparation

IAM Security Checklist

Root account has MFA enabled

Root account has no access keys

Individual IAM users created (no sharing)

IAM users have MFA enabled

IAM password policy is strong

IAM users grouped by role

Policies attached to groups, not users

Least privilege principle applied

Unused credentials removed

Access keys rotated every 90 days

IAM roles used for EC2 instances

Cross-account access uses roles

Service Control Policies implemented (Organizations)

Permission boundaries used where appropriate

Data Protection Checklist

S3 buckets have encryption enabled

S3 Block Public Access enabled

S3 versioning enabled for important data

EBS encryption enabled by default

RDS databases encrypted

Data encrypted in transit (TLS/SSL)

KMS used for key management

Automatic key rotation enabled

Sensitive data classified

DLP policies implemented (Macie)

Backup strategy defined

Backup testing performed regularly

Network Security Checklist

- VPC created for resources
- Public/private subnets separated
- Security groups follow least privilege
- NACLs configured for subnet protection
- VPC Flow Logs enabled
- No 0.0.0.0/0 on SSH/RDP
- Bastion hosts used for access
- VPC endpoints used for AWS services
- Network segmentation implemented
- WAF enabled for web applications
- Shield Standard active (automatic)
- DDoS response plan documented

Monitoring and Logging Checklist

- CloudTrail enabled in all regions
- CloudTrail log file validation enabled
- CloudTrail logs in separate account
- VPC Flow Logs enabled
- S3 access logging enabled
- ELB access logs enabled
- CloudWatch alarms configured
- GuardDuty enabled
- Security Hub enabled
- Config rules enabled
- Automated remediation configured
- Incident response plan documented

Compliance Checklist

Compliance requirements identified

AWS Artifact reports reviewed

BAA signed (if HIPAA required)

Compliance documentation maintained

Regular compliance audits performed

Config rules for compliance checking

Tags applied for governance

Resource inventory maintained

Exam Readiness Checklist

Understand Shared Responsibility Model

Know IAM components (users, groups, roles, policies)

Understand difference between authentication and authorization

Know when to use each security service

Understand compliance programs and industries

Know security best practices

Understand encryption (at rest and in transit)

Know network security concepts

Understand monitoring and logging services

Know incident response basics

AWS Config

Assess, audit, and evaluate AWS resource configurations.

Key Features:

- **Assess, audit, and evaluate configurations**
- Configuration history
- Configuration snapshots
- **Continuous monitoring** of resource configurations
- Real-time tracking
- Change detection

- **Track configuration changes over time**
- Who made changes
- When changes occurred
- What changed
- **Compliance auditing and security analysis**
- Configuration compliance
- Security posture assessment
- **Config Rules** define desired configurations
- AWS managed rules
- Custom rules (Lambda)
- Automatic or triggered evaluation
- **Automated remediation** of non-compliant resources
- SSM Automation documents
- Automatic or manual remediation

Use Cases:

- Continuous compliance monitoring
- Security analysis
- Change management
- Troubleshooting
- Configuration history

How It Works:

1. Enable AWS Config in your account
2. Select resources to monitor
3. Define Config Rules
4. Review compliance dashboard
5. Set up automated remediation (optional)

Integration:

- CloudTrail (who made the change)
- SNS (notifications)
- S3 (configuration snapshots)
- Systems Manager (remediation)

—

3.0.13 Review Questions

Test your knowledge of Domain 2: Security and Compliance.

Question 1

According to the Shared Responsibility Model, which security aspect is AWS responsible for?

A. Security group configuration B. Physical security of data centers C. Customer data encryption D. IAM user management

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: AWS is responsible for security OF the cloud, which includes physical security of data centers, hardware, and infrastructure. The customer is responsible for security IN the cloud, including security groups (A), data encryption (C), and IAM user management (D).

¡/details¿

—

Question 2

Which service provides DDoS protection at no additional cost?

A. AWS WAF B. AWS Shield Advanced C. AWS Shield Standard D. Amazon GuardDuty

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C

Explanation: AWS Shield Standard provides automatic DDoS protection for all AWS customers at no additional cost. Shield Advanced (B) costs $3,000/month, WAF (A) has its own pricing, and GuardDuty (D) is for threat detection, not DDoS protection.

¡/details¿

—

Question 3

What is the best practice for granting permissions to a group of developers?

A. Attach policies directly to each user B. Create an IAM group, attach policies to the group, add users to the group C. Share the root account credentials D. Create one IAM user that everyone shares

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: The best practice is to create IAM groups, attach policies to the groups, and then add users to appropriate groups. This simplifies management and follows security best practices. Sharing credentials (C and D) is never recommended, and attaching policies to individual users (A) is harder to manage.

¡/details¿

—

Question 4

Which service uses machine learning to discover and protect sensitive data in S3?

A. Amazon GuardDuty B. Amazon Inspector C. Amazon Macie D. AWS Config

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C

Explanation: Amazon Macie uses machine learning to discover, classify, and protect sensitive data (like PII) in Amazon S3. GuardDuty (A) is for threat detection, Inspector (B) is for vulnerability assessment, and Config (D) is for configuration compliance.

¡/details¿

—

Question 5

Which IAM entity provides temporary security credentials?

A. IAM User B. IAM Group C. IAM Role D. IAM Policy

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C

Explanation: IAM Roles provide temporary security credentials that are automatically rotated. Users (A) have long-term credentials, Groups (B) are collections of users, and Policies (D) define permissions but don't provide credentials.

¡/details¿

—

Question 6

Where can you download AWS compliance reports and certifications?

A. AWS Config B. AWS Artifact C. AWS Inspector D. AWS Organizations

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: AWS Artifact is the self-service portal where you can download AWS compliance reports, certifications (ISO, SOC, PCI), and agreements. It's available at no cost to all AWS customers.

¡/details¿

—

Question 7

What is the primary purpose of AWS Config?

A. Encrypt data at rest B. Track configuration changes and compliance C. Detect threats using machine learning D. Protect against DDoS attacks

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: AWS Config tracks configuration changes over time and evaluates compliance against desired configurations. KMS handles encryption (A), GuardDuty detects threats (C), and Shield protects against DDoS (D).

¡/details¿

—

Question 8

Which authentication factor does MFA add to username/password?

A. Something you know B. Something you have C. Something you are D. Somewhere you are

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: MFA adds "something you have" (the MFA device) to "something you know" (the password), providing two-factor authentication. The password is "something you know" (A), biometrics would be "something you are" (C), and location would be "somewhere you are" (D).

¡/details¿

—

Question 9

Which service would you use to centrally manage multiple AWS accounts and apply governance policies?

A. IAM B. AWS Organizations C. AWS Config D. AWS Control Tower

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: AWS Organizations allows you to centrally manage multiple AWS accounts, provide consolidated billing, and apply Service Control Policies (SCPs) for governance. IAM (A) manages access within a single account, Config (C) tracks configurations, and while Control Tower (D) can also manage accounts, Organizations is the core service tested at the Cloud Practitioner level.

¡/details¿

—

Question 10

Which compliance program is specifically for healthcare data in the United States?

A. PCI DSS B. GDPR C. HIPAA D. SOC 2

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C

Explanation: HIPAA (Health Insurance Portability and Accountability Act) is the US regulation for protecting healthcare data and PHI (Protected Health Information). PCI DSS (A) is for payment cards, GDPR (B) is EU data privacy, and SOC 2 (D) is for general security controls.

¡/details¿

—

Question 11

Which AWS service should you use to discover and protect sensitive data like credit card numbers in S3?

A. AWS Config B. Amazon Macie C. AWS WAF D. Amazon Inspector

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: Amazon Macie uses machine learning to discover, classify, and protect sensitive data like PII, credit card numbers, and other confidential information in S3 buckets. Config (A) tracks configurations, WAF (C) protects web applications, and Inspector (D) assesses vulnerabilities.

¡/details¿

—

Question 12

Your company needs to encrypt data at rest in S3 with full control over the encryption keys, including rotation. Which solution should you use?

A. SSE-S3 (Server-Side Encryption with S3-Managed Keys) B. SSE-KMS with customer managed CMK C. SSE-C (Server-Side Encryption with Customer-Provided Keys) D. Client-side encryption

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: SSE-KMS with customer managed CMK gives you full control over encryption keys, including rotation, while AWS handles the encryption process. SSE-S3 (A) doesn't give you control over keys, SSE-C (C) requires you to provide keys with each request, and client-side encryption (D) requires you to manage the entire encryption process.

¡/details¿

—

Question 13

Which service provides automated vulnerability assessment for EC2 instances and container images?

A. Amazon GuardDuty B. AWS Security Hub C. Amazon Inspector D. AWS Systems Manager

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C

Explanation: Amazon Inspector is an automated security assessment service that checks for vulnerabilities in EC2 instances, container images in ECR, and Lambda functions. GuardDuty (A) is for threat detection, Security Hub (B) is a centralized security view, and Systems Manager (D) is for operational management.

¡/details¿

—

Question 14

According to the Shared Responsibility Model, who is responsible for patching the guest operating system on an EC2 instance?

A. AWS B. Customer C. Both AWS and Customer D. Neither, it's automated

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: The customer is responsible for patching the guest OS on EC2 instances. This falls under "security IN the cloud." AWS is responsible for patching the

hypervisor and infrastructure (security OF the cloud). For managed services like RDS, AWS handles the patching.

¡/details¿

—

Question 15

Which feature of AWS Organizations allows you to restrict actions across all accounts in your organization?

A. IAM Policies B. Resource Access Manager C. Service Control Policies (SCPs) D. Permission Boundaries

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C

Explanation: Service Control Policies (SCPs) allow you to set maximum available permissions across accounts in AWS Organizations. They act as guardrails and can even restrict the root user. IAM policies (A) work within a single account, RAM (B) is for resource sharing, and permission boundaries (D) set maximum permissions for IAM entities.

¡/details¿

—

Question 16

What is the primary purpose of AWS CloudTrail?

A. Monitor resource utilization B. Log API activity for auditing C. Detect security threats D. Track configuration changes

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: CloudTrail logs API activity in your AWS account, providing an audit trail of who did what, when, and from where. CloudWatch (A) monitors resource utilization, GuardDuty (C) detects threats, and Config (D) tracks configuration changes.

¡/details¿

—

Question 17

Which of the following is NOT a valid MFA device option for AWS?

A. Virtual MFA device (smartphone app) B. Hardware MFA device (YubiKey) C. SMS text message D. Fingerprint scanner

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: D

Explanation: AWS does not support fingerprint scanners or other biometric authentication for MFA. Valid options include virtual MFA devices (A), hardware MFA devices (B), and SMS text messages (C), although SMS is not recommended for root accounts.

¡/details¿

—

Question 18

A startup is building a web application that needs to authenticate users via Facebook and Google. Which AWS service should they use?

A. AWS IAM B. Amazon Cognito C. AWS Directory Service D. AWS Single Sign-On

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: Amazon Cognito supports web identity federation, allowing users to authenticate with social identity providers like Facebook, Google, and Amazon. IAM (A) is for AWS resource access, Directory Service (C) is for Microsoft AD integration, and SSO (D) is for AWS account and business application access.

¡/details¿

—

Question 19

Which encryption option for S3 provides an audit trail of when keys were used and by whom?

A. SSE-S3 B. SSE-KMS C. SSE-C D. Client-side encryption

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: SSE-KMS integrates with CloudTrail, providing an audit trail of when encryption keys were used and by whom. SSE-S3 (A) doesn't provide this visibility, SSE-C (C) means you manage keys outside AWS, and client-side encryption (D) is entirely managed by you.

¡/details¿

—

Question 20

What is the purpose of VPC Flow Logs?

A. Log API calls in your VPC B. Capture network traffic information C. Monitor VPC configuration changes D. Detect malware in network traffic

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: VPC Flow Logs capture information about IP traffic going to and from network interfaces in your VPC, including source/destination IPs, ports, and protocols. CloudTrail (A) logs API calls, Config (C) monitors configuration changes, and while flow logs can help security analysis, they don't directly detect malware (D).

¡/details¿

—

Question 21

Which compliance program is specifically designed for US federal government agencies?

A. HIPAA B. PCI DSS C. FedRAMP D. SOC 2

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C

Explanation: FedRAMP (Federal Risk and Authorization Management Program) is the US government's cloud security standard. HIPAA (A) is for healthcare, PCI DSS (B) is for payment cards, and SOC 2 (D) is a general security audit framework.

¡/details¿

—

Question 22

A company wants to share an encrypted S3 bucket with another AWS account. What must be configured?

A. S3 bucket policy only B. KMS key policy and S3 bucket policy C. IAM role only D. VPC peering

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B

Explanation: To share an encrypted S3 bucket cross-account, you need to update both the KMS key policy (to allow the other account to decrypt) and the S3 bucket policy (to allow access to objects). Just one or the other won't work. VPC peering (D) is not required for S3 access.

¡/details¿

—

Question 23

Which AWS service protects against DDoS attacks at no additional cost?

A. AWS WAF B. AWS Shield Advanced C. AWS Shield Standard D. Amazon GuardDuty

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C

Explanation: AWS Shield Standard provides DDoS protection at no additional cost and is automatically enabled for all AWS customers. Shield Advanced (B) costs $3,000/month, WAF (A) has its own pricing for rule complexity and requests, and GuardDuty (D) is for threat detection, not DDoS protection.

¡/details¿

—

Question 24

What is the difference between security groups and Network ACLs?

A. Security groups are stateful; NACLs are stateless B. Security groups are stateless; NACLs are stateful C. Both are stateful D. Both are stateless

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: A

Explanation: Security groups are stateful (return traffic is automatically allowed), while Network ACLs are stateless (you must explicitly allow both inbound and outbound traffic). This is a critical difference for the exam.

¡/details¿

—

Question 25

A company must ensure that all data stored in AWS is encrypted at rest and in transit. Which services should they use? (Choose TWO)

A. AWS KMS for encryption at rest B. AWS CloudHSM for encryption in transit C. TLS/SSL for encryption in transit D. AWS Certificate Manager for encryption at rest

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: A and C

Explanation: AWS KMS provides encryption at rest for services like S3, EBS, and RDS (A). TLS/SSL provides encryption in transit for data moving between systems (C). CloudHSM (B) is for hardware-based key storage but not specifically for transit encryption, and ACM (D) provides certificates for TLS/SSL but doesn't encrypt at rest.

¡/details¿

—

3.0.14 Advanced Exam Tips and Scenarios

Exam Tip 1: Shared Responsibility Model Questions

How to identify: Questions ask "who is responsible for..."

Decision tree:

1. Is it infrastructure (physical, network hardware, data centers)? → **AWS**
2. Is it a managed service (RDS, Lambda, DynamoDB)? → **AWS manages infrastructure, you manage data and access**
3. Is it EC2? → **AWS manages hypervisor, you manage OS, applications, data**
4. Is it customer data, encryption, or IAM? → **Always customer**

Common tricky scenarios:

- "Who patches RDS database engine?" → **AWS**
- "Who patches EC2 operating system?" → **Customer**
- "Who configures security groups?" → **Customer**
- "Who secures AWS data centers?" → **AWS**

Exam Tip 2: IAM Policy Evaluation Logic

Order of evaluation:

1. Explicit DENY → Always wins
2. Explicit ALLOW → If no deny exists
3. Implicit DENY → Default if no allow

Remember: One explicit deny overrules all allows!

Example scenario:

```
User has policy: Allow s3:*
Group has policy: Deny s3:DeleteBucket
Result: User can do everything EXCEPT delete buckets
```

Exam Tip 3: Encryption Service Selection

Question type: "Which encryption service should you use when..."
Decision matrix:

Requirement	Solution
Simple S3 encryption	SSE-S3
Need audit trail of key usage	SSE-KMS
Must control keys outside AWS	SSE-C or Client-side
Encrypt EBS volumes	KMS (default)
Encrypt in transit	TLS/SSL, ACM
Meet compliance requirements	KMS with customer managed CMK
Hardware-based key storage	CloudHSM

Exam Tip 4: Security Service Selection

Question type: "Which service detects/protects/monitors..."
Quick reference:

Need	Service
Detect threats with ML	GuardDuty
Find vulnerabilities	Inspector
Discover sensitive data	Macie
Protect against DDoS	Shield
Protect web applications	WAF
Manage encryption keys	KMS
Audit API calls	CloudTrail
Track configurations	Config
Compliance reports	Artifact
Centralized security view	Security Hub

Exam Tip 5: Compliance Program Matching

Pattern recognition for exam:

- "Healthcare data" or "PHI" → **HIPAA**
- "Credit card" or "payment data" → **PCI DSS**
- "Government" or "federal agency" → **FedRAMP**
- "EU residents" or "data privacy" → **GDPR**
- "Audit report" or "financial controls" → **SOC reports**
- "International security standard" → **ISO 27001**

Exam Tip 6: MFA Scenarios

When MFA is the answer:

- Question mentions "additional layer of security"
- Scenario involves privileged users or root account
- Compliance requirement for sensitive operations
- "Something you have" factor is mentioned

MFA is NOT the answer for:

- Service-to-service authentication (use roles)
- Programmatic access from applications (use roles)
- Long-term credential storage (use IAM roles)

Exam Tip 7: Network Security Scenarios

Security Group vs NACL:

Scenario	Use
Need to explicitly deny an IP	NACL
Need stateful filtering	Security Group
Subnet-level protection	NACL
Instance-level protection	Security Group
Process rules in order	NACL
Simple allow rules	Security Group

Exam Tip 8: Identity Federation

Scenario patterns:

- "Corporate users need AWS access" → **SAML federation or IAM Identity Center**
- "Mobile app users" → **Cognito**
- "Social login (Facebook, Google)" → **Cognito**
- "Active Directory integration" → **Directory Service or IAM Identity Center**
- "Multiple AWS accounts, single login" → **IAM Identity Center**

Exam Tip 9: Data Protection Scenarios

Pattern matching:

- "Prevent public access to S3" → **S3 Block Public Access**
- "Track who accesses S3 objects" → **S3 Server Access Logging + CloudTrail**
- "Find sensitive data in S3" → **Macie**
- "Encrypt data before upload" → **Client-side encryption**
- "AWS manages encryption" → **SSE-S3 or SSE-KMS**

Exam Tip 10: Cost Considerations

Free services/features:

- IAM (completely free)
- CloudTrail (first trail free)
- Shield Standard (free DDoS protection)
- S3 SSE-S3 encryption (no extra cost)
- VPC (core features free)
- AWS managed CMKs (free, pay for API calls only)

Paid services:

- Shield Advanced ($3,000/month)
- GuardDuty (pay per GB analyzed)
- Macie (pay per GB scanned)
- Inspector (pay per assessment)
- WAF (pay per rule and requests)
- Customer managed CMKs ($1/month + API calls)

Exam Tip 11: Incident Response Questions

Scenario: "What should you do first when..."

1. Isolate affected resources
2. Preserve evidence (snapshots, logs)
3. Investigate and analyze
4. Remediate
5. Document lessons learned

Key services:

- CloudTrail for forensics
- VPC Flow Logs for network analysis
- GuardDuty for threat detection
- Systems Manager for remediation

Exam Tip 12: Common Exam Traps

Watch out for:

1. "Most cost-effective" → Usually the simpler, managed option
2. "Least operational overhead" → Usually the fully managed service
3. "Most secure" → Usually involves encryption, MFA, least privilege
4. "Best practice" → Follow AWS recommendations (IAM roles, not keys)

Red flags:

- Hardcoding credentials → Never correct
- Root account for daily tasks → Never correct
- Wildcard (*) permissions → Usually incorrect
- Public access to production data → Usually incorrect

—

3.0.15 Key Takeaways

Exam Tip

Remember for the Exam:

¿

Key Point

- **Shared Responsibility Model:** AWS = infrastructure; Customer = data and configuration - **IAM Best Practices:** Root account protection, least privilege, use groups and roles, enable MFA - **Shield Standard:** Free DDoS protection for everyone - **GuardDuty:** Threat detection with ML - **Macie:** Sensitive data discovery in S3 - **Artifact:** Download compliance reports - **Config:** Track configuration changes and compliance - **Organizations:** Multi-account management with consolidated billing and SCPs

—

← Previous: Cloud Concepts — Back to Main — Next: Cloud Technology and Services →

Chapter 4

Domain 3: Cloud Technology and Services

This is the largest domain of the AWS Certified Cloud Practitioner exam, covering core AWS services across compute, storage, networking, databases, and more.

4.0.1 Table of Contents

—

4.0.2 AWS Global Infrastructure

Regions

Geographic areas with multiple Availability Zones

- **Current Count**: 33 Regions worldwide (and growing)
- **Isolation**: Each Region is completely isolated from other Regions
- **Selection Criteria**:
- **Compliance requirements**: Data sovereignty and regulatory requirements
- **Proximity to users**: Lower latency for end users

- **Available services**: Not all services are available in all Regions
- **Pricing**: Costs vary by Region

Exam Tip

Exam Tip: Choose the right Region based on compliance, latency, service availability, and cost considerations.

Availability Zones (AZs)

Discrete data centers within a Region

- **Composition**: One or more discrete data centers per AZ
- **Redundancy**: Each AZ has redundant power, networking, and connectivity
- **Physical Separation**: AZs are physically separated within a Region
- **Connectivity**: Connected with high-bandwidth, low-latency networking
- **Minimum Count**: At least 3 AZs per Region (most have more)
- **High Availability**: Deploy resources across multiple AZs for fault tolerance

Key Benefits:

- Fault isolation
- High availability through redundancy
- Disaster recovery within a Region

Real-World Example: Netflix deploys its content delivery infrastructure across multiple AZs within each Region. If one AZ experiences issues, their application automatically routes traffic to healthy AZs, ensuring uninterrupted streaming for millions of users.

Common Configuration Mistakes:

1. Deploying all resources in a single AZ (no fault tolerance)
2. Not considering cross-AZ data transfer costs
3. Assuming AZ names (us-east-1a) are the same across AWS accounts (they're randomized)
4. Not testing failover between AZs before production deployment

Service Limits by Infrastructure:

- Maximum VPCs per Region: 5 (soft limit, can be increased)
- Maximum subnets per VPC: 200
- Elastic IPs per Region: 5 (soft limit)
- VPC Peering connections per VPC: 125

Edge Locations

Content delivery endpoints worldwide

- **Count**: 400+ Edge Locations globally
- **Primary Use**: CloudFront content caching
- **Performance**: Lower latency for end users
- **Services**: Also used by Route 53, AWS Shield, and AWS WAF
- **Coverage**: More Edge Locations than Regions

AWS Local Zones

Region extensions for ultra-low latency

- Extension of a Region placed closer to specific geographic areas
- Single-digit millisecond latency to end users
- Ideal for latency-sensitive applications (gaming, live video, AR/VR)
- Not available in all locations

AWS Wavelength

5G edge computing

- Embeds AWS compute and storage services within 5G networks
- Ultra-low latency applications
- Mobile edge computing use cases
- Reduces data routing to application servers

AWS Outposts

On-premises AWS infrastructure

- Fully managed service extending AWS infrastructure to on-premises facilities
- Same AWS APIs, tools, and hardware available on-premises
- Enables true hybrid cloud deployments
- AWS manages and maintains the infrastructure
- Run AWS services locally with consistent experience

Key Point

Key Point: Remember the hierarchy: **Regions** contain **Availability Zones**. **Edge Locations** are separate and used primarily for content delivery.

Global Infrastructure Comparison Table

Component	Count	Primary Use	Redundancy Level
Regions	33+	Full AWS service deployment	Isolated from each otl
Availability Zones	100+ (3+ per Region)	Fault-tolerant deployments	Within Region
Edge Locations	400+	Content caching	N/A
Local Zones	16+	Ultra-low latency compute	Extension of parent F
Wavelength Zones	20+	5G edge computing	Within telecom netwo

Infrastructure Selection Flowchart

```
START: Where should I deploy my resources?

> Need global distribution?
   > YES: Deploy in multiple Regions
      > Consider: CloudFront for content, Route 53 for DNS routing

   > NO: Single Region deployment

       > Need high availability?
          > YES: Deploy across multiple AZs
             > Use: Multi-AZ RDS, ALB across AZs, Auto Scaling

          > NO: Single AZ (only for dev/test)

       > Need ultra-low latency?
           > For specific metro areas: Use Local Zones
           > For mobile 5G apps: Use Wavelength
           > For on-premises: Use Outposts
```

Migration Scenario: Data Center to AWS

Scenario: Company with on-premises data center in Chicago needs to migrate to AWS.
Current Setup:

- 100 servers across 2 data centers
- Customers primarily in US and Europe
- Compliance requires data residency in US

AWS Migration Strategy:

1. **Region Selection**: Choose us-east-1 (N. Virginia) for primary and us-west-2 (Oregon) for DR

- Reason: Established Regions with all services available
- Cost: Lower pricing than newer Regions

- Latency: Good for US customers

1. **Network Connectivity**:

- Phase 1: Site-to-Site VPN (immediate, low cost)
- Phase 2: Direct Connect (after 3 months, for production traffic)
- Cost: VPN ~$0.05/hour, Direct Connect ~$0.30/hour (1 Gbps)

1. **High Availability Architecture**:

- Deploy across 3 AZs in us-east-1
- Application Load Balancer across all AZs
- RDS Multi-AZ for databases
- S3 for object storage (automatically multi-AZ)

1. **European Customers**:

- CloudFront distribution with origin in us-east-1
- Edge locations in Europe cache content
- Route 53 latency-based routing

Cost Comparison:

- On-premises: $50,000/month (hardware, power, cooling, staff)
- AWS initial: $35,000/month (no hardware investment)
- AWS optimized (after 6 months with RIs): $22,000/month
- Savings: 56% reduction in monthly costs

—

4.0.3 Compute Services

Amazon EC2 (Elastic Compute Cloud)

Virtual servers in the cloud

Amazon EC2 provides resizable compute capacity in the cloud, allowing you to launch virtual servers (instances) on demand.

Instance Types

Category	Use Case	Examples
General Purpose	Balanced compute, memory, networking	T3, M5
Compute Optimized	High-performance processors	C5, C6
Memory Optimized	Large datasets in memory	R5, X1
Storage Optimized	High sequential read/write access	I3, D2
Accelerated Computing	GPU, FPGA workloads	P3, G4

Detailed Instance Type Comparison

Instance Family	vCPU Range	Memory Range	Network Performance	Best Use Cas
T3/T4g	2-8	0.5-32 GB	Up to 5 Gbps	Burstable, web
M6i/M7g	2-128	8-512 GB	Up to 50 Gbps	Balanced applic
C6i/C7g	2-128	4-256 GB	Up to 50 Gbps	Compute-intens
R6i/R7g	2-128	16-1024 GB	Up to 50 Gbps	Memory-intensi
X2iedn	16-128	256-4096 GB	Up to 100 Gbps	Extreme memo
I4i	2-128	16-1024 GB	Up to 75 Gbps	Storage-intensiv
P4d	96	1152 GB	400 Gbps	ML training
G5	4-96	16-768 GB	Up to 100 Gbps	Graphics-intens

Real-World Use Case: E-Commerce Website
Scenario: Online retailer with variable traffic patterns, peak during holidays.
Solution Architecture:

- **Frontend Web Servers**: T3.medium instances (burstable for normal traffic)
- Cost: $0.0416/hour = ˜$30/month each
- Auto Scaling: 2-20 instances based on demand
- Normal: 4 instances = $120/month
- Peak: 15 instances = $450/month
- **Application Servers**: M5.large instances (consistent performance)
- Cost: $0.096/hour = ˜$70/month each
- Reserved Instances (3-year): 75% discount = $17.50/month each
- Deploy: 8 instances = $140/month with RIs
- **Database**: R5.xlarge (memory-optimized for database)
- Cost: $0.252/hour = ˜$184/month
- Reserved Instance: $46/month

Total Monthly Cost:

- Normal traffic: $306/month

- Peak traffic: $636/month
- Average: ˜$400/month vs $2,000+ on-premises

Performance Metrics:

- Page load time: ¡2 seconds
- Database query response: ¡100ms
- Handles 10,000 concurrent users during peak
- 99.99% uptime with Multi-AZ deployment

EC2 Pricing Models ##### 1. On-Demand Instances

- **Billing**: Pay per hour or per second (minimum 60 seconds)
- **Commitment**: No upfront commitment or long-term contract
- **Cost**: Highest per-hour cost
- **Best For**:
- Unpredictable workloads
- Short-term, spiky workloads
- Testing and development
- Applications with flexible start/stop times

2. Reserved Instances (RI)

- **Commitment**: 1 or 3 year terms
- **Discount**: Up to 75% compared to On-Demand pricing
- **Types**:
- **Standard RI**: Maximum discount, cannot change instance type
- **Convertible RI**: Lower discount, can change instance type/family
- **Payment Options**: All upfront, partial upfront, or no upfront
- **Best For**: Steady-state workloads with predictable usage

3. Savings Plans

- **Commitment**: Consistent compute usage measured in $/hour
- **Term**: 1 or 3 year commitment
- **Discount**: Up to 72% compared to On-Demand

- **Flexibility**: More flexible than Reserved Instances
- **Coverage**: Applies to EC2, Lambda, and Fargate
- **Best For**: Flexible compute usage with commitment to consistent spend

4. Spot Instances

- **Mechanism**: Bid on unused EC2 capacity
- **Discount**: Up to 90% compared to On-Demand pricing
- **Interruption**: Can be terminated by AWS with 2-minute warning
- **Best For**:
- Fault-tolerant applications
- Flexible workloads
- Batch processing
- Big data analysis
- **Not Suitable For**: Critical workloads or databases

5. Dedicated Hosts

- **Definition**: Physical EC2 server dedicated exclusively to your use
- **Cost**: Most expensive option
- **Use Cases**:
- Regulatory compliance requirements
- Server-bound software licenses
- Socket/core visibility needed for licensing
- **Control**: Full control over instance placement

6. Dedicated Instances

- **Definition**: Instances run on hardware dedicated to a single customer
- **Sharing**: May share hardware with other instances in your account
- **Cost**: Less expensive than Dedicated Hosts
- **Isolation**: Hardware-level isolation from other AWS accounts

EC2 Pricing Model Comparison Table

Pricing Model	Discount vs On-Demand	Commitment	Interruption Risk	B
On-Demand	0% (baseline)	None	None	Sl
Reserved (Standard)	Up to 75%	1 or 3 years	None	St
Reserved (Convertible)	Up to 66%	1 or 3 years	None	F
Savings Plans	Up to 72%	1 or 3 years	None	F
Spot Instances	Up to 90%	None	Can be interrupted	Fa
Dedicated Hosts	Varies	1 or 3 years	None	L

Cost Comparison Example: m5.xlarge in us-east-1

Scenario	Pricing Model	Hourly Cost	Month
Dev/Test (8hrs/day, 22 days/month)	On-Demand	$0.192	$337
Production (24/7)	On-Demand	$0.192	$140
Production (24/7)	Standard RI (3yr, all upfront)	$0.112	$82
Production (24/7)	Savings Plan (3yr)	$0.118	$86
Batch Processing (avg 50% uptime)	Spot	$0.038	$14

Cost Optimization Strategy:

1. **Baseline workload**: Use Standard RIs or Savings Plans (75% savings)
2. **Variable workload**: Use Savings Plans for flexibility
3. **Peak capacity**: Auto Scale with On-Demand
4. **Batch/fault-tolerant**: Use Spot Instances (90% savings)

Common Configuration Mistakes:

1. Running On-Demand for steady-state workloads (missing 75% savings)
2. Using Spot Instances for databases or critical workloads
3. Over-provisioning instance size (wasting resources)
4. Not enabling detailed monitoring for performance optimization
5. Forgetting to delete stopped instances (still incurs EBS charges)
6. Not using Auto Scaling (manually managing capacity)
7. Storing data on instance store for persistent data (data loss on stop)
8. Not tagging instances (difficult cost allocation)

Service Limits and Quotas:

- On-Demand instance limit: Varies by instance family (typically 20-1280 vCPUs)
- Spot instance limit: 20 Spot instances per Region (soft limit)

- Reserved Instances: 20 per month (soft limit)
- Elastic IPs: 5 per Region (soft limit)
- EBS volumes: 5,000 per Region (soft limit)
- EBS snapshots: 10,000 per Region (soft limit)

Monitoring and Troubleshooting:

1. **CloudWatch Metrics** (5-minute intervals, free):

- CPUUtilization
- NetworkIn/NetworkOut
- DiskReadOps/DiskWriteOps
- StatusCheckFailed

1. **Detailed Monitoring** (1-minute intervals, paid):

- Enable for production workloads
- Cost: $0.14 per instance per month
- Better for Auto Scaling responsiveness

1. **Common Issues**:

- High CPU: Resize instance or optimize application
- High memory: Use memory-optimized instance type
- Network bottleneck: Use enhanced networking or larger instance
- Disk I/O bottleneck: Use Provisioned IOPS EBS volumes

1. **Troubleshooting Commands**:

‘ # Check system status aws ec2 describe-instance-status –instance-ids i-1234567890abcdef0
View CloudWatch metrics aws cloudwatch get-metric-statistics –namespace AWS/EC2
\ –metric-name CPUUtilization –dimensions Name=InstanceId,Value=i-xxx
Check security group rules aws ec2 describe-security-groups –group-ids sg-xxx ‘

Auto Scaling Automatically adjust capacity based on demand

- **Scaling Actions**:
- **Scale Out**: Add instances when demand increases
- **Scale In**: Remove instances when demand decreases
- **Scaling Policies**:
- **Target Tracking**: Maintain a specific metric (e.g., 50% CPU utilization)
- **Step Scaling**: Scale based on CloudWatch alarm thresholds
- **Scheduled Scaling**: Scale based on predictable time-based patterns
- **Benefits**:
- Improved availability
- Cost optimization
- Fault tolerance
- Works seamlessly with Elastic Load Balancing

AWS Lambda

Serverless compute service
Run code without provisioning or managing servers.
Key Features:

- **Zero Server Management**: No infrastructure to provision or manage
- **Automatic Scaling**: Scales automatically from a few requests to thousands
- **Subsecond Metering**: Pay only for compute time consumed (billed per 100ms)
- **Language Support**: Python, Node.js, Java, Go, C#, Ruby, PowerShell
- **Execution Limit**: Maximum 15 minutes per execution
- **Event-Driven**: Triggered by events from AWS services or custom applications

Benefits:

- No server management overhead
- Continuous automatic scaling
- Cost-effective for variable workloads
- Built-in high availability and fault tolerance

Common Use Cases:

- Real-time file processing

- Data transformation and ETL
- Serverless backends for web/mobile apps
- IoT backends
- Scheduled tasks (cron jobs)

Lambda vs EC2 Comparison

Aspect	AWS Lambda	Amazon EC2
Management	Fully managed, zero administration	You manage OS, patches, scaling
Scaling	Automatic, instant (0-10,000+ concurrent)	Manual or Auto Scaling (minutes)
Pricing	Per request + duration (100ms increments)	Per hour/second of instance runtime
Max Duration	15 minutes per invocation	Unlimited (runs continuously)
Cold Start	Yes (50-200ms initial delay)	No (always running)
State	Stateless (ephemeral storage)	Stateful (persistent storage)
Best For	Event-driven, sporadic workloads	Long-running, steady workloads

Real-World Use Case: Image Processing Service

Scenario: Photography platform processes user-uploaded images (resize, thumbnail, watermark).

Lambda Solution:

```
Architecture:
User uploads image to S3
  > S3 triggers Lambda function
      > Lambda processes image
          > Saves processed images back to S3
          > Updates DynamoDB with metadata
```

Cost Analysis (1 million images per month):

- Average processing time: 3 seconds per image
- Memory allocation: 1024 MB
- Compute: 1M requests × 3 seconds × $0.0000166667/GB-second = $50
- Requests: 1M × $0.20 per 1M = $0.20
- **Total: $50.20/month**

EC2 Comparison:

- t3.medium running 24/7: ˜$30/month (On-Demand)
- BUT: Needs management, updates, monitoring
- AND: Wastes capacity during low-traffic periods
- **Total with overhead: $100-150/month**

Lambda Advantages for this use case:

- 66% cost savings
- Zero server management
- Automatic scaling (handles traffic spikes)
- Only pay for actual processing time

Cost Comparison Example: Lambda Pricing

Monthly Requests	Avg Duration	Memory	Compute Cost	Request Cost	Total Cos
100,000	200ms	512 MB	$0.17	$0.02	$0.19
1,000,000	1 second	1024 MB	$16.67	$0.20	$16.87
10,000,000	500ms	512 MB	$41.67	$2.00	$43.67
100,000,000	200ms	256 MB	$33.33	$20.00	$53.33

Integration Example: Serverless Web Application

```
Architecture:
CloudFront (CDN)
  > S3 (Static website hosting)
      > API Gateway (REST API)
          > Lambda (Business logic)
              > DynamoDB (User data)
              > RDS Aurora Serverless (Transactional data)
              > S3 (File storage)
```

Benefits:

- No servers to manage
- Automatic scaling from 0 to millions of requests
- Pay only for actual usage
- High availability built-in

Common Configuration Mistakes:

1. Not setting appropriate timeout (default 3s, max 15min)
2. Allocating too much or too little memory (affects CPU allocation)
3. Not handling cold starts for latency-sensitive applications
4. Storing state in /tmp (lost between invocations, max 10 GB)
5. Not implementing proper error handling and retries
6. Exceeding concurrent execution limit (default 1,000)
7. Not using Lambda Layers for shared dependencies
8. Embedding sensitive data in code (use environment variables/Secrets Manager)

Service Limits and Quotas:

- Concurrent executions: 1,000 (soft limit, can be increased)
- Function timeout: 15 minutes (hard limit)
- Deployment package size: 50 MB (zipped), 250 MB (unzipped)
- /tmp directory storage: 10 GB
- Environment variables: 4 KB total
- Memory allocation: 128 MB to 10,240 MB (64 MB increments)
- Ephemeral storage: 512 MB to 10,240 MB

Monitoring and Troubleshooting:

1. **CloudWatch Metrics** (automatic):

- Invocations
- Duration
- Errors
- Throttles
- Concurrent Executions

1. **CloudWatch Logs**:

- All console.log/print statements
- Execution start/end
- Error traces
- Custom metrics

1. **AWS X-Ray** (distributed tracing):

- Trace requests through entire application
- Identify performance bottlenecks
- Visualize service map

1. **Common Issues**:

- Cold starts: Use provisioned concurrency (extra cost)
- Timeout errors: Increase timeout or optimize code
- Out of memory: Increase memory allocation

- Throttling: Request concurrent execution limit increase

Performance Optimization Tips:

1. Minimize deployment package size
2. Use Lambda Layers for shared dependencies
3. Keep functions warm with scheduled invocations (if needed)
4. Optimize memory allocation (more memory = more CPU)
5. Use environment variables for configuration
6. Connection pooling for database connections
7. Lazy load dependencies outside handler function

Compute Service Selection Flowchart

```
START: Which compute service should I use?

> Need full control over OS and applications?
   > YES: Amazon EC2
       > Simple setup needed? → Lightsail
       > Application deployment focus? → Elastic Beanstalk
       > Full control? → EC2

> Using containers?
   > YES:
       > Already use Kubernetes? → EKS
       > Want AWS-native? → ECS
       > Want serverless containers? → Fargate
       > Batch processing? → AWS Batch

> Event-driven, short-running tasks?
    > YES: AWS Lambda
        > Workflows needed? → Step Functions + Lambda
        > APIs? → API Gateway + Lambda
        > Event processing? → EventBridge + Lambda
```

Migration Scenario: Monolith to Serverless
Current State: Monolithic PHP application on 3 EC2 instances

- Monthly cost: $150 (EC2) + $50 (RDS) = $200
- Maintenance: 10 hours/month
- Scaling: Manual, 30-minute deployment

Target State: Serverless architecture

- Static content: S3 + CloudFront

- APIs: API Gateway + Lambda
- Database: Aurora Serverless
- Authentication: Cognito

Migration Steps:

1. Extract static assets to S3 (Week 1)
2. Create CloudFront distribution (Week 1)
3. Migrate APIs to Lambda (Weeks 2-4)
4. Migrate database to Aurora Serverless (Week 5)
5. Implement Cognito authentication (Week 6)
6. Decommission EC2 instances (Week 7)

After Migration:

- Monthly cost: $60 (80% traffic reduction)
- Maintenance: 2 hours/month (75% reduction)
- Scaling: Automatic, instant
- Deployment: Minutes (CI/CD pipeline)
- Performance: 40% faster (CloudFront CDN)

Amazon Lightsail

Simplified cloud platform for simple workloads

- Easy-to-use virtual private servers (VPS)
- Predictable monthly pricing
- Bundled resources: compute, storage, networking, DNS
- Pre-configured application stacks (WordPress, Magento, LAMP, etc.)
- Ideal for:
- Simple web applications
- Blogs and websites
- Small business applications
- Development and test environments
- Perfect for users new to AWS or with simple requirements

AWS Elastic Beanstalk

Platform as a Service (PaaS)
Deploy and manage applications without infrastructure complexity.
Key Features:

- **Language Support**: Java, .NET, PHP, Node.js, Python, Ruby, Go, Docker
- **Automatic Management**: Capacity provisioning, load balancing, auto-scaling, health monitoring
- **Developer Control**: Retain full control over underlying AWS resources
- **No Additional Charge**: Pay only for the AWS resources used
- **Quick Deployment**: Deploy applications in minutes

Best For:

- Web applications
- Developers who want to focus on code, not infrastructure
- Standard application architectures

Amazon ECS (Elastic Container Service)

Fully managed container orchestration
Run and scale Docker containers on AWS.
Launch Types:

- **EC2 Launch Type**:
- You manage the underlying EC2 instances
- More control over infrastructure
- Good for cost optimization with Reserved Instances
- **Fargate Launch Type**:
- Serverless container deployment
- AWS manages the infrastructure
- Pay only for container resources

Features:

- Deep AWS integration
- Service discovery
- Load balancing
- Auto Scaling

Use Cases:

- Microservices architectures
- Batch processing
- Machine learning applications

Amazon EKS (Elastic Kubernetes Service)

Managed Kubernetes service

- Fully managed Kubernetes control plane
- Compatible with standard Kubernetes tooling and plugins
- Automatic Kubernetes version upgrades and patching
- Integrates with AWS services (IAM, VPC, CloudWatch)
- Multi-AZ control plane for high availability
- Best for teams already invested in Kubernetes ecosystem

AWS Fargate

Serverless compute engine for containers

- Works with both ECS and EKS
- No need to provision, configure, or scale EC2 instances
- Pay only for the vCPU and memory resources your containers use
- Automatic scaling and load balancing
- Focus on building applications, not managing infrastructure

Key Point

Key Point: Compute spectrum from most to least control: **EC2** (full control) → **Containers (ECS/EKS)** (moderate control) → **Lambda/Fargate** (least control, most abstraction)

—

4.0.4 Storage Services

Amazon S3 (Simple Storage Service)

Object storage service for any amount of data

S3 is a highly durable, scalable object storage service designed to store and retrieve any amount of data from anywhere.

Key Concepts

Concept	Description
Buckets	Containers for objects with globally unique names
Objects	Files stored in buckets (up to 5 TB each)
Keys	Unique identifier for each object in a bucket
Durability	99.999999999% (11 nines)
Availability	Varies by storage class

S3 Storage Classes ##### 1. S3 Standard

- **Use Case**: Frequently accessed data
- **Performance**: Low latency and high throughput
- **Availability**: 99.99%
- **Cost**: Most expensive storage cost
- **Best For**: Active databases, frequently accessed content, dynamic websites

2. S3 Intelligent-Tiering

- **Automation**: Automatically moves objects between access tiers
- **Optimization**: Cost optimization for unknown or changing access patterns
- **Tiers**: Frequent Access, Infrequent Access, Archive Instant Access, Archive Access, Deep Archive Access
- **Monitoring Fee**: Small monthly fee per object
- **No Retrieval Fees**: Between Frequent and Infrequent Access tiers

3. S3 Standard-IA (Infrequent Access)

- **Use Case**: Infrequently accessed data that needs rapid access when required
- **Cost**: Lower storage cost, but retrieval fee applies
- **Availability**: 99.9%
- **Minimum Duration**: 30-day minimum storage charge
- **Best For**: Backups, disaster recovery, long-term storage

4. S3 One Zone-IA

- **Storage**: Single Availability Zone (not multiple AZs)
- **Cost**: 20% less than Standard-IA
- **Availability**: 99.5%

- **Risk**: Data lost if AZ is destroyed
- **Best For**: Recreatable data, secondary backup copies

5. S3 Glacier Instant Retrieval

- **Use Case**: Archive data requiring instant access
- **Retrieval**: Millisecond retrieval times
- **Cost**: Lower storage cost than Standard-IA
- **Minimum Duration**: 90-day minimum storage charge
- **Best For**: Medical images, news media assets accessed once per quarter

6. S3 Glacier Flexible Retrieval

- **Use Case**: Archive data with retrieval times from minutes to hours
- **Retrieval Options**:
- **Expedited**: 1-5 minutes
- **Standard**: 3-5 hours
- **Bulk**: 5-12 hours (lowest cost)
- **Minimum Duration**: 90-day minimum storage charge
- **Best For**: Backup and archive data accessed 1-2 times per year

7. S3 Glacier Deep Archive

- **Use Case**: Long-term archive and digital preservation
- **Cost**: Lowest cost storage class
- **Retrieval Time**: 12-48 hours
- **Minimum Duration**: 180-day minimum storage charge
- **Best For**: Compliance archives, digital preservation, data retained for 7-10+ years

S3 Features Versioning

- Keep multiple versions of an object
- Protect against accidental deletion
- Can be enabled/suspended per bucket

Lifecycle Policies

- Automatically transition objects between storage classes
- Automatically delete objects after a specified time
- Cost optimization through automation

Encryption

- **Server-Side Encryption (SSE)**: S3 encrypts objects
- SSE-S3: S3-managed keys
- SSE-KMS: AWS KMS-managed keys
- SSE-C: Customer-provided keys
- **Client-Side Encryption**: Encrypt before uploading

Access Control

- Bucket policies (resource-based)
- IAM policies (identity-based)
- Access Control Lists (ACLs) - legacy
- S3 Block Public Access settings

Additional Features

- Static website hosting
- Cross-Region Replication (CRR)
- Same-Region Replication (SRR)
- Transfer Acceleration (via CloudFront edge locations)
- Event notifications
- S3 Select (query data using SQL)

S3 Storage Class Cost Comparison

Storage Class	Storage Cost (per GB/month)	Retrieval Cost	Min
S3 Standard	$0.023	None	None
S3 Intelligent-Tiering	$0.023-$0.0125 + $0.0025 monitoring	None (auto-tier)	None
S3 Standard-IA	$0.0125	$0.01 per GB	30 d
S3 One Zone-IA	$0.01	$0.01 per GB	30 d
S3 Glacier Instant Retrieval	$0.004	$0.03 per GB	90 d
S3 Glacier Flexible Retrieval	$0.0036	$0.01-$0.03 per GB	90 d
S3 Glacier Deep Archive	$0.00099	$0.02 per GB	180

Cost Example: 100 TB of Data Storage

Scenario	Storage Class	Monthly Storage	Retrieval (10% monthly
Active Website	S3 Standard	$2,355	$0
Backup (weekly access)	S3 Standard-IA	$1,280	$102
Compliance Archive	Glacier Deep Archive	$102	$205
Unknown Pattern	Intelligent-Tiering	$850-$2,355	$0

Real-World Use Case: Media Company
Scenario: Video streaming platform with 500 TB of content.
Storage Strategy:

1. **Recent content** (20 TB, last 30 days): S3 Standard

- High access rate, low latency needed
- Cost: 20,000 GB × $0.023 = $460/month

1. **Popular library** (100 TB, accessed weekly): S3 Intelligent-Tiering

- Access patterns vary by content popularity
- Cost: ˜$1,700/month (auto-optimizes)

1. **Archive content** (300 TB, accessed rarely): Glacier Flexible Retrieval

- Old shows, accessed 1-2 times per year
- Cost: 300,000 GB × $0.0036 = $1,080/month

1. **Legal/compliance** (80 TB, 7-year retention): Glacier Deep Archive

- Almost never accessed
- Cost: 80,000 GB × $0.00099 = $79/month

Total: $3,319/month vs $11,500/month if everything in S3 Standard (71% savings)

S3 Performance Benchmarks

Metric	Performance	Notes
Request Rate	3,500 PUT/COPY/POST/DELETE per second per prefix	Can scale by using
Request Rate	5,500 GET/HEAD per second per prefix	Automatic scaling
Throughput	No limit	Scales automatical
Latency	100-200ms (first byte)	Standard, IA, One
Durability	99.999999999% (11 nines)	Designed to sustai
Availability SLA	99.9-99.99%	Varies by storage c

Common Configuration Mistakes:

1. Using S3 Standard for infrequently accessed data (overpaying)
2. Not enabling versioning for critical data (risk of data loss)
3. Not using lifecycle policies (manual tier management)
4. Allowing public access unintentionally (security risk)
5. Not enabling server-side encryption (compliance issues)
6. Not using S3 Transfer Acceleration for global uploads
7. Storing small objects inefficiently (minimum billable size)
8. Not implementing proper backup/replication strategy

Service Limits and Quotas:

- Buckets per account: 100 (soft limit, can be increased to 1,000)
- Object size: 5 TB maximum
- Single PUT: 5 GB maximum (use multipart for larger)
- Bucket name: 3-63 characters, globally unique
- Bucket policy: 20 KB maximum
- Tags per object: 10

Integration Example: Automated Data Pipeline

```
Architecture:
On-premises data
  > AWS DataSync / S3 Transfer
      > S3 (Standard)
          > Lambda (process new files)
             > DynamoDB (metadata)

          > Lifecycle Policy (30 days)
```

```
        > S3 Intelligent-Tiering

    > Lifecycle Policy (90 days)
        > Glacier Flexible Retrieval

    > Lifecycle Policy (365 days)
        > Glacier Deep Archive
```

Monitoring and Troubleshooting:

1. **CloudWatch Metrics**:

- NumberOfObjects
- BucketSizeBytes
- AllRequests
- 4xxErrors / 5xxErrors
- FirstByteLatency

1. **S3 Storage Lens** (analytics):

- Usage and activity metrics
- Cost optimization recommendations
- Data protection insights
- Access patterns

1. **S3 Access Logging**:

- Track all requests to bucket
- Security and access audits
- Usage analysis

1. **Common Issues**:

- Slow upload: Use S3 Transfer Acceleration or multipart upload
- 403 Forbidden: Check bucket policy and IAM permissions
- 503 SlowDown: Reduce request rate or use prefixes
- High costs: Implement lifecycle policies and use appropriate storage class

Best Practices:

1. **Security**:

- Enable S3 Block Public Access

- Use bucket policies and IAM roles
- Enable versioning for critical data
- Enable server-side encryption
- Use S3 Object Lock for compliance

1. **Cost Optimization**:

- Use S3 Intelligent-Tiering for unknown patterns
- Implement lifecycle policies
- Delete incomplete multipart uploads
- Use S3 Storage Lens for optimization insights
- Consider Requester Pays for public datasets

1. **Performance**:

- Use prefixes to distribute load
- Enable S3 Transfer Acceleration for global users
- Use CloudFront for frequently accessed content
- Implement multipart upload for large files
- Use byte-range fetches for large downloads

1. **Data Protection**:

- Enable versioning
- Use Cross-Region Replication for DR
- Implement MFA Delete for critical buckets
- Regular backup testing
- Use S3 Object Lock for WORM requirements

Storage Service Comparison Table

Service	Type	Use Case	Shared Access	Pricing
S3	Object	Backups, web content, data lakes	Via API/URL	Per GB s
EBS	Block	EC2 instance storage	Single instance	Per GB p
EFS	File	Shared file storage	Multiple instances	Per GB u
FSx for Windows	File	Windows file shares	Multiple instances	Per GB p
FSx for Lustre	File	HPC, ML training	Multiple instances	Per GB p
Storage Gateway	Hybrid	On-premises to cloud	On-premises + cloud	Per GB s
Snow Family	Physical	Offline data transfer	Physical device	Per devic

Amazon EBS (Elastic Block Store)

Block-level storage volumes for EC2 instances

Persistent block storage that can be attached to EC2 instances, similar to a hard drive.

Key Characteristics:

- Persistent storage that exists independently of instance lifetime
- Attached to a single EC2 instance at a time
- Automatically replicated within its Availability Zone
- Snapshots stored in Amazon S3
- Can detach and reattach to different instances
- Supports encryption at rest and in transit

EBS Volume Types

Type	Category	Use Case	Performance
gp3	General Purpose SSD	Balanced price/performance	3,000-16,000 IO
gp2	General Purpose SSD	Balanced price/performance	Baseline 3 IOP
io2/io1	Provisioned IOPS SSD	Mission-critical, high-performance	Up to 64,000 IO
st1	Throughput Optimized HDD	Frequently accessed, throughput-intensive	Good for big d
sc1	Cold HDD	Infrequently accessed	Lowest cost, fil

Best Practices:

- Take regular snapshots for backup
- Use Provisioned IOPS for database workloads
- General Purpose SSD for most use cases

Amazon EFS (Elastic File System)

Managed NFS file system for EC2

- **File System Type**: Network File System (NFS) v4.1
- **Shared Access**: Multiple EC2 instances can access simultaneously
- **Automatic Scaling**: Grows and shrinks automatically as you add/remove files
- **Pricing**: Pay only for storage used (no pre-provisioning)
- **Availability**: Regional service spanning multiple Availability Zones
- **Performance Modes**:
- **General Purpose**: Low latency (web serving, CMS)
- **Max I/O**: Higher latency, massively parallel (big data, media processing)

Use Cases:

- Content management systems
- Web serving
- Data sharing and collaboration
- Development environments
- Container storage

Comparison:

- **EBS**: Single instance, must be in same AZ
- **EFS**: Multiple instances, cross-AZ access
- **S3**: Object storage, unlimited instances via API

AWS Storage Gateway

Hybrid cloud storage service
Connects on-premises environments to AWS cloud storage.

Gateway Types 1. File Gateway

- Presents NFS/SMB file shares
- Files stored as objects in S3
- Local cache for frequently accessed data
- Use Case: Cloud-backed file shares

2. Volume Gateway

- Presents iSCSI block storage volumes
- Two modes:
- **Cached Volumes**: Primary data in S3, cache on-premises
- **Stored Volumes**: Primary data on-premises, async backup to S3
- Use Case: Block storage backup and disaster recovery

3. Tape Gateway

- Virtual tape library (VTL)
- Integrates with existing backup software
- Store virtual tapes in S3 and Glacier
- Use Case: Replace physical tape backup infrastructure

Benefits:

- Seamless cloud integration
- Local caching for low latency
- Cost-effective storage
- Disaster recovery

AWS Snow Family

Physical devices for data migration and edge computing
Move large amounts of data into and out of AWS using secure physical devices.

Devices

Device	Storage Capacity	Use Case
Snowcone	8 TB usable	Smallest, portable, edge computing
Snowball Edge Storage Optimized	80 TB	Large data migrations
Snowball Edge Compute Optimized	42 TB + compute	Edge computing + storage
Snowmobile	100 PB	Massive data center migration

Snowball Edge Features:

- Can run EC2 instances and Lambda functions
- Cluster multiple devices together
- Storage and compute in disconnected environments

Process:

1. Order device from AWS

2. AWS ships device to your location
3. Connect device and copy data
4. Ship device back to AWS
5. AWS loads data into your S3 bucket

When to Use:

- **Snowcone/Snowball**: Terabytes to petabytes of data
- **Snowmobile**: 10 PB or more
- Limited bandwidth or expensive network transfer
- Physical data migration requirements

Exam Tip

Exam Tip: Choose storage based on use case: - **S3**: Object storage, web content, backups - **EBS**: EC2 instance block storage, databases - **EFS**: Shared file system across multiple instances - **Storage Gateway**: Hybrid cloud storage - **Snow Family**: Large-scale data migration

—

4.0.5 Database Services

Amazon RDS (Relational Database Service)

Managed relational database service

RDS makes it easy to set up, operate, and scale relational databases in the cloud.
Supported Database Engines:

- MySQL
- PostgreSQL
- MariaDB
- Oracle Database
- Microsoft SQL Server
- Amazon Aurora

Key Features:

- **Automated Management**: Backups, patching, scaling
- **No OS Access**: Fully managed (no SSH access)
- **High Availability**: Multi-AZ deployments
- **Scalability**: Read replicas for read-heavy workloads

- **Security**: Encryption at rest and in transit
- **Monitoring**: CloudWatch integration

Pricing: Pay for instance type, storage, backups, and data transfer

Multi-AZ Deployments High availability and disaster recovery

- **Replication**: Synchronous replication to standby instance in different AZ
- **Automatic Failover**: Automatic failover to standby if primary fails
- **Purpose**: High availability and disaster recovery (not performance)
- **Downtime**: Minimal during failover (typically under 1 minute)
- **Backup**: Backups taken from standby to avoid I/O impact

When to Use: Production workloads requiring high availability

Read Replicas Scale read-heavy workloads

- **Replication**: Asynchronous replication from primary
- **Purpose**: Improve read performance, not high availability
- **Location**: Can be in same AZ, different AZ, or different Region
- **Promotion**: Can be promoted to standalone database
- **Limit**: Up to 5 read replicas per primary database
- **Use Cases**: Reporting, analytics, read-heavy applications

Comparison:

- **Multi-AZ**: Synchronous, same Region, automatic failover, HA/DR
- **Read Replicas**: Asynchronous, can be cross-Region, read scaling

Amazon Aurora

AWS cloud-native relational database

- **Compatibility**: MySQL and PostgreSQL compatible
- **Performance**:
- 5x faster than standard MySQL
- 3x faster than standard PostgreSQL
- **Read Replicas**: Up to 15 Aurora read replicas
- **Storage**: Automatically scales from 10 GB to 128 TB

- **Durability**: 6 copies of data across 3 Availability Zones
- **Availability**: Self-healing storage, automated failover
- **Cost**: More expensive than RDS, but better performance and features

Aurora Serverless:

- On-demand, auto-scaling configuration
- Automatically starts up, scales, and shuts down
- Pay per second for resources consumed
- Ideal for infrequent, intermittent, or unpredictable workloads

When to Choose Aurora:

- Need better performance than standard RDS
- High availability requirements
- Rapid scaling needs
- MySQL or PostgreSQL compatibility required

Amazon DynamoDB

Fully managed NoSQL database service
Fast and flexible NoSQL database for any scale.
Key Characteristics:

- **Type**: Key-value and document database
- **Performance**: Single-digit millisecond latency at any scale
- **Serverless**: Fully managed, no servers to provision
- **Scaling**: Automatic scaling of throughput and storage
- **Availability**: Multi-AZ replication, highly available

Features:

- **Global Tables**: Multi-region, multi-active replication
- **DynamoDB Streams**: Capture item-level changes
- **DAX (DynamoDB Accelerator)**: In-memory cache for microsecond latency
- **Transactions**: ACID transactions support
- **Backup**: Point-in-time recovery, on-demand backups

Pricing Models:

- **On-Demand**: Pay per request (unpredictable workloads)

- **Provisioned Capacity**: Reserve read/write capacity units (predictable workloads)

Use Cases:

- Mobile and web applications
- Gaming applications
- IoT data storage
- Session management
- Real-time bidding

Amazon ElastiCache

In-memory caching service
Managed Redis or Memcached for improved application performance.
Supported Engines:

- **Redis**: Advanced data structures, persistence, replication, pub/sub
- **Memcached**: Simple caching, multi-threading

Benefits:

- **Performance**: Microsecond latency
- **Reduced Database Load**: Cache frequently accessed data
- **Scalability**: Easily scale in/out
- **Availability**: Multi-AZ with automatic failover (Redis)

Common Use Cases:

- Database query result caching
- Session stores for web applications
- Real-time analytics
- Leaderboards and counting
- Message queues (Redis)

When to Use:

- Improve read performance
- Reduce database costs
- Store session data
- Real-time applications

Amazon Redshift

Fully managed data warehouse

Fast, scalable data warehouse for analytics.

Key Features:

- **Scale**: Petabyte-scale data warehouse
- **Storage**: Columnar storage format
- **Processing**: Massively parallel processing (MPP)
- **Query Language**: Standard SQL
- **Integration**: Integrates with BI tools (Tableau, Power BI, QuickSight)
- **Performance**: 10x better performance than traditional data warehouses

Use Cases:

- Business intelligence and analytics
- Historical data analysis
- Complex queries on large datasets
- Data consolidation from multiple sources

Redshift Spectrum:

- Query data directly in S3 without loading
- Extend queries beyond Redshift data warehouse

Amazon Neptune

Fully managed graph database

- **Type**: Graph database service
- **Models**: Supports Property Graph and RDF graph models
- **Query Languages**: Gremlin and SPARQL
- **Performance**: Fast query execution on billions of relationships
- **Availability**: Multi-AZ, read replicas

Use Cases:

- Social networking applications
- Recommendation engines
- Fraud detection
- Knowledge graphs
- Network and IT operations

Amazon DocumentDB

MongoDB-compatible document database

- **Compatibility**: MongoDB-compatible (supports MongoDB APIs)
- **Management**: Fully managed by AWS
- **Scaling**: Scale storage and compute independently
- **Availability**: Multi-AZ, automated backups
- **Performance**: 2x throughput improvement over MongoDB

Use Cases:

- Content management systems
- Product catalogs
- User profiles
- Mobile and web app backends

Key Point

Key Point: Database selection guide: - **RDS/Aurora**: Relational databases, ACID transactions, SQL - **DynamoDB**: NoSQL key-value, high scale, low latency - **Redshift**: Data warehouse, analytics, BI - **ElastiCache**: In-memory caching, session storage - **Neptune**: Graph databases, relationships - **DocumentDB**: Document database, MongoDB workloads

Database Service Detailed Comparison

Database	Type	Engine Options
RDS	Relational	MySQL, PostgreSQL, MariaDB, Oracle, SQL Server, Aurora
Aurora	Relational	MySQL, PostgreSQL
DynamoDB	NoSQL Key-Value	N/A
ElastiCache	In-Memory	Redis, Memcached
Redshift	Data Warehouse	PostgreSQL-compatible
Neptune	Graph	Gremlin, SPARQL
DocumentDB	Document	MongoDB-compatible
Timestream	Time Series	N/A

Database Selection Flowchart

```
START: Which database service should I use?

> Need SQL and ACID transactions?
   > YES: Relational Database
       > Maximum performance needed? → Aurora
```

```
        > Oracle/SQL Server licensing? → RDS (Oracle/SQL Server)
        > MySQL/PostgreSQL? → RDS or Aurora
        > Analytics/BI workload? → Redshift

> Need to cache or session storage?
   > YES: ElastiCache
        > Advanced data structures? → Redis
        > Simple caching? → Memcached

> Need document database?
   > YES:
        > MongoDB workloads? → DocumentDB
        > Flexible NoSQL? → DynamoDB

> Need graph database?
   > YES: Neptune

> Need NoSQL at massive scale?
   > YES: DynamoDB
        > Predictable traffic? → Provisioned capacity
        > Variable traffic? → On-Demand capacity

> Time-series data (IoT, metrics)?
    > YES: Timestream
```

Real-World Database Use Cases

Use Case 1: E-Commerce Platform
Requirements:

- Product catalog: 1M products
- User accounts: 10M users
- Orders: 1M transactions/month
- Shopping carts: Temporary, high read/write

Architecture:

```
Frontend (S3 + CloudFront)
  > API Gateway + Lambda
      > DynamoDB (Shopping carts, sessions)
         - On-Demand capacity
         - Single-digit millisecond latency
         - Cost: \~{}\$200/month

      > ElastiCache Redis (Product catalog cache)
         - cache.t3.medium: \$0.068/hour
         - Cost: \~{}\$50/month
         - 99\% cache hit rate
```

```
   > Aurora MySQL (Orders, inventory)
      - db.r5.large: \$0.29/hour
      - Multi-AZ for HA
      - Cost: \~{}\$430/month

   > Redshift (Analytics, business intelligence)
       - dc2.large: \$0.25/hour (2 nodes)
       - Cost: \~{}\$360/month
```

Total Database Cost: ~$1,040/month
Performance:

- Cart operations: ¡10ms (DynamoDB)
- Product lookups: ¡5ms (ElastiCache)
- Order processing: ¡50ms (Aurora)
- Analytics queries: seconds to minutes (Redshift)

Use Case 2: Social Media Application
Requirements:

- User profiles and relationships
- Activity feeds
- Real-time messaging
- Content recommendations

Architecture:

```
Application Layer
  > Neptune (User relationships, friend graphs)
     - db.r5.large: \$0.348/hour
     - Cost: \~{}\$254/month
     - Query: "Friends of friends" in milliseconds

  > DynamoDB (User profiles, posts, activity feeds)
     - Global Tables for multi-region
     - Cost: \~{}\$500/month (variable)

  > ElastiCache Redis (Real-time messaging, sessions)
     - cache.r5.large cluster
     - Cost: \~{}\$200/month

  > DocumentDB (Content metadata, analytics)
      - db.r5.large: \$0.29/hour
      - Cost: \~{}\$212/month
```

Total: ~$1,166/month for millions of users

Database Cost Comparison Examples

Scenario: MySQL Database for Production Application

Workload	AWS Service	Configuration	Monthly Co
Small (¡ 100 GB)	RDS MySQL	db.t3.medium, 100 GB	$75
Small Serverless	Aurora Serverless v2	0.5-1 ACU	$45-90
Medium (500 GB)	RDS MySQL Multi-AZ	db.m5.large, 500 GB	$385
Medium	Aurora MySQL	db.r5.large, 500 GB	$430
Large (2 TB)	Aurora MySQL	db.r5.2xlarge + 5 read replicas	$1,850
On-premises equivalent	Self-managed	3 servers, licenses, staff	$3,000+

Scenario: NoSQL Database - DynamoDB vs Self-Managed

Metric	DynamoDB	Self-Managed Cassandr
Setup Time	Minutes	Days to weeks
Management	Fully managed	You manage everything
Scaling	Automatic, instant	Manual cluster manageme
Performance	Single-digit milliseconds	Tuning required
High Availability	Built-in, multi-AZ	Must configure
Cost (10GB, 1M reads/writes per day)	˜$25/month	˜$150/month (instances +
Cost (1TB, 100M reads/writes per day)	˜$1,250/month	˜$1,500/month
Staff Required	None	1-2 DBAs

Database Migration Scenarios

Scenario 1: Oracle to Aurora PostgreSQL

Current State: On-premises Oracle RAC

- Database size: 2 TB
- Monthly license cost: $5,000
- Hardware and maintenance: $3,000/month
- **Total**: $8,000/month

Migration Strategy:

1. **Assessment** (Week 1):

- Use AWS Schema Conversion Tool (SCT)
- Identify incompatible schema objects
- Estimate conversion effort

1. **Schema Conversion** (Weeks 2-3):

- Convert DDL using SCT

- Modify incompatible SQL
- Test converted schema

1. **Data Migration** (Week 4):

- Use AWS DMS for initial load
- Setup ongoing replication
- Validate data integrity

1. **Cutover** (Week 5):

- Stop writes to source
- Final sync with DMS
- Switch application to Aurora
- Monitor performance

After Migration:

- Aurora PostgreSQL: db.r5.2xlarge Multi-AZ
- Monthly cost: $950
- **Savings**: $7,050/month (88% reduction)
- **ROI**: Migration cost recovered in 2 months

Scenario 2: Self-Managed MongoDB to DocumentDB
Current State: MongoDB on EC2

- 3 r5.xlarge instances: $438/month
- Management overhead: 20 hours/month
- Backup storage: $100/month
- **Total**: $538/month + management time

Migration Process:

1. Create DocumentDB cluster
2. Use mongodump/mongorestore
3. Test application compatibility
4. Switch connection strings
5. Decommission EC2 instances

After Migration:

- DocumentDB: db.r5.large Multi-AZ
- Monthly cost: $424/month
- **Savings**: $114/month + management time
- **Benefits**: Automated backups, patching, monitoring

Common Database Configuration Mistakes

RDS/Aurora:

1. Not enabling automated backups
2. Single-AZ for production databases
3. Not using read replicas for read-heavy workloads
4. Over-provisioning instance size
5. Not enabling encryption at rest
6. Public accessibility enabled
7. Not monitoring slow query logs
8. Inadequate maintenance window planning

DynamoDB:

1. Not using on-demand for variable workloads
2. Over-provisioning read/write capacity
3. Poor partition key design (hot partitions)
4. Not using DAX for read-heavy workloads
5. Not implementing TTL for temporary data
6. Missing Global Secondary Indexes
7. Not using DynamoDB Streams for change capture
8. Storing large items (¿ 400 KB inefficient)

ElastiCache:

1. Not implementing proper key expiration
2. Single-node for production (no HA)
3. Wrong cache eviction policy
4. Not monitoring cache hit ratio
5. Storing too much data per key
6. Not using cluster mode for Redis
7. Connection pooling not implemented
8. Cache stampede not handled

Database Service Limits

RDS:

- DB instances per Region: 40 (can be increased)
- Max storage: 64 TB (SQL Server), 128 TB (others)
- Read replicas: 5 per primary (15 for Aurora)
- Automated backup retention: 35 days max
- Manual snapshots: 100 per Region (can be increased)

DynamoDB:

- Table size: Unlimited
- Item size: 400 KB max
- Partition throughput: 3,000 RCU / 1,000 WCU per partition
- Global Secondary Indexes: 20 per table
- Local Secondary Indexes: 5 per table
- Tables per Region: 2,500 (soft limit)

ElastiCache:

- Nodes per Region: 300 (soft limit)
- Nodes per cluster: 90 (Redis), 40 (Memcached)
- Parameter groups: 150 per Region
- Subnet groups: 150 per Region

Database Monitoring and Troubleshooting

RDS/Aurora Monitoring:

1. **CloudWatch Metrics**:

- DatabaseConnections
- CPUUtilization
- FreeableMemory
- ReadLatency / WriteLatency
- ReadIOPS / WriteIOPS

1. **Performance Insights**:

- Top SQL queries by load

- Wait events analysis
- Database load visualization
- Cost: Free for 7 days retention, paid for longer

1. **Enhanced Monitoring**:

- OS-level metrics (50+ metrics)
- Real-time monitoring (1-second granularity)
- Process and thread information

DynamoDB Monitoring:

1. **CloudWatch Metrics**:

- ConsumedReadCapacityUnits
- ConsumedWriteCapacityUnits
- UserErrors / SystemErrors
- ThrottledRequests
- ConditionalCheckFailedRequests

1. **DynamoDB Contributor Insights**:

- Most accessed items
- Throttled requests
- Hot partitions identification

1. **Common Issues**:

- Hot partitions: Redesign partition key
- Throttling: Increase capacity or use on-demand
- Large items: Break into smaller items
- High costs: Review capacity settings and use on-demand

Database Best Practices

General Practices:

1. **Security**:

- Enable encryption at rest and in transit
- Use IAM database authentication
- Store credentials in Secrets Manager
- Apply least privilege access
- Regular security patching

1. **High Availability**:

- Multi-AZ deployments for production
- Regular backup testing
- Automated failover testing
- Cross-Region replication for DR

1. **Performance**:

- Right-size instances based on metrics
- Use read replicas for read scaling
- Implement connection pooling
- Monitor slow queries
- Regular index maintenance

1. **Cost Optimization**:

- Use Reserved Instances for steady workloads
- Right-size instances (avoid over-provisioning)
- Delete old snapshots
- Use Aurora Serverless for variable workloads
- Implement data lifecycle policies

—

4.0.6 Networking and Content Delivery

Amazon VPC (Virtual Private Cloud)

Isolated virtual network in AWS

VPC allows you to provision a logically isolated section of the AWS Cloud where you can launch AWS resources in a virtual network you define.

Core Concepts:

- **Your Private Network**: Define your own IP address range
- **CIDR Blocks**: Define IP address range (e.g., 10.0.0.0/16)
- **Subnets**: Divide VPC into subnets across Availability Zones
- **Route Tables**: Control traffic routing
- **Isolation**: Complete control over networking environment

VPC Components Subnets

Segments of your VPC's IP address range where you launch AWS resources.

- **Public Subnet**:
- Has a route to an Internet Gateway
- Instances can have public IP addresses
- Accessible from the internet
- **Private Subnet**:
- No direct route to the internet
- Instances typically use private IP addresses only
- Access internet via NAT Gateway

Internet Gateway (IGW)

- Horizontally scaled, redundant, highly available
- Allows communication between VPC and the internet
- Supports IPv4 and IPv6
- One IGW per VPC

NAT Gateway

- Enables instances in private subnet to access the internet
- Prevents internet from initiating connections to private instances
- Managed by AWS (highly available within AZ)

- Alternative: NAT Instance (EC2 instance, customer-managed)

Route Tables

- Control where network traffic is directed
- Each subnet must be associated with a route table
- Determines routing for traffic leaving the subnet

Security Security Groups
Virtual firewall for EC2 instances (instance-level).

- **Level**: Instance level (first layer of defense)
- **Rules**: Allow rules only (no deny rules)
- **Statefulness**: Stateful - return traffic automatically allowed
- **Default**: All inbound traffic denied, all outbound allowed
- **Example**: Allow HTTP (port 80) from anywhere, SSH (port 22) from specific IP

Network ACLs (Access Control Lists)
Firewall for subnets (subnet-level).

- **Level**: Subnet level (second layer of defense)
- **Rules**: Both allow and deny rules
- **Statefulness**: Stateless - must explicitly allow return traffic
- **Evaluation**: Rules processed in order (numbered)
- **Default**: Default NACL allows all inbound/outbound traffic

Comparison:

Feature	Security Groups	Network ACLs
Level	Instance	Subnet
State	Stateful	Stateless
Rules	Allow only	Allow and Deny
Rule Processing	All rules evaluated	Rules in number order
Applies To	Instances	All instances in subnet

Connectivity VPC Peering

- Connect two VPCs privately
- Route traffic using private IP addresses
- Can peer VPCs across accounts and Regions
- Non-transitive (no chaining)

VPN Gateway

- Connect on-premises network to VPC via VPN
- Encrypted connection over the internet
- Site-to-Site VPN

Direct Connect

- Dedicated private connection from on-premises to AWS
- Bypass the public internet
- Covered in detail below

Amazon CloudFront

Global Content Delivery Network (CDN)
Deliver content to users with low latency and high transfer speeds.
Key Features:

- **Edge Locations**: 400+ globally distributed edge locations
- **Caching**: Cache content closer to end users
- **Origin Support**: S3, EC2, ELB, on-premises servers
- **Performance**: Reduced latency for global users
- **Security**: Integration with AWS Shield (DDoS protection) and AWS WAF

Use Cases:

- Static content delivery (images, CSS, JavaScript)
- Dynamic content delivery
- Video streaming (live and on-demand)
- API acceleration
- Software distribution

How It Works:

1. User requests content

2. Request routed to nearest edge location
3. CloudFront checks cache
4. If cached, content delivered immediately
5. If not cached, CloudFront retrieves from origin, caches, and delivers

Benefits:

- Improved performance
- Cost reduction (reduced origin load)
- Global reach
- Enhanced security

Amazon Route 53

Highly available and scalable DNS web service

Core Functions:

1. **Domain Registration**: Register domain names
2. **DNS Routing**: Route internet traffic to resources
3. **Health Checking**: Monitor resource health and route traffic accordingly

Routing Policies:

Policy	Description	Use Case
Simple	Single resource	One web server
Weighted	Distribute traffic by percentage	A/B testing, gradual migration
Latency	Route to lowest latency endpoint	Global applications
Failover	Active-passive failover	Disaster recovery
Geolocation	Route based on user's location	Content localization, compliance
Geoproximity	Route based on resource and user location	Bias traffic to specific locations
Multi-value	Return multiple IP addresses	Simple load balancing with health che

Features:

- 100% availability SLA
- Global anycast network
- DNSSEC for domain security
- Integration with AWS services
- Traffic flow for complex routing

Elastic Load Balancing (ELB)

Automatically distribute incoming traffic across multiple targets
Load balancers improve application availability and fault tolerance.

Load Balancer Types Application Load Balancer (ALB)
Layer 7 load balancer for HTTP/HTTPS traffic.

- **OSI Layer**: Layer 7 (Application)
- **Protocols**: HTTP, HTTPS, gRPC
- **Routing**: Path-based, host-based, HTTP header-based
- **Targets**: EC2 instances, containers, IP addresses, Lambda functions
- **Features**:
- SSL/TLS termination
- WebSocket support
- HTTP/2 support
- Sticky sessions
- Authentication (OIDC, SAML)
- **Best For**: Web applications, microservices, container-based applications

Network Load Balancer (NLB)
Layer 4 load balancer for TCP/UDP traffic.

- **OSI Layer**: Layer 4 (Transport)
- **Protocols**: TCP, UDP, TLS
- **Performance**: Millions of requests per second, ultra-low latency
- **Static IP**: Static IP addresses per AZ
- **Targets**: EC2 instances, containers, IP addresses
- **Features**:
- Extreme performance
- Static/Elastic IP addresses
- TLS termination
- Preserve source IP
- **Best For**: High performance, low latency requirements, TCP/UDP applications

Gateway Load Balancer
Layer 3 load balancer for virtual appliances.

- **OSI Layer**: Layer 3 (Network)
- **Protocol**: IP
- **Use Case**: Deploy, scale, and manage third-party virtual appliances
- **Examples**: Firewalls, intrusion detection systems, deep packet inspection
- **Features**:
- GENEVE protocol support
- High availability for appliances
- Centralized security management

Classic Load Balancer (CLB)
Previous generation load balancer.

- **Status**: Being phased out
- **OSI Layer**: Layer 4 and Layer 7
- **Recommendation**: Use ALB or NLB for new applications

Choosing a Load Balancer:

- **ALB**: HTTP/HTTPS applications, microservices
- **NLB**: TCP/UDP applications, extreme performance
- **GWLB**: Third-party virtual appliances

AWS Direct Connect

Dedicated private network connection
Establish a dedicated private connection from on-premises to AWS.
Key Features:

- **Connection Type**: Private, dedicated network connection
- **Bandwidth**: 1 Gbps, 10 Gbps, or 100 Gbps
- **Bypass Internet**: Traffic doesn't traverse the public internet
- **Performance**: Consistent network performance
- **Access**: Connect to both public AWS services and VPCs
- **Cost**: Reduced bandwidth costs for large data transfer

Benefits:

- Predictable network performance
- Reduced bandwidth costs
- Consistent latency
- Enhanced security (private connection)
- Access to public and private AWS resources

Setup Time: Weeks to months to provision
Use Cases:

- Large data transfers
- Real-time data feeds
- Hybrid cloud architectures
- Accessing VPCs across multiple Regions

AWS VPN

Encrypted connection over the internet

Site-to-Site VPN Connect on-premises network to AWS VPC.

- **Connection**: Encrypted IPsec VPN tunnel over the internet
- **Components**: Customer Gateway (on-premises) + Virtual Private Gateway (AWS)
- **Setup Time**: Minutes to hours
- **Cost**: Lower cost than Direct Connect
- **Bandwidth**: Limited by internet connection

Client VPN Connect individual users to AWS or on-premises networks.

- **Use Case**: Remote user access
- **Connection**: Encrypted TLS VPN connection
- **Access**: AWS VPC resources and on-premises resources

VPN vs Direct Connect:

Feature	Site-to-Site VPN	Direct Connect
Connection	Over internet	Private dedicated line
Setup Time	Minutes	Weeks/months
Cost	Lower	Higher
Bandwidth	Internet-dependent	1-100 Gbps
Security	Encrypted	Private (optionally encrypted)

Performance	Variable	Consistent

—

4.0.7 Management and Governance

AWS CloudFormation

Infrastructure as Code (IaC)
Define and provision AWS infrastructure using template files.
Key Concepts:

- **Templates**: JSON or YAML files defining resources
- **Stacks**: Collection of AWS resources managed as a single unit
- **Change Sets**: Preview changes before applying

Features:

- **Automation**: Automate infrastructure provisioning
- **Version Control**: Track infrastructure changes over time
- **Consistency**: Ensure consistent deployments across environments
- **Reusability**: Reuse templates across accounts and Regions
- **Rollback**: Automatic rollback on errors
- **Cost**: No additional charge (pay for resources created)

Use Cases:

- Disaster recovery
- Environment replication (dev, test, prod)
- Compliance and governance
- Infrastructure version control

Benefits:

- Faster provisioning
- Reduced errors
- Consistent environments
- Easier management of complex architectures

AWS CloudTrail

Governance, compliance, and auditing

Log and monitor all API activity in your AWS account.

Key Features:

- **API Logging**: Records all API calls in your account
- **Who, What, When, Where**: Comprehensive audit trail
- **Enabled by Default**: 90-day event history automatically
- **Long-term Storage**: Store logs in S3 for retention beyond 90 days
- **Integration**: CloudWatch Logs for monitoring and alerting

Event Types:

- **Management Events**: Control plane operations (create instance, modify security group)
- **Data Events**: Data plane operations (S3 object access, Lambda function invocations)
- **Insights Events**: Unusual API activity detection

Use Cases:

- Security analysis and compliance auditing
- Troubleshooting operational issues
- Change tracking
- Incident investigation
- Compliance requirements

Benefits:

- Complete visibility into account activity
- Simplified compliance auditing
- Security analysis and incident response
- Operational troubleshooting

Amazon CloudWatch

Monitoring and observability service

Collect and track metrics, collect and monitor logs, set alarms, and automatically react to changes.

CloudWatch Components CloudWatch Metrics

- Numerical time-series data points
- Built-in metrics for most AWS services (CPU, network, disk)
- Custom metrics for application-specific data
- Retention: Up to 15 months

CloudWatch Logs

- Collect and store log files from resources
- Monitor and troubleshoot systems and applications
- Log Groups and Log Streams organization
- Retention policies (never expire to 1 day)

CloudWatch Alarms

- Trigger actions based on metric thresholds
- States: OK, ALARM, INSUFFICIENT_DATA
- Actions: SNS notifications, Auto Scaling, EC2 actions
- Composite alarms for complex conditions

CloudWatch Events / EventBridge

- Event-driven automation
- Respond to state changes in AWS resources
- Schedule automated actions (cron jobs)
- Integration with Lambda, SQS, SNS, and more

CloudWatch Dashboards

- Customizable home pages for monitoring
- Visualize metrics and logs
- Share across teams

Use Cases:

- Application monitoring
- Performance optimization
- Troubleshooting
- Capacity planning
- Automated responses to operational changes

AWS Systems Manager

Operational hub for AWS resources
Centralized operations management for AWS and on-premises resources.
Key Capabilities:
Operations Management

- **Session Manager**: Secure shell access without SSH keys or bastion hosts
- **Run Command**: Execute commands on multiple instances
- **Patch Manager**: Automate OS and application patching
- **Maintenance Windows**: Schedule operations tasks

Configuration Management

- **Parameter Store**: Centralized storage for configuration data and secrets
- **State Manager**: Maintain consistent configuration
- **Inventory**: Collect metadata from managed instances

Insights

- **OpsCenter**: Centralized operational issues management
- **CloudWatch Dashboard Integration**: Unified view

Benefits:

- Reduced operational overhead
- Improved security (no SSH keys needed)
- Centralized management
- Automated operations

Common Use Cases:

- Patch management across fleet
- Securely access instances
- Store application secrets
- Automate operational tasks

AWS Trusted Advisor

Best practice recommendations
Real-time guidance to help provision resources following AWS best practices.

Five Pillars of Recommendations 1. Cost Optimization

- Identify unused or underutilized resources
- Recommendations to reduce costs
- Examples: Idle RDS instances, unattached EBS volumes, unused Elastic IPs

2. Performance

- Improve service performance
- Examples: High utilization of EC2 instances, EBS throughput optimization

3. Security

- Close security gaps
- Examples: S3 bucket permissions, security group rules, MFA on root account

4. Fault Tolerance

- Increase availability and redundancy
- Examples: Multi-AZ deployments, EBS snapshot age, Route 53 health checks

5. Service Limits (Service Quotas)

- Check usage against service limits
- Avoid service disruptions from hitting limits

Access Levels Free (All Customers)

- 7 core checks:
- S3 bucket permissions
- Security groups - specific ports unrestricted
- IAM use
- MFA on root account
- EBS public snapshots
- RDS public snapshots
- Service limits for common services

Business or Enterprise Support

- All checks (50+ checks)
- Automated notifications
- AWS Support API access
- Programmatic access

Dashboard:

- Color-coded status: Red (action recommended), Yellow (investigation recommended), Green (no problem)
- Downloadable reports

AWS Control Tower

Set up and govern multi-account AWS environment
Automated setup and governance for multi-account AWS environments.
Key Features:

- **Landing Zone**: Automated multi-account setup based on best practices
- **Guardrails**: Governance rules for compliance
- **Preventive**: Prevent policy violations (using SCPs)
- **Detective**: Detect policy violations (using AWS Config)
- **Account Factory**: Automated account provisioning and configuration
- **Dashboard**: Centralized visibility and compliance monitoring

Built On:

- AWS Organizations
- AWS Service Catalog
- AWS CloudFormation
- AWS IAM Identity Center (SSO)
- AWS Config

Use Cases:

- Setting up new multi-account environments
- Governance for enterprise organizations
- Compliance requirements
- Account standardization

AWS Service Catalog

Create and manage catalogs of approved IT services
Enable centralized management of commonly deployed IT services.
Key Features:

- **Product Portfolio**: Catalog of approved CloudFormation templates
- **Access Control**: Control who can deploy which services
- **Versioning**: Manage product versions
- **Constraints**: Define rules for product usage
- **Self-Service**: End users deploy approved resources

Benefits:

- Standardized deployments
- Centralized management
- Governance and compliance
- Cost control
- Faster time to market

Use Cases:

- Standardize infrastructure deployments
- Control cloud resource sprawl
- Enable self-service for developers
- Maintain compliance

—

4.0.8 Additional Services

Amazon SNS (Simple Notification Service)

Pub/sub messaging service
Fully managed messaging service for application-to-application (A2A) and application-to-person (A2P) communication.

Key Concepts:

- **Topics**: Communication channels
- **Publishers**: Send messages to topics
- **Subscribers**: Receive messages from topics
- **Message Filtering**: Subscribers receive only relevant messages

Supported Protocols:

- HTTP/HTTPS
- Email/Email-JSON
- SMS
- Amazon SQS
- AWS Lambda
- Mobile push notifications

Use Cases:

- Application alerts and notifications

- Push notifications to mobile devices
- Email and SMS messaging
- Fanout pattern (one message to many subscribers)
- Decoupled microservices

Benefits:

- Simple setup and operation
- High throughput
- Message durability and delivery
- Message filtering

Amazon SQS (Simple Queue Service)

Fully managed message queue service
Decouple and scale microservices, distributed systems, and serverless applications.
Queue Types:
Standard Queues

- **Throughput**: Unlimited
- **Delivery**: At-least-once delivery (messages may be delivered multiple times)
- **Ordering**: Best-effort ordering
- **Use Case**: High throughput applications where occasional duplicates are acceptable

FIFO Queues

- **Throughput**: Up to 3,000 messages per second (higher with batching)
- **Delivery**: Exactly-once processing
- **Ordering**: Strict ordering guaranteed
- **Use Case**: When message order is critical

Features:

- **Retention**: Messages retained up to 14 days
- **Visibility Timeout**: Hide message while being processed
- **Dead-Letter Queues**: Handle failed messages
- **Delay Queues**: Postpone delivery of messages
- **Long Polling**: Reduce empty responses and costs

Use Cases:

- Decouple application components
- Buffer between producers and consumers
- Batch processing
- Asynchronous processing

SNS vs SQS:

- **SNS**: Push notifications, pub/sub, fanout
- **SQS**: Pull-based queue, decouple components, buffer
- **Together**: SNS sends to multiple SQS queues for fanout architecture

AWS Step Functions

Serverless workflow orchestration

Coordinate multiple AWS services into serverless workflows.

Key Features:

- **Visual Workflows**: Graphical workflow designer
- **State Machines**: Define workflows as state machines
- **Service Integration**: Direct integration with AWS services
- **Error Handling**: Built-in retry and error handling
- **Standard and Express Workflows**:
- **Standard**: Long-running (up to 1 year), exactly-once execution
- **Express**: High-volume, short duration (up to 5 minutes), at-least-once execution

Use Cases:

- Orchestrate microservices
- Data processing pipelines
- Automate IT and security processes
- Build complex workflows from Lambda functions

Benefits:

- Visual workflow management
- Built-in error handling
- Serverless and scalable
- Audit trail of executions

Amazon EventBridge

Serverless event bus service

Connect applications using events from AWS services, SaaS applications, and custom applications.

Key Concepts:

- **Events**: State changes or notifications
- **Event Buses**: Receive and route events
- **Rules**: Match events and route to targets
- **Targets**: Destinations for events (Lambda, SQS, SNS, etc.)

Event Sources:

- AWS services (EC2, S3, etc.)
- SaaS applications (Salesforce, Datadog, etc.)
- Custom applications

Features:

- **Schema Registry**: Discover and manage event schemas
- **Archive and Replay**: Store and replay events
- **Cross-Account Events**: Send events across AWS accounts
- **Filtering**: Advanced event pattern matching

Previously: CloudWatch Events (EventBridge is the enhanced version)

Use Cases:

- React to state changes in AWS services
- Schedule automated actions
- Integrate SaaS applications
- Event-driven architectures

AWS Batch

Fully managed batch processing

Run batch computing workloads at any scale.

Key Features:

- **Automatic Provisioning**: Dynamically provisions compute resources
- **Job Queues**: Queue and prioritize jobs
- **Job Definitions**: Define how jobs are run

- **Scaling**: Automatically scales based on job volume
- **Integration**: Uses EC2 and Spot Instances

Use Cases:

- Data processing and analysis
- Image and video rendering
- Financial risk modeling
- Scientific simulations
- ETL (Extract, Transform, Load) jobs

Benefits:

- No infrastructure management
- Cost optimization with Spot Instances
- Automatic scaling
- Job dependency management

Amazon Athena

Interactive query service for S3

Analyze data in Amazon S3 using standard SQL.

Key Features:

- **Serverless**: No infrastructure to manage
- **SQL Queries**: Standard SQL support
- **Pay Per Query**: Charged based on data scanned
- **Integration**: Works with AWS Glue Data Catalog
- **Formats**: Supports CSV, JSON, ORC, Avro, Parquet

Use Cases:

- Ad-hoc data analysis
- Log analysis
- Business intelligence
- Query CloudTrail logs
- Analyze application logs

Benefits:

- No ETL required

- Fast query performance
- Cost-effective (pay only for queries)
- Easy to use (just SQL)

Best Practices:

- Use columnar formats (Parquet, ORC) for better performance
- Partition data to reduce scanned data
- Compress data to reduce costs

AWS Glue

Fully managed ETL (Extract, Transform, Load) service
Prepare data for analytics and machine learning.

Key Components:

AWS Glue Data Catalog

- Centralized metadata repository
- Stores table definitions, schemas, and metadata
- Integration point for Athena, EMR, Redshift Spectrum

AWS Glue Crawlers

- Automatically discover and catalog data
- Infer schemas
- Update the Data Catalog

AWS Glue ETL Jobs

- Serverless ETL operations
- Generate code in Python or Scala
- Visual ETL editor
- Built-in transformations

Use Cases:

- Prepare data for analytics
- Data lake management
- Database migration
- Data pipeline automation

Benefits:

- Serverless (no infrastructure management)
- Automatic schema discovery
- Integrated with AWS analytics services
- Cost-effective

Amazon QuickSight

Business intelligence (BI) service
Create visualizations, perform ad-hoc analysis, and get business insights.
Key Features:

- **Serverless**: No servers to manage
- **Visualizations**: Interactive dashboards and visualizations
- **ML Insights**: Automatic insights powered by machine learning
- **SPICE Engine**: In-memory calculation engine for fast performance
- **Sharing**: Share dashboards with users and groups
- **Embedded Analytics**: Embed in applications

Data Sources:

- AWS services (RDS, Aurora, Redshift, S3, Athena)
- On-premises databases
- SaaS applications (Salesforce, Jira)
- File uploads (Excel, CSV, JSON)

Pricing:

- Pay per session (users pay only when accessing dashboards)
- Author and Reader pricing tiers

Use Cases:

- Business intelligence and reporting
- Data visualization
- Embedded analytics
- Ad-hoc analysis

Amazon Kinesis

Real-time data streaming
Collect, process, and analyze real-time streaming data.

Kinesis Services Kinesis Data Streams

- Capture and store data streams in real-time
- Shards for parallel processing
- Retain data for 1-365 days
- Use Case: Custom real-time applications

Kinesis Data Firehose

- Load streaming data into data stores
- Destinations: S3, Redshift, Elasticsearch, Splunk, HTTP endpoints
- Near real-time (60 seconds minimum)
- Automatic scaling
- Use Case: Simple data ingestion pipeline

Kinesis Data Analytics

- Analyze streaming data with SQL or Apache Flink
- Real-time analytics and insights
- Use Case: Real-time dashboards, metrics, anomaly detection

Kinesis Video Streams

- Capture and store video streams
- Process and analyze video
- Use Case: Video analytics, machine learning on video

Common Use Cases:

- Real-time log and event data collection
- IoT device telemetry
- Clickstream analysis
- Real-time analytics
- Video processing and analysis

Amazon SageMaker

Build, train, and deploy machine learning models

Fully managed service for the complete machine learning workflow.

Key Capabilities:

Build

- Integrated Jupyter notebooks
- Pre-built algorithms and frameworks
- Data labeling service (Ground Truth)
- Feature Store

Train

- One-click training
- Automatic model tuning (hyperparameter optimization)
- Distributed training
- Managed Spot training for cost savings

Deploy

- One-click deployment
- Real-time and batch inference
- Multi-model endpoints
- Auto-scaling

Additional Features:

- SageMaker Studio (ML IDE)
- Model monitoring and debugging
- MLOps capabilities
- Pre-built ML solutions

Use Cases:

- Predictive analytics
- Recommendation systems
- Fraud detection
- Image and text classification
- Time series forecasting

Amazon Rekognition

Image and video analysis
Add image and video analysis to applications using deep learning.
Capabilities:

- **Object and Scene Detection**: Identify thousands of objects and scenes
- **Facial Analysis**: Detect faces and analyze attributes (age, gender, emotions)
- **Face Comparison**: Compare faces for verification
- **Face Recognition**: Identify known faces in images and videos
- **Celebrity Recognition**: Recognize thousands of celebrities
- **Text Detection**: Detect and extract text from images
- **Content Moderation**: Detect inappropriate content
- **Video Analysis**: Track people, objects, activities in videos

Use Cases:

- User verification
- Content moderation
- Searchable image library
- Sentiment analysis
- Security and surveillance
- Media analysis

Amazon Comprehend

Natural Language Processing (NLP) service
Extract insights and relationships from text using machine learning.
Capabilities:

- **Sentiment Analysis**: Determine positive, negative, neutral, or mixed sentiment
- **Entity Recognition**: Identify people, places, brands, events, etc.
- **Key Phrase Extraction**: Extract important phrases from text
- **Language Detection**: Identify the dominant language
- **Topic Modeling**: Organize documents by topic
- **Custom Classification**: Train custom models for specific domains

Use Cases:

- Customer feedback analysis

- Document classification
- Social media monitoring
- Knowledge management
- Business intelligence from documents

Amazon Lex

Build conversational interfaces (chatbots)
Build chatbots and voice assistants using the same technology as Amazon Alexa.
Key Features:

- **Automatic Speech Recognition (ASR)**: Convert speech to text
- **Natural Language Understanding (NLU)**: Understand intent
- **Multi-turn Conversations**: Context management across conversation
- **8 kHz Telephony Audio**: Support for phone calls
- **Integration**: Connect to AWS Lambda, mobile apps, messaging platforms

Use Cases:

- Customer service chatbots
- Virtual assistants
- Voice-enabled applications
- Interactive Voice Response (IVR) systems
- Information bots

Components:

- **Intents**: Actions users want to perform
- **Utterances**: Phrases users might say
- **Slots**: Parameters for intents
- **Fulfillment**: Lambda function to fulfill the intent

AWS Migration Hub

Track application migrations

Centralized location to track progress of application migrations across multiple AWS and partner solutions.

Key Features:

- **Single Dashboard**: View migration status across tools
- **Migration Tracking**: Track server and database migrations
- **Integration**: Works with AWS migration tools and partner tools
- **Application Grouping**: Group servers by application

Integrated Tools:

- AWS Application Migration Service
- AWS Database Migration Service
- CloudEndure Migration
- ATADATA ATAmotion
- RiverMeadow Server Migration

Benefits:

- Centralized visibility
- Track progress across multiple tools
- Identify and troubleshoot issues
- Plan and execute migrations

AWS Database Migration Service (DMS)

Migrate databases to AWS

Migrate databases to AWS quickly and securely while the source database remains operational.

Key Features:

- **Minimal Downtime**: Source database remains operational during migration
- **Continuous Data Replication**: Ongoing replication after initial migration
- **Homogeneous Migrations**: Same database engine (Oracle to Oracle)
- **Heterogeneous Migrations**: Different database engines (Oracle to Aurora)
- **Schema Conversion**: AWS Schema Conversion Tool (SCT) for heterogeneous migrations

Supported Sources:

- Oracle, SQL Server, MySQL, PostgreSQL, MongoDB, SAP, DB2
- On-premises and cloud databases

Supported Targets:

- Amazon RDS, Aurora, Redshift, DynamoDB, S3, DocumentDB

Use Cases:

- Migrate to cloud
- Replicate for development/test
- Database consolidation
- Continuous data replication

Benefits:

- Cost-effective
- Minimal downtime
- Supports most databases
- Reliable and self-healing

AWS Application Discovery Service

Discover on-premises applications for migration planning
Collect information about on-premises data centers to plan migrations.
Discovery Methods:
Agentless Discovery (Application Discovery Service Agentless Collector)

- VMware environment only
- Collects VM inventory, configuration, performance
- No agent installation required

Agent-based Discovery (Application Discovery Agent)

- Install agent on servers
- Collects system configuration, performance, running processes, network connections
- Works on physical and virtual servers

Data Collected:

- Server specifications (CPU, memory, disk)
- Resource utilization
- Network dependencies

- Running processes and applications

Integration:

- Data exported to Amazon S3
- Integrate with AWS Migration Hub
- Visualize dependencies with AWS Application Discovery Service

Use Cases:

- Migration planning
- Identify dependencies between applications
- Right-size target infrastructure
- Total cost of ownership (TCO) analysis

—

4.0.9 Review Questions

Test your knowledge of AWS Cloud Technology and Services:

Question 1

Which AWS service provides a serverless compute platform that automatically scales and charges based on the number of requests and compute time?

A) Amazon EC2 B) AWS Lambda C) Amazon Lightsail D) AWS Elastic Beanstalk ¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) AWS Lambda

Lambda is serverless, automatically scales, and charges based on requests and compute time. EC2 requires instance management, Lightsail has predictable pricing, and Elastic Beanstalk is a PaaS that still provisions underlying resources. ¡/details¿

—

Question 2

Your company needs to store frequently accessed files that must be available immediately. Which S3 storage class would be the MOST cost-effective?

A) S3 Glacier Deep Archive B) S3 Standard-IA C) S3 Standard D) S3 One Zone-IA ¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) S3 Standard

For frequently accessed data requiring immediate availability, S3 Standard is the appropriate choice. Glacier Deep Archive has long retrieval times, and IA (Infrequent Access) classes charge retrieval fees that would add up with frequent access. ¡/details¿

—

Question 3

Which EC2 pricing model would be MOST appropriate for a production database that must run continuously for the next three years?

A) On-Demand Instances B) Spot Instances C) Reserved Instances D) Dedicated Hosts

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) Reserved Instances

For steady-state workloads running continuously, Reserved Instances provide the best cost savings (up to 75%). On-Demand is most expensive, Spot Instances can be interrupted (unsuitable for databases), and Dedicated Hosts are for specific compliance/licensing needs. ¡/details¿

—

Question 4

What is the PRIMARY difference between Security Groups and Network ACLs?

A) Security Groups are stateful, Network ACLs are stateless B) Security Groups apply to subnets, Network ACLs apply to instances C) Security Groups allow deny rules, Network ACLs only allow rules D) Security Groups are free, Network ACLs have additional costs

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: A) Security Groups are stateful, Network ACLs are stateless

Security Groups are stateful (return traffic automatically allowed), while Network ACLs are stateless (must explicitly allow return traffic). Security Groups apply to instances, NACLs apply to subnets. Security Groups only have allow rules, NACLs have both allow and deny rules. Both are free. ¡/details¿

—

Question 5

Which database service would be BEST for a social networking application that needs to efficiently query relationships between users?

A) Amazon RDS B) Amazon DynamoDB C) Amazon Neptune D) Amazon Redshift

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) Amazon Neptune

Neptune is a graph database designed for highly connected data and relationship queries (social networks, recommendation engines). RDS is for relational data, DynamoDB for key-value data, and Redshift for analytics/data warehousing. ¡/details¿

—

Question 6

Your application needs to send notifications to multiple subscribers via email, SMS, and HTTP endpoints. Which service should you use?

A) Amazon SQS B) Amazon SNS C) AWS Step Functions D) Amazon EventBridge

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Amazon SNS

SNS is a pub/sub messaging service that can send notifications to multiple subscribers across multiple protocols (email, SMS, HTTP). SQS is a message queue, Step Functions orchestrates workflows, and EventBridge is for event-driven architectures. ¡/details¿

—

Question 7

Which AWS service allows you to define infrastructure as code using JSON or YAML templates?

A) AWS CloudTrail B) AWS CloudFormation C) AWS Config D) AWS Systems Manager

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) AWS CloudFormation

CloudFormation uses templates (JSON/YAML) to define infrastructure as code. CloudTrail logs API calls, Config tracks resource configurations, and Systems Manager manages operations. ¡/details¿

—

Question 8

What is the MINIMUM number of Availability Zones in an AWS Region?

A) 1 B) 2 C) 3 D) 4

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) 3

All AWS Regions have a minimum of 3 Availability Zones to provide high availability and fault tolerance. ¡/details¿

—

Question 9

Which AWS service provides a private, dedicated network connection from your on-premises data center to AWS?

A) AWS VPN B) AWS Direct Connect C) Amazon CloudFront D) AWS Transit Gateway

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) AWS Direct Connect

Direct Connect provides a dedicated private network connection. VPN uses encrypted connection over the internet, CloudFront is a CDN, and Transit Gateway connects VPCs and on-premises networks. ¡/details¿

—

Question 10

Your company needs to migrate 100 TB of data to AWS. Network bandwidth is limited. Which service should you use?

A) AWS DataSync B) AWS Snowball C) Amazon S3 Transfer Acceleration D) AWS Direct Connect

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) AWS Snowball

For large data migrations with limited bandwidth, Snowball Edge (80 TB capacity) is ideal. You would use 2 devices for 100 TB. DataSync is for ongoing transfers, Transfer Acceleration speeds up S3 uploads over internet, and Direct Connect takes weeks to provision. ¡/details¿

—

Question 11

Which service would you use to analyze data in S3 using standard SQL without moving the data?

A) Amazon Redshift B) Amazon Athena C) AWS Glue D) Amazon EMR

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Amazon Athena

Athena allows you to query data directly in S3 using SQL without loading data into a database. Redshift is a data warehouse (requires loading data), Glue is for ETL, and EMR is for big data processing frameworks. ¡/details¿

—

Question 12

What AWS service provides real-time guidance to help you provision resources following AWS best practices?

A) AWS CloudTrail B) AWS Trusted Advisor C) AWS Inspector D) AWS Config

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) AWS Trusted Advisor

Trusted Advisor provides real-time best practice recommendations across 5 categories. CloudTrail logs API calls, Inspector assesses application security, and Config tracks resource configurations. ¡/details¿

—

Question 13

Which load balancer type operates at Layer 7 and can route traffic based on URL path?

A) Classic Load Balancer B) Network Load Balancer C) Application Load Balancer D) Gateway Load Balancer

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) Application Load Balancer

ALB operates at Layer 7 (application layer) and supports advanced routing including path-based and host-based routing. NLB operates at Layer 4, Gateway Load Balancer at Layer 3, and Classic Load Balancer is being phased out. ¡/details¿

—

Question 14

Which AWS service would you use to coordinate multiple AWS services into a serverless workflow?

A) Amazon SQS B) Amazon SNS C) AWS Step Functions D) AWS Lambda

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) AWS Step Functions

Step Functions orchestrates multiple AWS services into serverless workflows with visual workflow design. SQS is a message queue, SNS is pub/sub messaging, and Lambda executes individual functions. ¡/details¿

—

Question 15

Your application requires block storage for an EC2 instance. Which service should you use?

A) Amazon S3 B) Amazon EBS C) Amazon EFS D) AWS Storage Gateway

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Amazon EBS

EBS provides block storage for EC2 instances. S3 is object storage, EFS is file storage for multiple instances, and Storage Gateway is for hybrid cloud storage. ¡/details¿

—

Question 16

Which DynamoDB capacity mode should you choose for unpredictable, variable workloads?

A) Provisioned Capacity B) Reserved Capacity C) On-Demand D) Spot Capacity

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) On-Demand

On-Demand capacity mode is ideal for unpredictable workloads as you pay per request. Provisioned Capacity requires setting read/write capacity units in advance. Reserved and Spot Capacity don't exist for DynamoDB. ¡/details¿

—

Question 17

What is the purpose of AWS CloudFront Edge Locations?

A) Run EC2 instances closer to users B) Cache content closer to users for faster delivery C) Store backups in multiple locations D) Host databases in multiple regions

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Cache content closer to users for faster delivery

Edge Locations are part of CloudFront CDN and cache content to reduce latency. They don't run EC2 instances, store backups, or host databases. ¡/details¿

—

Question 18

Which AWS service logs all API calls made in your AWS account for auditing and compliance?

A) Amazon CloudWatch B) AWS CloudTrail C) AWS Config D) AWS Inspector

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) AWS CloudTrail

CloudTrail logs all API calls for governance, compliance, and auditing. CloudWatch monitors metrics and logs, Config tracks resource configurations, and Inspector assesses security vulnerabilities. ¡/details¿

—

Question 19

Which storage service is best for shared file storage accessible by multiple EC2 instances simultaneously?

A) Amazon EBS B) Amazon S3 C) Amazon EFS D) Instance Store

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) Amazon EFS

EFS provides shared NFS file system accessible by multiple EC2 instances simultaneously. EBS attaches to single instance, S3 is object storage (not file system), and Instance Store is ephemeral. ¡/details¿

—

Question 20

Your company needs to run Docker containers without managing the underlying infrastructure. Which service combination is MOST appropriate?

A) Amazon ECS with EC2 launch type B) Amazon ECS with Fargate launch type C) Amazon EKS with EC2 nodes D) Amazon EC2 with Docker installed

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Amazon ECS with Fargate launch type

ECS with Fargate is serverless - you don't manage any infrastructure. ECS with EC2 and EKS with EC2 require managing EC2 instances. Running Docker on EC2 requires full infrastructure management. ¡/details¿

—

Question 21

A company needs to process large amounts of data for scientific simulations. The workload runs for 8 hours per day with predictable schedules. Which EC2 pricing model provides the BEST cost optimization?

A) On-Demand Instances with Auto Scaling B) Spot Instances C) Reserved Instances with Scheduled Reserved Instances D) Dedicated Hosts

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) Reserved Instances with Scheduled Reserved Instances

Scheduled Reserved Instances allow you to reserve capacity for predictable recurring schedules (daily, weekly, monthly). Since the workload runs predictably for 8 hours/-day, this provides significant savings compared to On-Demand while ensuring capacity availability. Spot Instances could be interrupted, and Dedicated Hosts are unnecessarily expensive for this use case. ¡/details¿

—

Question 22

Which storage service provides the LOWEST cost for storing 200 TB of data that must be retained for 10 years for compliance but will never be accessed unless required by auditors?

A) S3 Standard B) S3 Glacier Flexible Retrieval C) S3 Glacier Deep Archive D) S3 One Zone-IA

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) S3 Glacier Deep Archive

Glacier Deep Archive offers the lowest storage cost ($0.00099 per GB/month) and is specifically designed for long-term archival storage with retrieval times of 12-48 hours. The 200 TB would cost approximately $200/month compared to $4,600/month in S3 Standard. The long retrieval time is acceptable for compliance data rarely accessed. ¡/details¿

—

Question 23

Your application needs to make millions of cache lookups per second with sub-millisecond latency. Which service should you use?

A) Amazon RDS with read replicas B) Amazon DynamoDB with DAX C) Amazon ElastiCache D) Amazon Aurora

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Amazon DynamoDB with DAX

DynamoDB Accelerator (DAX) provides microsecond latency for millions of requests per second, making it ideal for extreme performance requirements. ElastiCache could also work, but DAX is specifically designed for DynamoDB and provides the best integration. RDS and Aurora have higher latency (milliseconds). ¡/details¿

—

Question 24

A company wants to analyze customer behavior by querying relationships like "customers who bought this also bought that." Which database is MOST appropriate?

A) Amazon RDS B) Amazon DynamoDB C) Amazon Neptune D) Amazon Redshift

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) Amazon Neptune

Neptune is a graph database designed for querying highly connected data and relationships. It excels at queries like friend-of-friend, recommendations, and relationship traversals. RDS and DynamoDB would require complex joins or denormalization, and Redshift is for analytics, not real-time relationship queries. ¡/details¿

—

Question 25

Which combination provides the MOST cost-effective solution for a serverless web application with unpredictable traffic?

A) EC2 Auto Scaling + RDS B) Lambda + DynamoDB On-Demand + S3 C) ECS Fargate + Aurora Serverless D) Lightsail + RDS

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Lambda + DynamoDB On-Demand + S3

This combination provides true serverless scaling with pay-per-use pricing. Lambda charges only for actual executions, DynamoDB On-Demand charges per request, and S3 charges for storage and requests. This is ideal for unpredictable traffic with zero waste during low-traffic periods. Option C is also serverless but more expensive. ¡/details¿

—

Question 26

Your company needs to ensure that EC2 instances in a private subnet can download software updates from the internet without being directly accessible from the internet. What should you configure?

A) Internet Gateway B) NAT Gateway C) Virtual Private Gateway D) VPC Peering

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) NAT Gateway

NAT Gateway enables instances in private subnets to initiate outbound connections to the internet while preventing inbound connections from the internet. An Internet Gateway would require instances to be in public subnets with public IPs, making them directly accessible. ¡/details¿

—

Question 27

Which AWS service automatically distributes incoming application traffic across multiple targets and can route based on URL path?

A) Network Load Balancer B) Application Load Balancer C) Classic Load Balancer D) CloudFront

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Application Load Balancer

ALB operates at Layer 7 and supports content-based routing including path-based, host-based, and HTTP header-based routing. NLB operates at Layer 4 and doesn't inspect application-layer content. CloudFront is a CDN, not a load balancer. ¡/details¿

—

Question 28

A company needs to migrate 500 TB of data to AWS but has limited network bandwidth (10 Mbps). What is the MOST efficient migration method?

A) AWS DataSync over internet B) AWS Direct Connect C) AWS Snowball Edge devices D) S3 Transfer Acceleration

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) AWS Snowball Edge devices

At 10 Mbps, transferring 500 TB would take approximately 463 days. Snowball Edge devices (80 TB capacity each) can transfer this data in weeks. You would order 7 devices, copy data locally, and ship them to AWS. Direct Connect takes weeks to provision, and DataSync/Transfer Acceleration would still be limited by bandwidth. ¡/details¿

—

Question 29

Which RDS feature provides automatic failover to a standby instance in a different Availability Zone?

A) Read Replicas B) Multi-AZ deployment C) Automated Backups D) Manual Snapshots

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Multi-AZ deployment

Multi-AZ provides synchronous replication to a standby instance and automatic failover (typically under 1 minute). Read Replicas use asynchronous replication and require manual promotion. Backups and snapshots don't provide automatic failover. ¡/details¿

—

Question 30

Your application needs to store session data that expires after 24 hours. Which service combination is MOST cost-effective?

A) RDS with custom cleanup scripts B) DynamoDB with Time-To-Live (TTL) C) S3 with Lifecycle policies D) ElastiCache with manual deletion

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) DynamoDB with Time-To-Live (TTL)

DynamoDB TTL automatically deletes expired items at no additional cost, making it ideal for session data. ElastiCache could work but requires configuration and memory management. RDS and S3 are not optimized for high-frequency session data with automatic expiration. ¡/details¿

—

Question 31

Which service would you use to create a visual workflow that coordinates multiple Lambda functions with error handling and retry logic?

A) Amazon SQS B) AWS Step Functions C) Amazon EventBridge D) AWS Batch

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) AWS Step Functions

Step Functions provides visual workflow orchestration with built-in error handling, retries, and state management. It's specifically designed to coordinate multiple AWS services including Lambda. SQS is just a message queue, EventBridge routes events, and Batch is for batch computing. ¡/details¿

—

Question 32

A company wants to analyze application logs stored in S3 using SQL without loading data into a database. Which service should they use?

A) Amazon Redshift B) Amazon Athena C) Amazon EMR D) AWS Glue

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Amazon Athena

Athena allows direct SQL queries against data in S3 without loading or transformation. You pay only for queries run ($5 per TB scanned). Redshift requires loading

data, EMR requires cluster management, and Glue is for ETL (though it works well with Athena). ¡/details¿

—

Question 33

Which AWS service provides recommendations for cost optimization, security, performance, and fault tolerance?

A) AWS CloudTrail B) AWS Config C) AWS Trusted Advisor D) AWS Inspector

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) AWS Trusted Advisor

Trusted Advisor provides real-time best practice recommendations across five categories: cost optimization, performance, security, fault tolerance, and service limits. CloudTrail logs API calls, Config tracks configurations, and Inspector assesses security vulnerabilities. ¡/details¿

—

Question 34

Your application experiences a 10x traffic increase during a 2-hour window every day. Which compute solution provides automatic scaling with minimal management?

A) EC2 with manual scaling B) EC2 with Auto Scaling C) Lambda with CloudWatch Events D) Lightsail

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) Lambda with CloudWatch Events

Lambda automatically scales from zero to thousands of concurrent executions with no configuration needed. While EC2 Auto Scaling works, it takes minutes to provision new instances. Lambda scales instantly and you pay only for the 2-hour peak period. CloudWatch Events can trigger scheduled scaling if needed. ¡/details¿

—

Question 35

Which database service provides automatic scaling of both storage and compute capacity?

A) Amazon RDS B) Amazon Aurora Serverless C) Amazon DynamoDB D) Amazon Redshift

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Amazon Aurora Serverless

Aurora Serverless automatically adjusts database capacity based on application needs, scaling both compute (Aurora Capacity Units) and storage. DynamoDB can auto-scale throughput but not in the same way. RDS requires manual instance resizing for compute, though storage can auto-scale. Redshift requires manual cluster resizing. ¡/details¿

—

Question 36

A company needs to connect their VPC to an on-premises data center with a consistent, private connection at 10 Gbps. Which service should they use?

A) Site-to-Site VPN B) AWS Direct Connect C) VPC Peering D) Transit Gateway

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) AWS Direct Connect

Direct Connect provides dedicated private network connectivity at speeds from 1 Gbps to 100 Gbps with consistent performance. Site-to-Site VPN goes over the internet with variable performance, VPC Peering connects VPCs (not on-premises), and Transit Gateway connects networks but doesn't provide the physical connection. ¡/details¿

—

Question 37

Which S3 storage class automatically moves objects between access tiers based on changing access patterns?

A) S3 Standard B) S3 Intelligent-Tiering C) S3 Standard-IA D) S3 Glacier

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) S3 Intelligent-Tiering

Intelligent-Tiering automatically moves objects between Frequent Access and Infrequent Access tiers based on access patterns, optimizing costs without performance impact or operational overhead. Other storage classes require manual management or lifecycle policies. ¡/details¿

—

Question 38

Your application needs shared file storage accessible by multiple EC2 instances across different Availability Zones. Which service should you use?

A) Amazon EBS B) Amazon EFS C) Instance Store D) Amazon S3

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Amazon EFS

EFS provides shared NFS file storage accessible by multiple EC2 instances simultaneously across AZs. EBS can only attach to one instance at a time, Instance Store is ephemeral and instance-specific, and S3 is object storage (not a file system). ¡/details¿

—

Question 39

Which service would you use to track WHO made WHAT change to AWS resources and WHEN?

A) Amazon CloudWatch B) AWS CloudTrail C) AWS Config D) AWS X-Ray

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) AWS CloudTrail

CloudTrail logs all API calls (who, what, when, where) for governance, compliance, and auditing. CloudWatch monitors metrics and logs, Config tracks resource configurations over time, and X-Ray traces application requests. ¡/details¿

—

Question 40

A company needs to run Docker containers without managing EC2 instances. Which service combination should they use?

A) ECS with EC2 launch type B) ECS with Fargate launch type C) EKS with EC2 nodes D) EC2 with Docker installed

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) ECS with Fargate launch type

ECS with Fargate is serverless container orchestration where AWS manages all infrastructure. You define containers and Fargate handles provisioning, scaling, and management. ECS with EC2 and EKS with EC2 require managing instances. ¡/details¿

—

Question 41

Which database is optimized for storing and querying time-series data from IoT devices?

A) Amazon RDS B) Amazon DynamoDB C) Amazon Timestream D) Amazon Redshift

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) Amazon Timestream

Timestream is purpose-built for time-series data with automatic data lifecycle management and built-in time-series analytics functions. While DynamoDB can store time-series data, Timestream is optimized for this use case with better performance and cost efficiency. ¡/details¿

—

Question 42

Your application needs to send notifications to thousands of subscribers via multiple protocols (email, SMS, mobile push). Which service should you use?

A) Amazon SQS B) Amazon SNS C) Amazon EventBridge D) Amazon SES

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Amazon SNS

SNS (Simple Notification Service) is a pub/sub messaging service that can send notifications to multiple subscribers across different protocols (email, SMS, HTTP, mobile push, SQS, Lambda). SQS is for queuing, EventBridge for event routing, and SES is only for email. ¡/details¿

—

Question 43

Which service helps you discover and catalog data sources to prepare for migration to AWS?

A) AWS Migration Hub B) AWS Application Discovery Service C) AWS Database Migration Service D) AWS DataSync

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) AWS Application Discovery Service

Application Discovery Service collects information about on-premises servers including configuration, performance, and dependencies to plan migrations. Migration Hub tracks progress, DMS migrates databases, and DataSync transfers data. ¡/details¿

—

Question 44

A company needs to ensure their CloudFormation templates follow organizational standards and prevent non-compliant resources from being created. Which service should they use?

A) AWS Config B) AWS CloudTrail C) AWS Control Tower D) AWS Organizations ¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: C) AWS Control Tower

Control Tower provides guardrails (preventive and detective) to enforce compliance across accounts. Preventive guardrails use SCPs to prevent non-compliant actions. Config detects but doesn't prevent, CloudTrail logs actions, and Organizations provides account management but not guardrails. ¡/details¿

—

Question 45

Which ElastiCache engine supports complex data structures like sorted sets, lists, and pub/sub messaging?

A) Memcached B) Redis C) Both support these features equally D) Neither supports these features

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: B) Redis

Redis supports advanced data structures (strings, hashes, lists, sets, sorted sets), persistence, replication, and pub/sub messaging. Memcached is simpler and only supports simple key-value caching with multi-threading. For advanced use cases, Redis is the better choice. ¡/details¿

—

4.0.10 Summary

Domain 3: Cloud Technology and Services represents the largest portion (34%) of the AWS Certified Cloud Practitioner exam. This domain covers:

Key Areas:

- **Global Infrastructure**: Regions, AZs, Edge Locations, and specialized infrastructure
- **Compute**: From full control (EC2) to fully serverless (Lambda)
- **Storage**: Object (S3), block (EBS), file (EFS), and hybrid (Storage Gateway)
- **Databases**: Relational, NoSQL, caching, analytics, and specialized databases
- **Networking**: VPC, load balancing, content delivery, and connectivity
- **Management**: IaC, monitoring, logging, governance, and operations

- **Additional Services**: Messaging, analytics, ML/AI, and migration tools

Study Tips:

1. Understand the use cases for each service
2. Know when to choose one service over another
3. Remember pricing models, especially for EC2 and S3
4. Understand high availability and fault tolerance patterns
5. Know the differences between similar services (e.g., SNS vs SQS, RDS vs DynamoDB)

—

← Previous: Security and Compliance — Next: Billing, Pricing, and Support →

Chapter 5

Domain 4: Billing, Pricing, and Support

5.0.1 Table of Contents

- Case Study 2: Data Analytics Workload
- Case Study 3: Development Environment
- Cost Management Tools
- AWS Pricing Calculator
- AWS Cost Explorer
- AWS Budgets
- AWS Cost and Usage Report
- AWS Cost Anomaly Detection
- TCO Calculator Walkthrough
- Tagging Strategies for Cost Allocation
- Tag Best Practices
- Common Tagging Schemas
- Tag Enforcement
- Multi-Account Billing Setup
- Organization Structure
- Best Practices
- Cost Allocation
- Consolidated Billing and AWS Organizations
- Cost Anomaly Detection Deep Dive
- Setup and Configuration
- Alert Examples
- Response Workflows
- AWS Support Plans
- Support Plan Comparison
- Support Plan Decision Matrix
- Detailed Feature Comparison
- Additional Support Resources
- Cost Optimization Strategies
- Service-Specific Optimization

- Cost Governance and FinOps
- FinOps Framework
- Governance Policies
- Accountability and Ownership
- Billing Troubleshooting
- Common Issues
- Resolution Steps
- Review Questions

—

5.0.2 AWS Pricing Fundamentals

Core Principles

AWS pricing is built on several foundational principles that differentiate cloud computing from traditional on-premises infrastructure:

1. **Pay-as-you-go**: Pay only for what you use

- No upfront commitments required
- Start and stop resources at any time
- Only charged for actual consumption

1. **Pay less when you reserve**: Reserved capacity discounts

- Commit to usage for 1 or 3 years
- Receive significant discounts (up to 75%)
- Available for EC2, RDS, ElastiCache, Redshift, and more

1. **Pay less with volume-based discounts**: Use more, pay less per unit

- Tiered pricing automatically applies as usage increases
- Data transfer and storage pricing decreases with volume
- No negotiations required

1. **No upfront costs**: No capital expenditure

- Trade capital expense (CAPEX) for variable expense (OPEX)
- No infrastructure to purchase upfront

- Start with zero investment

1. **No termination fees**: Stop anytime

- No contracts or long-term commitments (unless you choose Reserved Instances)
- Delete resources when no longer needed
- Stop paying immediately

—

AWS Free Tier

AWS offers three types of free tier offerings to help new customers get started and experiment with services:

1. Always Free Services that never expire and are available to all AWS customers:

- **DynamoDB**: 25 GB of storage
- **Lambda**: 1 million requests per month
- **SNS**: 1 million publishes
- **CloudWatch**: 10 custom metrics and alarms
- **AWS Free Tier dashboard**: Monitor usage

2. 12 Months Free Services available for 12 months starting from account creation date:

- **EC2**: 750 hours/month of t2.micro or t3.micro instances
- **S3**: 5 GB of standard storage
- **RDS**: 750 hours/month of db.t2.micro database instances
- **CloudFront**: 50 GB data transfer out
- **Elastic Load Balancing**: 750 hours per month

Key Point

Note: The 750 hours of EC2 is enough to run one t2.micro instance continuously for a full month.

3. **Trials** Short-term free trials for specific services:

- **SageMaker**: 2 months free
- **Inspector**: 90 days free
- **Lightsail**: 1 month free (first month)
- **Amazon Comprehend Medical**: Various trial periods

Important

Important: Always set up billing alerts when using Free Tier to avoid unexpected charges if you exceed limits.

—

5.0.3 Pricing Models by Service Category

Compute Pricing

Amazon EC2

- **Instance Hours**: Pay for running instances (charged per second with 60-second minimum)
- **Pricing varies by**:
- Instance type (t2.micro, m5.large, etc.)
- Region (us-east-1 vs. eu-west-1)
- Operating system (Linux, Windows, RHEL)
- Tenancy (Shared vs. Dedicated)
- **Additional charges**:
- Data transfer out
- EBS storage volumes
- Elastic IP addresses (when not attached)

EC2 Purchase Options:

Purchase Option	Description	Discount	Use Case
On-Demand	Pay by the second, no commitment	Baseline	Short-term, unpredict
Reserved Instances	1 or 3 year commitment	Up to 75%	Steady-state, predicta
Spot Instances	Bid on unused capacity	Up to 90%	Fault-tolerant, flexible
Savings Plans	Commitment to consistent usage ($/hour)	Up to 72%	Flexible compute usag
Dedicated Hosts	Physical server dedicated to you	Varies	Compliance, licensing

AWS Lambda

- **Requests**: $0.20 per 1 million requests
- **Compute time**: Charged per GB-second
- Duration calculated from code execution start to return/termination
- Rounded up to nearest 1ms
- **Free tier**: 1 million requests/month (always free)
- **No charges** when code is not running

—

Storage Pricing

Amazon S3 Pricing components:

1. **Storage**: Pay for GB/month stored

- Varies by storage class (Standard, Infrequent Access, Glacier, etc.)
- Standard: ˜$0.023 per GB/month
- Standard-IA: ˜$0.0125 per GB/month
- Glacier: ˜$0.004 per GB/month

1. **Requests**:

- PUT, COPY, POST, LIST requests: $0.005 per 1,000
- GET, SELECT requests: $0.0004 per 1,000

1. **Data transfer**:

- Transfer IN: Free
- Transfer OUT to internet: Tiered pricing (first 10 TB/month at $0.09/GB)
- Transfer to CloudFront: Free

1. **Management features**:

- S3 Inventory, Analytics, Object Tagging

Amazon EBS

- **Provisioned storage**: Pay for capacity provisioned per GB/month
- gp3: $0.08/GB-month
- gp2: $0.10/GB-month
- io2: $0.125/GB-month + IOPS charges
- **Snapshots**: Incremental backup storage per GB/month
- **Varies by volume type**: General Purpose (SSD), Provisioned IOPS (SSD), Throughput Optimized (HDD)

Key Point

Key Difference: EBS charges for provisioned capacity, not used capacity. A 100 GB volume costs the same whether you store 10 GB or 100 GB.

—

Database Pricing

Amazon RDS Pricing components:

1. **Instance hours**: Based on instance class (db.t2.micro, db.m5.large)
2. **Storage**: Per GB/month of provisioned storage
3. **Backup storage**: Automated backups beyond database size
4. **Data transfer**: Standard AWS data transfer rates
5. **Additional features**:

- Multi-AZ deployment (doubles cost)
- Read replicas (charged as separate instances)

Amazon DynamoDB Two capacity modes:

1. **On-Demand**:

- Pay per request
- No capacity planning required
- Good for unpredictable workloads
- Write Request Units (WRU) and Read Request Units (RRU)

1. **Provisioned Capacity**:

- Pay for provisioned read/write capacity units

- Auto Scaling available
- More cost-effective for predictable workloads
- Reserve capacity for additional discounts

1. **Storage**: $0.25 per GB/month (first 25 GB free with Always Free tier)

—

Network Pricing

Understanding data transfer costs is crucial for cost optimization:

- **Data transfer IN**: Generally **free** from the internet to AWS
- **Data transfer OUT to internet**: **Charged** with tiered pricing
- First 10 TB/month: $0.09/GB
- Next 40 TB/month: $0.085/GB
- Over 150 TB/month: $0.05/GB
- **Data transfer between Regions**: **Charged** at inter-region rates
- **Data transfer within same Region**:
- Between AZs: $0.01/GB in each direction
- Within same AZ: Free (using private IPs)
- **CloudFront data transfer out**: Lower cost than direct from services
- **VPC Endpoints**: Reduce data transfer costs for S3 and DynamoDB

Key Point

Cost Optimization Tip: Use CloudFront CDN to cache content at edge locations, reducing data transfer costs from origin services.

—

5.0.4 Detailed Pricing Examples and Calculations

EC2 Pricing Scenario

Let's calculate the monthly cost for different EC2 purchasing options:

Scenario: Web application requiring 5 x m5.large instances (2 vCPU, 8 GB RAM) running 24/7 in us-east-1

On-Demand Pricing

```
Instance: m5.large
Rate: \$0.096 per hour
Hours per month: 730 hours (average)
Number of instances: 5

Monthly cost per instance: \$0.096 × 730 = \$70.08
Total monthly cost: \$70.08 × 5 = \$350.40/month
Annual cost: \$350.40 × 12 = \$4,204.80/year
```

1-Year Reserved Instance (Partial Upfront)

```
Upfront payment per instance: \$335
Monthly rate per instance: \$0.028/hour

Monthly recurring cost per instance: \$0.028 × 730 = \$20.44
Total upfront cost: \$335 × 5 = \$1,675
Total monthly cost: \$20.44 × 5 = \$102.20/month

First year total: \$1,675 + (\$102.20 × 12) = \$2,901.40
Savings vs On-Demand: \$4,204.80 - \$2,901.40 = \$1,303.40 (31\% savings)
```

3-Year Reserved Instance (All Upfront)

```
Upfront payment per instance: \$2,140
No monthly charges

Total upfront cost: \$2,140 × 5 = \$10,700
Monthly equivalent: \$10,700 ÷ 36 = \$297.22/month

Three-year total: \$10,700
Three-year On-Demand cost: \$4,204.80 × 3 = \$12,614.40
Savings: \$12,614.40 - \$10,700 = \$1,914.40 (15\% savings)
Annual savings: \$638.13/year (52\% annual savings)
```

Compute Savings Plan (1-Year, Partial Upfront)

```
Commitment: \$200/month
Coverage: Provides \~{}\$285 worth of On-Demand compute per month
Effective discount: \~{}30\%

Annual cost: \$200 × 12 = \$2,400 + upfront
Plus upfront: \~{}\$600
Total first year: \~{}\$3,000
Savings: \$4,204.80 - \$3,000 = \$1,204.80 (29\% savings)

Flexibility advantage: Can change instance types/sizes/regions
```

Spot Instance Pricing

```
Average spot price for m5.large: \~{}\$0.030/hour (varies by demand)
Potential savings: Up to 69\% off On-Demand

Monthly cost per instance: \$0.030 × 730 = \$21.90
Total monthly cost: \$21.90 × 5 = \$109.50/month
Annual cost: \$109.50 × 12 = \$1,314/year
Savings: \$4,204.80 - \$1,314 = \$2,890.80 (69\% savings)

Risk: Instances can be interrupted with 2-minute notice
Best for: Stateless applications with auto-restart capability
```

Cost Comparison Summary

Purchase Option	Monthly Cost	Annual Cost	3-Year Cost	Savings vs On-D
On-Demand	$350.40	$4,204.80	$12,614.40	Baseline (0%)
1-Yr RI (Partial)	$102.20 + $1,675 upfront	$2,901.40	-	31%
3-Yr RI (All Up)	$297.22 equiv	$3,566.67 equiv	$10,700	52%
Savings Plan	$250.00	$3,000.00	-	29%
Spot Instances	$109.50	$1,314.00	$3,942.00	69%

Additional Costs to Consider:

- EBS volumes: $0.10/GB-month (gp2) × 100 GB × 5 = $50/month
- Data transfer out: Variable based on usage
- Elastic Load Balancer: $16.20/month + $0.008/GB processed
- Total infrastructure estimate: Add 15-25% to compute costs

—

S3 Storage Cost Analysis

Scenario: 10 TB of data with different access patterns

Chapter 6

Hands-On Practice Labs

6.0.1 Table of Contents

- Troubleshooting FAQ
- Additional Practice Recommendations

—

6.0.2 Introduction

Exam Tip

Important: Hands-on experience is crucial for exam success. These labs will help you understand AWS services beyond theory and are designed to stay within Free Tier limits when done carefully.

Practical experience with AWS services provides:

- Deeper understanding of service capabilities
- Familiarity with AWS Management Console
- Confidence during the exam
- Real-world skills applicable to jobs
- Better retention of concepts

Total Time Investment: Approximately 11-12 hours for all 15 labs

—

6.0.3 Pre-Lab Checklist

Before starting any lab, ensure you have completed the following:

Essential Requirements

AWS Account created and activated (may take up to 24 hours)

Valid credit/debit card on file (required even for Free Tier)

Email access to confirm SNS subscriptions and receive alerts

Billing alerts set up (Lab 1 - do this FIRST!)

Free Tier dashboard bookmarked for easy monitoring

Account ID noted and saved securely

Technical Setup

Modern web browser (Chrome, Firefox, Safari, Edge - latest version)

Stable internet connection (minimum 5 Mbps recommended)

Text editor installed (VS Code, Sublime, Notepad++, or any editor)

SSH client available (built-in on Mac/Linux, PuTTY or OpenSSH for Windows)

Terminal/command prompt access and basic familiarity

Knowledge Prerequisites

Basic understanding of cloud computing concepts

Familiarity with IP addresses and networking basics (for VPC labs)

Basic command line experience (helpful but not required)

Understanding of JSON/YAML formats (for CloudFormation)

Safety Measures

Password manager or secure location for credentials

Notepad ready for documenting resource IDs and URLs

Calendar reminder set to check and clean up resources daily

Budget limit decided (recommend $10-20 maximum)

Understanding of charges - know what costs money vs. what's free

Best Practices Before Starting

Read entire lab before executing any steps

Take screenshots as you progress for documentation

Use consistent naming conventions (include date or lab number)

Tag all resources with project name for easy identification

Set aside uninterrupted time for each lab

Prepare to take notes on errors and solutions

Region Selection

Choose your primary region (recommend us-east-1 or us-west-2)

Verify Free Tier availability in chosen region

Note region for all labs to maintain consistency

Bookmark region selector for quick access

Time Planning

Review lab duration estimates

Add 25% buffer time for troubleshooting

Plan cleanup time (5-10 minutes per lab)

Schedule breaks between complex labs

Avoid starting labs late at night (may forget cleanup)

—

6.0.4 Prerequisites

Create Your AWS Account

Follow these steps to create your AWS account:

1. Visit https://aws.amazon.com
2. Click **"Create an AWS Account"**
3. Provide:

- Email address
- Password
- AWS account name

1. Enter contact information
2. Provide payment method

- Credit or debit card required
- You won't be charged if you stay within Free Tier

1. Verify identity via phone call or SMS
2. Select Support Plan: **Basic (Free)**
3. Wait for account activation (can take up to 24 hours)
4. Check email for confirmation

AWS Free Tier

Duration: 12 months from account creation date

Key Free Tier Services:

- **EC2:** 750 hours/month of t2.micro or t3.micro instances
- **S3:** 5 GB of standard storage
- **RDS:** 750 hours/month of db.t2.micro, db.t3.micro, or db.t4g.micro
- **Lambda:** 1 million free requests per month
- **CloudWatch:** 10 custom metrics and alarms

Always Free Services:

- DynamoDB: 25 GB storage
- Lambda: 1 million requests/month
- CloudFormation: No charge (pay for resources created)

Exam Tip

Set up billing alerts immediately to avoid unexpected charges. AWS Free Tier is generous, but mistakes can incur costs.

—

6.0.5 Important Notes

Safety and Cost Management

1. **Always clean up resources** after each lab to avoid charges
2. **Set billing alerts** before starting any hands-on work
3. **Use t2.micro or t3.micro** instance types (Free Tier eligible)
4. **Choose Free Tier regions** (us-east-1, us-west-2 recommended)
5. **Monitor Free Tier usage** in billing dashboard regularly
6. **Don't leave resources running** overnight or when not in use

Best Practices

- Complete labs in order (dependencies exist)
- Take notes and screenshots for future reference
- Read error messages carefully - they often contain solutions
- Use tags to identify lab resources for easy cleanup
- Stop services rather than terminate if you plan to return

Troubleshooting Resources

- AWS Documentation: https://docs.aws.amazon.com
- AWS re:Post (community forum): https://repost.aws
- Service Health Dashboard: https://status.aws.amazon.com
- AWS Support (if you have paid support plan)

—

6.0.6 Lab Difficulty Guide

Understanding Difficulty Ratings

Each lab is rated on three dimensions:

Technical Complexity:

- **Beginner:** Straightforward steps, minimal technical knowledge required
- **Intermediate:** Some technical concepts, may require troubleshooting
- **Advanced:** Complex networking/architecture, requires careful attention

Time Investment:

- Short: 15-30 minutes
- Medium: 30-60 minutes
- Long: 60+ minutes

Prerequisites:

- Minimal: Just AWS account
- Moderate: Completion of earlier labs helpful
- Extensive: Requires specific knowledge or completed labs

Lab Difficulty Matrix

Lab	Difficulty	Duration	Prerequisites	Complexity Score
Lab 1: Billing Alerts	Beginner	15 min	None	1/5
Lab 2: IAM & MFA	Beginner	30 min	None	2/5
Lab 3: EC2 Instance	Intermediate	45 min	Lab 2 recommended	3/5
Lab 4: S3 Website	Intermediate	40 min	Basic HTML	2/5
Lab 5: VPC Network	Advanced	60 min	Networking basics	4/5
Lab 6: RDS Database	Intermediate	30 min	Lab 3 & 5	3/5
Lab 7: CloudWatch	Beginner	25 min	Lab 3	2/5
Lab 8: Cost Tools	Beginner	30 min	None	1/5

Lab 9: Lambda	Intermediate	25 min	Basic Python	2/5
Lab 10: CloudFormation	Intermediate	20 min	YAML/JSON	3/5
Lab 11: Auto Scaling	Advanced	45 min	Lab 3 & 5	4/5
Lab 12: DynamoDB	Intermediate	35 min	Database concepts	2/5
Lab 13: SNS & SQS	Intermediate	30 min	Lab 9 helpful	3/5
Lab 14: Route 53	Intermediate	25 min	DNS knowledge	3/5
Lab 15: Organizations	Advanced	40 min	Multiple accounts	4/5

Recommended Learning Paths

Path 1: Absolute Beginners

1. Lab 1 (Billing) → Lab 2 (IAM) → Lab 8 (Cost Tools) → Lab 4 (S3)
2. Then proceed with: Lab 3 → Lab 7 → Lab 9 → Lab 12

Path 2: Developers

1. Lab 1 → Lab 2 → Lab 3 → Lab 9 (Lambda)
2. Then: Lab 4 → Lab 13 (Messaging) → Lab 12 (DynamoDB) → Lab 10 (IaC)

Path 3: Infrastructure/Operations

1. Lab 1 → Lab 2 → Lab 3 → Lab 5 (VPC)
2. Then: Lab 11 (Auto Scaling) → Lab 6 (RDS) → Lab 14 (Route 53) → Lab 15

Path 4: Complete Sequential (Recommended for Most)

- Follow labs 1-15 in order for comprehensive understanding

Skill Development Tracking

After completing each lab, self-assess your confidence:

Rating Scale:

- 1 = Need to review
- 2 = Understood with help
- 3 = Comfortable, could explain to others
- 4 = Expert, could troubleshoot independently
- 5 = Could teach this topic

Target Scores for Exam:

- Core labs (1-10): Aim for 4/5
- Advanced labs (11-15): Aim for 3/5

—

6.0.7 Lab 1: Set Up Billing Alerts and Budget

Duration: 15 minutes **Cost:** Free **Difficulty:** Beginner

Learning Objectives

By the end of this lab, you will be able to:

1. Enable IAM user access to billing information
2. Create CloudWatch billing alarms with SNS notifications
3. Configure AWS Budgets with multiple alert thresholds
4. Understand the difference between CloudWatch billing metrics and AWS Budgets
5. Monitor Free Tier usage to prevent unexpected charges
6. Set up proactive cost monitoring for AWS account safety

Why This Lab Matters

Real-World Scenario: A developer left an RDS database running in a personal AWS account and accumulated $450 in charges over a weekend. Proper billing alerts would have caught this within hours, limiting damage to less than $20.

Exam Relevance: The Cloud Practitioner exam heavily emphasizes cost management, billing, and monitoring. Questions often test your understanding of:

- AWS Budgets vs. Cost Explorer vs. CloudWatch billing alarms
- Free Tier limits and monitoring
- Billing alert best practices
- SNS notification mechanisms

Objective

Protect yourself from unexpected charges by setting up billing alerts and budgets.

Prerequisites

- Active AWS account
- Access to root account or IAM user with billing permissions

Step-by-Step Instructions

Key Point

What You'll See: The AWS Management Console with the naviga top showing your account name and region selector.

Part 1: Enable Billing Alerts

1. Sign in to **AWS Management Console** as root user or admin

- You should see the AWS Services search bar and dashboard

1. Click your **account name** (top right corner) → Select **"Account"**

- This opens the Account settings page
- Alternative path: Navigate via "My Account" link in dropdown

1. Scroll down to **"IAM User and Role Access to Billing Information"** section

- This section is about halfway down the page
- You'll see a description about allowing IAM users to access billing data
- **Why this matters:** By default, only root users can see billing info. This setting allows IAM users with appropriate permissions to view costs.

1. Click **"Edit"** button on the right side

- A popup or inline editor will appear

1. Check the box to **"Activate IAM Access"**

- When enabled, IAM users with billing permissions can see costs
- **Best Practice:** This allows you to use IAM users instead of root for daily billing monitoring

1. Click **"Update"**

- You'll see a success message: "Successfully updated IAM user/role access to billing information"

Validation Step:

- The setting should now show as "Activated"
- This is a one-time setup per AWS account

Common Issue: If you don't see this option, verify you're logged in as the root user (account owner), not an IAM user.

Part 2: Create CloudWatch Billing Alarm

1. Navigate to **CloudWatch** service
2. Select region: **N. Virginia (us-east-1)** (billing metrics only in us-east-1)
3. Go to **"Alarms"** → **"Billing"** → **"Create alarm"**
4. Click **"Select metric"**
5. Select **"Billing"** → **"Total Estimated Charge"**
6. Select **USD** checkbox
7. Click **"Select metric"**
8. Configure alarm:

- **Threshold type:** Static
- **Whenever:** Greater than
- **Amount:** $5 (or your preferred threshold)

1. Click **"Next"**

Part 3: Configure SNS Notification

1. Under **"Notification"**:

- **Select an SNS topic:** Create new topic
- **Topic name:** "Billing-Alerts"
- **Email endpoints:** Enter your email address

1. Click **"Create topic"**
2. Click **"Next"**
3. **Alarm name:** "Monthly-Billing-Alert"
4. **Alarm description:** "Alert when monthly charges exceed $5"
5. Click **"Next"**
6. Review and click **"Create alarm"**
7. **Check your email** and click confirmation link

Part 4: Create AWS Budget

1. Navigate to **"Billing and Cost Management"**
2. Click **"Budgets"** in left menu
3. Click **"Create budget"**
4. Select **"Cost budget"** → **"Next"**
5. Configure budget:

- **Budget name:** "Monthly-Cost-Budget"
- **Period:** Monthly
- **Budget amount:** $10
- **Budget scope:** All AWS services

1. Click **"Next"**
2. Configure alerts:

- **Alert 1:** 80% of budgeted amount
- **Email recipients:** Your email
- **Alert 2:** 100% of budgeted amount

1. Click **"Next"**
2. Review and click **"Create budget"**

Expected Outcomes

- CloudWatch billing alarm created and active
- Email confirmation received for SNS subscription
- Budget created with two alert thresholds
- Email notifications configured

Verification

1. Go to **CloudWatch** → **Alarms** → Verify alarm shows "OK" state
2. Go to **Budgets** → Verify budget shows current spend vs. budget
3. Check email for SNS confirmation

Troubleshooting

Problem: Billing metric not showing **Solution:** Ensure you're in us-east-1 region; billing metrics only available there

Problem: Email confirmation not received **Solution:** Check spam folder; resend confirmation from SNS console

Problem: Can't access billing information **Solution:** Enable IAM access to billing in Account settings

What You Should See at Each Step

After Creating CloudWatch Alarm:

- Alarm appears in CloudWatch → Alarms dashboard
- Status shows "Insufficient data" initially (normal - need 24 hours of data)
- After confirmation, status changes to "OK" (no alarm condition met)
- Graph shows estimated charges over time

After Confirming SNS Subscription:

- Email from "AWS Notifications" with subject "AWS Notification - Subscription Confirmation"
- After clicking link, browser shows "Subscription confirmed!"
- SNS console shows subscription status as "Confirmed"

After Creating Budget:

- Budget appears in Budgets dashboard
- Shows current spend vs. budgeted amount (likely $0.00 of $10.00)
- Alert thresholds displayed at 80% ($8) and 100% ($10)
- Forecast shows projected end-of-month spend

Real-World Tips

Recommended Alert Thresholds:

- For students/learners: $5 alarm, $10 budget
- For production accounts: Set based on expected monthly spend
- Conservative approach: Alert at 50%, 80%, and 100%

Multiple Budget Strategy:

- Total account budget: $10
- Per-service budgets: EC2 $3, RDS $3, S3 $2, Other $2

- Helps identify which service is driving costs

Alert Fatigue Prevention:

- Don't set thresholds too low (avoid constant alerts)
- Review and adjust monthly based on actual usage
- Use forecasted budgets for proactive monitoring

Best Practice - Multiple Notification Recipients:

- Add team members' emails to SNS topic
- Consider SMS notifications for critical budgets (note: SMS costs money)
- Set up Slack/Teams integration via Lambda (advanced)

Alternative Approaches

Method 1: AWS Budgets Only

- Skip CloudWatch billing alarm
- Use only AWS Budgets (simpler for beginners)
- Limitation: Less granular, updated 3x per day vs. CloudWatch every 5 minutes

Method 2: CloudWatch Only

- Skip AWS Budgets
- Create multiple CloudWatch alarms at different thresholds
- Limitation: Doesn't show forecasts or budget tracking UI

Method 3: Cost Anomaly Detection (Advanced)

- AWS provides ML-based anomaly detection
- Navigate to Cost Management → Cost Anomaly Detection
- Create monitor → Set alert preferences
- Automatically detects unusual spending patterns

Cleanup

Key Point

Note: Keep these alerts active for ongoing protection. No cleanup needed.

If you need to remove alerts later:

1. **Delete CloudWatch Alarm:**

- CloudWatch → Alarms → Select alarm → Actions → Delete
- Confirm deletion by typing alarm name
- Alarm deleted immediately

1. **Delete Budget:**

- Billing → Budgets → Select budget → Actions → Delete budget
- Type "delete" to confirm
- Budget removed from dashboard

1. **Delete SNS Topic:**

- SNS → Topics → Select topic → Delete
- Type "delete me" to confirm
- Associated subscriptions automatically deleted

1. **Verify Cleanup:**

- Check CloudWatch Alarms list (should be empty)
- Check Budgets dashboard (should show no budgets)
- Emails stop arriving

Cost Impact of Deletion:

- No ongoing costs for these free services
- Safe to keep active indefinitely

—

Post-Lab Knowledge Check

Test your understanding of Lab 1 concepts:

Question 1: What's the difference between CloudWatch billing alarms and AWS Budgets?

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer:

- **CloudWatch Alarms:** Monitor actual charges in near real-time (every 5-10 minutes), trigger SNS notifications when threshold exceeded. Only available in us-east-1 region.
- **AWS Budgets:** Track costs and usage against planned budgets, provide forecasting, update 3 times per day. Support usage-based budgets (not just cost). Available globally.
- **Use both together:** CloudWatch for immediate alerts, Budgets for tracking and forecasting.

¡/details¿

Question 2: Why must CloudWatch billing metrics be created in the us-east-1 region?

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: Billing is a global service, and AWS consolidates all billing data in the us-east-1 (N. Virginia) region. Billing metrics are only published to CloudWatch in this region. This is an AWS design decision to centralize global billing data.

Exam Tip: Remember this for the exam - it's a common trick question!

¡/details¿

Question 3: If your alarm shows "Insufficient data" status, is something wrong?

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: No, this is normal. "Insufficient data" means CloudWatch doesn't have enough data points yet to evaluate the alarm. This typically happens:

- Within first 24 hours of alarm creation
- For new AWS accounts with minimal usage
- After changing alarm parameters

Status will change to "OK" once sufficient data is available. If charges exceed threshold, status changes to "In alarm."

¡/details¿

Question 4: You set a $10 budget but received an alert at $8. Why?

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: You configured an alert threshold at 80% of your budget. 80% of $10 = $8. This is intentional and a best practice. Getting alerts before hitting 100% gives you time to investigate and take action before exceeding your budget.

Multiple thresholds (50%, 80%, 100%) provide escalating warnings as you approach your limit.

¡/details¿

Question 5: Can you set up billing alarms for individual services like EC2 or S3?

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer:

- **CloudWatch Billing Alarms:** Can only monitor total estimated charges for the account, not individual services.
- **AWS Budgets:** YES, can create service-specific budgets (e.g., EC2 only, S3 only).
- **Best Practice:** Use AWS Budgets for service-level cost tracking, CloudWatch alarms for total account spending.

¡/details¿

Question 6: What happens if you don't confirm the SNS subscription email?

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: The subscription remains in "Pending confirmation" status and you will NOT receive any alarm notifications. The alarm will still evaluate and trigger, but emails won't be sent.

Always check spam folder and confirm subscriptions within 3 days. After 3 days, you may need to recreate the subscription.

¡/details¿

Question 7: Your Free Tier includes 10 CloudWatch alarms. What happens if you create 11?

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer: You'll be charged $0.10 per alarm per month for the 11th alarm (and any beyond that). With 11 alarms, cost would be $0.10/month.

Best Practice for Free Tier:

- Stay within 10 alarms
- Use AWS Budgets (free) for additional monitoring
- Combine multiple thresholds in fewer alarms

¡/details¿

Question 8: How often should you check your Free Tier usage dashboard?

¡details¿ ¡summary¿Click to reveal answer¡/summary¿

Answer:

- **Minimum:** Weekly
- **Recommended:** Every 2-3 days when actively learning
- **Best Practice:** Daily while running labs
- **Set Calendar Reminder:** Add recurring reminder to check dashboard

Free Tier dashboard shows current month usage vs. limits with visual progress bars. Catches issues before they become charges.

¡/details¿

Key Takeaways

- **Billing alerts are your safety net** - Set them up before doing anything else in AWS
- **Use both CloudWatch alarms and Budgets** - They complement each other
- **Always confirm SNS subscriptions** - Check spam folder if needed
- **Monitor Free Tier usage** - Make it a daily habit during learning phase
- **Be conservative with thresholds** - Better to get warned early than surprised by charges
- **Billing metrics are us-east-1 only** - Remember this for the exam
- **Budgets can track usage, not just costs** - Useful for monitoring Free Tier hours
- **Free Tier is per service** - 750 EC2 hours + 750 RDS hours, not combined

Additional Resources

- AWS Billing and Cost Management Documentation
-

Chapter 7

Practice Questions and Exam Tips

7.1 Sample Questions by Domain

7.1.1 Domain 1: Cloud Concepts

Question 1: Which of the following are advantages of cloud computing? (Select TWO)

A. Trade capital expense for variable expense

B. Increase time to market

C. Benefit from massive economies of scale

D. Maintain infrastructure

Answer: A and C **Question 2:** Which migration strategy involves moving applications to the cloud without making changes?

A. Repurchasing

B. Rehosting

C. Refactoring

D. Replatforming

Answer: B (Lift and Shift)

7.1.2 Domain 2: Security and Compliance

Question 3: According to the Shared Responsibility Model, which security aspect is AWS responsible for?

A. Security group configuration

B. Physical security of data centers

C. Customer data encryption

D. IAM user management

Answer: B **Question 4:** Which service provides DDoS protection at no additional cost?

A. AWS WAF
B. AWS Shield Advanced
C. AWS Shield Standard
D. Amazon GuardDuty

Answer: C

7.1.3 Domain 3: Technology and Services

Question 5: Which compute service allows you to run code without provisioning servers?

A. Amazon EC2
B. AWS Lambda
C. Amazon ECS
D. AWS Elastic Beanstalk

Answer: B **Question 6:** Which storage class is most cost-effective for infrequently accessed data that needs millisecond retrieval?

A. S3 Standard
B. S3 Glacier Instant Retrieval
C. S3 Standard-IA
D. S3 Glacier Deep Archive

Answer: C

7.1.4 Domain 4: Billing and Pricing

Question 7: Which AWS Support plan provides a Technical Account Manager?

A. Basic
B. Developer
C. Business
D. Enterprise

Answer: D **Question 8:** Which service helps you create cost estimates for AWS solutions?

A. AWS Cost Explorer

B. AWS Pricing Calculator
C. AWS Budgets
D. AWS Cost and Usage Report

Answer: B

7.2 Test-Taking Strategies

7.2.1 Before the Exam

1. **Review all exam objectives**: Ensure you've covered each domain
2. **Take practice exams**: Identify weak areas
3. **Get hands-on experience**: Create AWS account, experiment with services
4. **Review AWS documentation**: Especially FAQs for key services
5. **Get adequate rest**: Sleep well before exam day

7.2.2 During the Exam

1. **Read questions carefully**: Pay attention to keywords
   - "MOST cost-effective"
   - "BEST"
   - "LEAST amount of effort"
   - "NOT" or "EXCEPT"
2. **Eliminate wrong answers**: Narrow down choices
3. **Watch for absolutes**: "Always," "never," "all," "none" are often wrong
4. **Flag difficult questions**: Return to them later
5. **Manage time**: Don't spend too long on one question
6. **Trust your preparation**: Your first instinct is often correct
7. **Use process of elimination**: Remove obviously incorrect answers
8. **Review flagged questions**: Use remaining time to review

7.2.3 Common Question Patterns

- **Scenario-based**: Describe situation, ask for best solution
- **Select TWO/THREE**: Multiple correct answers
- **Best practice**: What's the recommended approach?
- **Cost optimization**: Which option is most cost-effective?
- **Security**: Which option is most secure?
- **High availability**: Which design ensures uptime?

7.3 Common Pitfalls to Avoid

1. **Confusing service names**: EC2 vs. ECS vs. EKS; S3 vs. EBS vs. EFS
2. **Not understanding Shared Responsibility**: Know what AWS vs. customer manages
3. **Mixing up support plans**: Especially response times and TAM
4. **Forgetting storage classes**: S3 storage tiers and use cases
5. **Confusing pricing models**: On-Demand vs. Reserved vs. Spot
6. **Not knowing AWS global infrastructure**: Regions, AZs, Edge Locations
7. **Overlooking "EXCEPT" questions**: Read carefully
8. **Assuming real-world complexity**: Choose AWS-recommended simple solution

7.4 Key Concepts to Memorize

Exam Tip

Focus on understanding concepts rather than memorization, but these facts appear frequently on the exam.

7.4.1 Numbers to Remember

- S3 durability: 99.999999999% (11 nines)
- Minimum AZs per Region: 3
- Edge Locations: 400+
- Lambda max execution: 15 minutes
- RDS read replicas: Up to 5
- Support plan response times
- Free tier: 12 months for EC2, S3, RDS

7.4.2 Service Comparisons

- S3 vs. EBS vs. EFS
- RDS vs. DynamoDB vs. Redshift
- EC2 vs. Lambda vs. Elastic Beanstalk
- CloudWatch vs. CloudTrail vs. Config
- SNS vs. SQS
- Security Groups vs. NACLs
- ALB vs. NLB

% Include additional comprehensive chapters

Chapter 8

Comprehensive Service Comparisons and Decision Trees

This chapter provides detailed comparisons of AWS services to help you choose the right service for specific use cases. These comparisons are critical for the exam.

8.0.1 Storage Services Detailed Comparison

Feature & **Amazon S3** & **Amazon EBS** & **Amazon EFS** & **Instance Store**
Type & Object Storage & Block Storage & File Storage & Ephemeral Block
Use Case & Backups, archives, web content, data lakes & Boot volumes, databases, transactional da
Access & HTTP/S API, SDK, CLI & Attached to EC2 instance & NFSv4 protocol, multiple EC2 in
Durability & 11 nines (99.999999999%) & Replicated within AZ & Replicated across AZs in Region
Scalability & Unlimited & Up to 64 TB per volume & Petabyte scale & Fixed to instance type
Performance & Varies by class & Up to 256,000 IOPS & Scales with size & Highest IOPS for insta
Cost & Low per GB, varies by class & $0.08-0.125/GB-month & $0.30/GB-month & Included with i
Availability & 99.9-99.99% SLA & 99.8-99.9% & 99.99% & Dependent on instance
Backup & Versioning, lifecycle & Snapshots to S3 & AWS Backup & Not persistent
Multi-AZ & Yes & No (single AZ) & Yes & No
Concurrent Access & Unlimited & Single EC2 instance & Thousands of instances & Single instanc

When to Use Each Storage Service

Use S3 when:

- Storing static website content
- Backup and archiving
- Data lakes for analytics
- Distributing software/media
- Disaster recovery

Use EBS when:

- EC2 boot volumes required
- Databases requiring consistent IOPS
- Applications needing block storage
- Single instance access sufficient

Use EFS when:

- Multiple instances need shared access
- Content management systems
- Web serving from shared storage
- Big data analytics requiring shared access
- Lift-and-shift of on-premises applications

Use Instance Store when:

- Temporary storage needs
- Cache and buffers
- Scratch data
- Data replicated across instances
- Highest possible IOPS needed

8.0.2 Database Services Detailed Comparison

Service & **Type** & **Use Case** & **Scaling** & **Pricing Model** & **Managed**
RDS & Relational (SQL) & Traditional apps, OLTP & Vertical, Read Replicas & Instance + storag
Aurora & Relational (MySQL/PostgreSQL compatible) & High-performance OLTP & Auto-scaling
DynamoDB & NoSQL Key-Value & Web/mobile apps, gaming, IoT & Auto horizontal & On-dema
Redshift & Data Warehouse (OLAP) & Analytics, BI, big data & Add nodes & Nodes + storage &
ElastiCache & In-memory cache & Session stores, leaderboards & Add nodes & Nodes (hourly) & F
Neptune & Graph database & Social networks, recommendations & Vertical & Instances & Fully m
DocumentDB & Document (MongoDB compatible) & Content management, catalogs & Vertical, r
Timestream & Time series & IoT, DevOps metrics & Auto & Storage + queries & Fully managed

Database Selection Decision Tree

```
Need SQL and ACID transactions?
 Yes
   Traditional app, existing database → RDS
   Need extreme performance → Aurora
   Specific needs: Oracle/SQL Server → RDS with that engine

 Need NoSQL?
    Key-value, scale to millions of requests/sec → DynamoDB
```

```
    Document database, MongoDB compatible → DocumentDB
    Graph relationships → Neptune

Need caching?
 In-memory cache → ElastiCache
 Redis features needed → ElastiCache for Redis
 Simple caching → ElastiCache for Memcached

Need data warehouse/analytics?
 OLAP, BI, big data → Redshift
 Query data in S3 → Athena
 Real-time analytics → Kinesis Data Analytics

Time-series data?
 IoT, metrics, logs → Timestream
```

8.0.3 Compute Services Decision Tree

Choose Your Compute Service

Need full control over OS and configuration?

- Yes → Use **EC2**
- Want to save costs for predictable workloads? → Use **Reserved Instances**
- Fault-tolerant batch workloads? → Use **Spot Instances**

Want to run code without managing servers?

- Event-driven, ¡ 15 min execution → Use **Lambda**
- Long-running, stateless → Use **Fargate**

Need to deploy web applications quickly?

- Simple deployment, don't want infrastructure management → Use **Elastic Beanstalk**
- Need predictable pricing for small projects → Use **Lightsail**

Using containers?

- Want AWS-native orchestration → Use **ECS**
- Need Kubernetes compatibility → Use **EKS**
- Don't want to manage servers → Use **Fargate** (with ECS or EKS)

Running batch processing jobs?

- Use **AWS Batch**

Compute Service Selection Table

Requirement & Best Choice & Why
Full OS control & EC2 & Complete flexibility
Serverless, event-driven & Lambda & No servers, pay per use
Deploy app quickly & Elastic Beanstalk & PaaS, managed
Containers, Kubernetes & EKS & Kubernetes compatible
Containers, AWS native & ECS & Simpler than EKS
Containers, no servers & Fargate & Serverless containers
Batch processing & AWS Batch & Optimized for batch
Simple website, blog & Lightsail & Predictable pricing

8.0.4 Networking Components Deep Dive

Component & Purpose & Key Points
Internet Gateway (IGW) & Connect VPC to internet & One per VPC; enables internet access fo
NAT Gateway & Outbound internet from private subnets & Highly available; placed in public subr
NAT Instance & Alternative to NAT Gateway & EC2 instance; you manage; lower cost but less re
VPC Peering & Connect two VPCs & Non-transitive; can be cross-account/region; no overlapping
Transit Gateway & Hub for connecting VPCs & Simplifies complex network topologies; central ma
VPN Gateway & VPN connection to on-premises & IPsec VPN; encrypted over internet; quick set
Direct Connect & Dedicated connection to on-premises & Private, consistent bandwidth; expensive
VPC Endpoints & Private connection to AWS services & No internet required; Interface or Gatew
PrivateLink & Private connectivity to services & Access services in other VPCs; doesn't require VP

VPC Components Decision Guide

Public vs. Private Subnet:

- **Public Subnet**: Has route to Internet Gateway (0.0.0.0/0 → IGW)
- **Private Subnet**: No direct internet access; uses NAT for outbound

NAT Gateway vs. NAT Instance:

- **NAT Gateway**: AWS managed, highly available, easier, more expensive
- **NAT Instance**: EC2 instance you manage, cheaper, less reliable

VPC Peering vs. Transit Gateway:

- **VPC Peering**: Connect two VPCs directly, non-transitive
- **Transit Gateway**: Hub-and-spoke model, simplifies multiple VPC connections

VPN vs. Direct Connect:

- **VPN**: Quick setup (hours), encrypted, over internet, variable performance

- **Direct Connect**: Dedicated link, consistent performance, expensive, takes weeks

VPC Endpoint Types:

- **Interface Endpoint**: ENI with private IP, uses PrivateLink
- **Gateway Endpoint**: Route table entry, for S3 and DynamoDB only

8.0.5 Load Balancer Detailed Comparison

Feature	ALB	NLB	Gateway LB
OSI Layer	Layer 7 (Application)	Layer 4 (Transport)	Layer 3 (Network)
Protocol	HTTP, HTTPS, WebSocket	TCP, UDP, TLS	IP
Routing	Path-based, host-based, query string	IP address, port	N/A
Use Case	Web applications, microservices	High performance, low latency, static IP	Third-pa
Target Types	IP, instance, Lambda	IP, instance, ALB	IP, instance
Performance	Good	Extreme (millions req/sec)	High
Static IP	No (DNS only)	Yes (Elastic IP)	N/A
SSL Termination	Yes	Yes	No
WebSocket	Yes	Yes	No
Health Checks	Advanced	Basic	Advanced
Pricing	Per hour + LCU	Per hour + LCU	Per hour + LCU

When to Use Each Load Balancer

Application Load Balancer (ALB):

- Web applications
- Microservices architectures
- Need advanced routing (path, host, header-based)
- Target Lambda functions
- Need WebSocket support
- Content-based routing required

Network Load Balancer (NLB):

- Extreme performance required (millions of requests/second)
- Ultra-low latency needed
- Static IP addresses required
- TCP/UDP traffic
- Non-HTTP protocols
- Preserve source IP address

Gateway Load Balancer:

- Deploy third-party virtual appliances
- Firewalls, IDS/IPS
- Deep packet inspection
- Transparent network gateway

Classic Load Balancer (CLB):

- Legacy applications
- EC2-Classic network
- Being phased out - migrate to ALB or NLB

8.0.6 Security Services Complete Matrix

Service & What It Does & When to Use
IAM & Identity and access management & Control who can access what in AWS
AWS Organizations & Multi-account management & Centralize billing, apply policies across accou
AWS SSO & Single sign-on & Centrally manage access to multiple accounts and applications
Cognito & User authentication for apps & Add sign-up/sign-in to mobile and web apps
Directory Service & Managed Active Directory & Integrate AWS with existing Microsoft AD
Secrets Manager & Store and rotate secrets & Automatically rotate database credentials
KMS & Encryption key management & Create and control encryption keys
CloudHSM & Hardware security modules & Dedicated hardware for regulatory compliance
Certificate Manager & SSL/TLS certificates & Free certificates for ELB, CloudFront, API Gatewa
WAF & Web application firewall & Protect against SQL injection, XSS attacks
Shield Standard & DDoS protection & Automatic protection (free)
Shield Advanced & Enhanced DDoS protection & 24/7 DDoS Response Team, cost protection ($3
GuardDuty & Threat detection & Continuous monitoring for malicious activity
Inspector & Vulnerability assessment & Scan EC2 and container images for vulnerabilities
Macie & Data privacy and protection & Discover and protect sensitive data in S3
Detective & Security investigation & Analyze and investigate security issues
Security Hub & Security posture management & Centralized view of security alerts and compliance
Firewall Manager & Centralized firewall management & Manage WAF, Shield across accounts

Security Service Selection Guide

Identity and Access:

- Internal AWS users/services → IAM
- External app users → Cognito
- Multiple AWS accounts → Organizations + SSO
- On-premises AD integration → Directory Service

Data Protection:

- Encryption key management → KMS

- Regulatory compliance encryption → CloudHSM
- Store secrets → Secrets Manager
- Discover sensitive data → Macie

Threat Protection:

- DDoS protection → Shield (Standard for all, Advanced for critical)
- Web application attacks → WAF
- Threat detection → GuardDuty
- Vulnerability scanning → Inspector
- Security investigation → Detective

Compliance and Governance:

- Multi-account security → Security Hub
- Multi-account firewall → Firewall Manager
- Configuration compliance → Config
- Audit logging → CloudTrail

8.0.7 Service Limits Quick Reference

Service & Default Limit & Notes
EC2 Instances (On-Demand) & 20 per region & Can request increase
VPCs per Region & 5 & Can request increase
Internet Gateways per Region & 5 & One per VPC typically
S3 Buckets per Account & 100 & Soft limit, can increase
S3 Object Size & 5 TB max & Use multipart upload for ¿ 100 MB
RDS DB Instances & 40 per region & Can request increase
Lambda Concurrent Executions & 1,000 & Can request increase
Lambda Function Timeout & 15 minutes max & Cannot be increased
CloudFormation Stacks & 200 per region & Can request increase
IAM Users per Account & 5,000 & Use roles/federated identities instead
IAM Groups per Account & 300 & Plan group structure carefully

Exam Tip

Most service limits can be increased by requesting through AWS Support. Some hard limits (like Lambda 15-minute timeout) cannot be changed.

8.0.8 Storage Selection Guide

Use Case & Service & Reason

Backups, archives & S3 Glacier & Lowest cost
Website content & S3 + CloudFront & Scalable, global
EC2 boot volume & EBS & Block storage
Shared file system & EFS & Multi-attach
Windows file shares & FSx for Windows & SMB, AD integration
HPC workloads & FSx for Lustre & High performance
Temporary data & Instance Store & Highest IOPS
Offline transfer & Snow Family & Large data volumes

8.0.9 Key Comparison Summary

Storage Decision Matrix

- **Objects/Files** → S3
- **Block storage for EC2** → EBS
- **Shared file system** → EFS
- **Archive** → S3 Glacier
- **Large migration** → Snow Family

Database Decision Matrix

- **Relational/SQL** → RDS or Aurora
- **NoSQL key-value** → DynamoDB
- **Caching** → ElastiCache
- **Data warehouse** → Redshift
- **Graph** → Neptune

Compute Decision Matrix

- **Full control** → EC2
- **Serverless functions** → Lambda
- **Containers with orchestration** → ECS/EKS
- **Serverless containers** → Fargate
- **Simple web apps** → Lightsail
- **Quick app deployment** → Elastic Beanstalk

Networking Decision Matrix

- **Private network** → VPC
- **Content delivery** → CloudFront
- **DNS** → Route 53
- **Load balancing** → ALB/NLB
- **Hybrid connectivity** → Direct Connect or VPN

8.0.10 Serverless Services Comparison

Service & **Type** & **Use Case** & **Execution Model** & **Pricing** & **Integration**
Lambda & Function as a Service & Event-driven code execution, APIs, automation & Up to 15 min
Fargate & Serverless containers & Long-running containers, microservices & Continuous & Per vCP
AppSync & GraphQL API & Real-time and offline mobile/web apps & Request-based & Per query
Step Functions & Workflow orchestration & Coordinate multiple Lambda functions, services & Sta

Serverless Service Selection

Use Lambda when:

- Event-driven processing (S3 uploads, DynamoDB streams)
- API backends via API Gateway
- Scheduled tasks (CloudWatch Events)
- Real-time file/stream processing
- Execution time ¡ 15 minutes
- Stateless operations

Use Fargate when:

- Long-running applications
- Microservices that run continuously
- Migrating Docker containers to serverless
- Need more than 15 minutes execution
- Application requires persistent connections
- Want container benefits without managing servers

Use AppSync when:

- Building GraphQL APIs
- Real-time data synchronization needed
- Offline mobile app support required
- Multiple data sources (DynamoDB, Lambda, RDS)
- Need built-in caching and subscriptions

Use Step Functions when:

- Coordinating multiple Lambda functions
- Complex workflows with branching logic
- Need retry and error handling
- Long-running workflows (up to 1 year)
- Visual workflow management required
- Orchestrating microservices

8.0.11 Analytics Services Comparison

Service & Type & Best For & Data Source & Query Method & Pricing
Athena & Interactive query & Ad-hoc SQL queries on S3 & S3 (CSV, JSON, Parquet, ORC) & SQ
EMR & Big Data platform & Hadoop, Spark workloads & S3, HDFS & MapReduce, Spark, Hive, P
Redshift & Data warehouse & OLAP, BI, complex queries & S3, databases via COPY & SQL (Post
QuickSight & BI visualization & Dashboards and reports & Athena, Redshift, RDS, S3 & Visual in
Glue & ETL service & Data preparation, cataloging & S3, RDS, Redshift & Python/Scala (Spark) &
Kinesis & Real-time streaming & Real-time data processing & Streams, applications, IoT & Kinesis
Lake Formation & Data lake & Build and secure data lakes & S3 & Integrated with Athena, EMR
MSK & Managed Kafka & Streaming with Kafka & Applications & Kafka API & Per broker hour

Analytics Service Selection Guide

Use Athena when:

- Querying data already in S3
- Serverless, no infrastructure to manage
- Ad-hoc analysis and exploration
- Cost-effective for infrequent queries
- No need to load data into database

Use EMR when:

- Need full Hadoop/Spark ecosystem
- Complex data processing pipelines
- Custom big data frameworks
- ML model training at scale
- Have existing Hadoop workloads

Use Redshift when:

- Data warehouse for OLAP workloads
- Complex joins across large datasets
- BI tools need fast query performance
- Petabyte-scale analytics
- Consistent query performance needed

Use QuickSight when:

- Creating business dashboards
- Visualizing data from multiple sources
- Sharing insights with stakeholders
- Mobile-friendly BI needed

- Cost-effective per-user pricing

Use Glue when:

- ETL jobs to prepare data
- Automatic schema discovery needed
- Data catalog for analytics services
- Moving data between data stores
- Serverless ETL without managing servers

Use Kinesis when:

- Real-time data streaming
- Log and event data collection
- Real-time analytics
- Click stream analysis
- IoT telemetry processing

8.0.12 Migration Tools Comparison

Tool & Purpose & Use Case & Data Size & Transfer Method & Timeframe
MGN (Application Migration Service) & Lift-and-shift server migration & Migrate physical/vi
DMS (Database Migration Service) & Database migration & Migrate databases with minimal d
DataSync & Data transfer & Migrate file data to/from AWS & GBs to PBs & Network (accelerated
Snow Family & Physical data transfer & Offline bulk data transfer & TBs to exabytes & Physical d
Transfer Family & SFTP/FTPS/FTP & Migrate file transfer workflows & Any & Network (legacy
Migration Hub & Migration tracking & Centralized migration planning and tracking & N/A & Da

Migration Tool Details

AWS Application Migration Service (MGN):

- Formerly CloudEndure Migration
- Automated lift-and-shift for servers
- Continuous block-level replication
- Minimal downtime (minutes)
- Supports physical, virtual, cloud servers
- Free for 90 days (pay only for resources)

Database Migration Service (DMS):

- Homogeneous migrations (Oracle to Oracle)
- Heterogeneous migrations (Oracle to Aurora)

- Continuous replication for minimal downtime
- Schema Conversion Tool (SCT) for different engines
- Supports most commercial and open-source databases

DataSync:

- 10x faster than open-source tools
- Automates data transfer
- Validates data integrity
- Use for: NFS/SMB to EFS, FSx
- Bandwidth optimization and encryption
- Schedule transfers

Snow Family:

- **Snowcone**: 8 TB usable, edge computing
- **Snowball Edge**: 80 TB usable, compute options
- **Snowmobile**: 100 PB, truck-sized container
- Use when: Limited bandwidth, weeks to transfer over network
- Includes compute for edge processing
- Ruggedized for harsh environments

Transfer Family:

- Managed SFTP, FTPS, FTP service
- Stores files in S3 or EFS
- Migrate legacy file transfer workflows
- Integrate with existing authentication
- Fully managed, no servers to maintain

Migration Decision Tree

```
How much data to migrate?
 < 10 TB with good bandwidth → DataSync over network
 10-80 TB or limited bandwidth → Snowcone/Snowball
 > 80 TB or very limited bandwidth → Snowball Edge/Snowmobile

What type of data?
 Servers/VMs → MGN (Application Migration Service)
 Databases → DMS (Database Migration Service)
 Files/NAS → DataSync or Snow Family
 SFTP workflows → Transfer Family

Need ongoing synchronization?
 Yes → DataSync (scheduled) or DMS (continuous)
```

```
 No → One-time Snow Family transfer

Network bandwidth available?
 > 100 Mbps → DataSync
 10-100 Mbps → Consider Snowcone
 < 10 Mbps → Snow Family required
```

8.0.13 Container Orchestration Detailed Comparison

Feature & **ECS** & **EKS** & **Fargate (with ECS)** & **Fargate (with EKS)**
Orchestrator & AWS proprietary & Kubernetes & ECS (serverless) & Kubernetes (serverless)
Learning Curve & Easier & Steeper & Easiest & Moderate
Control Plane & Free & $0.10/hour per cluster & Free & $0.10/hour per cluster
Infrastructure Management & You manage EC2 & You manage nodes & AWS manages & AWS
Kubernetes Compatibility & No & Yes & No & Yes
Use Case & AWS-native containers & Kubernetes workloads & Serverless containers & Serverless K
Pricing & EC2 costs only & Cluster + EC2 costs & Per vCPU + memory & Per vCPU + memory
Add-ons/Ecosystem & AWS services & Kubernetes ecosystem & AWS services & Both
Multi-cloud & AWS only & Portable & AWS only & More portable

Container Service Selection

Use ECS when:

- Deep AWS integration preferred
- Don't need Kubernetes
- Want simpler container orchestration
- Team familiar with AWS services
- No existing Kubernetes investment
- Cost-sensitive (no control plane cost)

Use EKS when:

- Need Kubernetes compatibility
- Existing Kubernetes workloads
- Want portability across clouds
- Team has Kubernetes expertise
- Need Kubernetes ecosystem tools
- Hybrid/multi-cloud strategy

Use Fargate (with ECS or EKS) when:

- Don't want to manage servers
- Variable workloads

- Want pay-per-task pricing
- Security isolation per task
- Quick deployment without infrastructure
- Development and testing environments

Container Launch Types:

- **EC2 Launch Type**: You manage EC2 instances, more control, potentially lower cost
- **Fargate Launch Type**: Serverless, no EC2 management, pay per task

Container Architecture Patterns

ECS Architecture:

- Task Definition (container config)
- Service (maintains desired count)
- Cluster (logical grouping)
- EC2 or Fargate launch type

EKS Architecture:

- Managed Kubernetes control plane
- Worker nodes (EC2 or Fargate)
- Standard Kubernetes objects (Pods, Deployments)
- Compatible with kubectl and Kubernetes tools

8.0.14 Monitoring and Logging Services Comparison

Service & Purpose & What It Monitors & Data Retention & Use Case & Pricing
CloudWatch & Monitoring and observability & Metrics, logs, events, alarms & 15 months (metrics)
CloudTrail & API activity logging & All API calls in AWS account & 90 days (event history), inde
Config & Resource configuration tracking & AWS resource configurations & Configurable & Compli
X-Ray & Distributed tracing & Application performance, requests & 30 days & Debugging, perform
VPC Flow Logs & Network traffic logging & Network interfaces in VPC & S3 or CloudWatch & N
EventBridge & Event bus & Events from AWS, SaaS, custom & N/A (routing) & Event-driven arc

Monitoring Service Details

CloudWatch:

- **Metrics**: CPU, disk, network for EC2, RDS, etc.
- **Logs**: Application logs, system logs

- **Alarms**: Trigger actions based on metrics
- **Dashboards**: Visualize metrics
- **Events/EventBridge**: React to state changes
- **Insights**: Query and analyze log data
- **Synthetics**: Monitor endpoints and APIs
- Default vs. detailed monitoring

CloudTrail:

- Logs every API call in your account
- Who, what, when, where for all actions
- Essential for security and compliance
- Integrates with CloudWatch Logs for alerts
- Multi-region and organization trails
- Integrity validation with log file hashing

Config:

- Records configuration changes over time
- Configuration history for resources
- Compliance checking with Config Rules
- Integrates with Systems Manager for remediation
- Relationship tracking between resources
- Multi-account, multi-region support

X-Ray:

- End-to-end request tracing
- Service map visualization
- Identify performance bottlenecks
- Track requests across microservices
- Integrates with Lambda, ECS, Elastic Beanstalk
- Annotations and metadata for filtering

Monitoring Decision Guide

Use CloudWatch when:

- Need performance metrics and alarms
- Monitoring resource utilization
- Setting up automated responses to metrics
- Collecting and analyzing logs

- Creating operational dashboards

Use CloudTrail when:

- Security auditing required
- Compliance and governance needs
- Tracking who did what in AWS
- Investigating security incidents
- Forensic analysis of API activity

Use Config when:

- Need resource inventory
- Tracking configuration changes over time
- Compliance auditing against rules
- Understanding resource relationships
- Point-in-time configuration history

Use X-Ray when:

- Debugging distributed applications
- Analyzing microservices performance
- Identifying latency issues
- Tracing requests across services
- Understanding application behavior

8.0.15 AI/ML Services Comparison

Service & **Type** & **Use Case** & **Expertise Required** & **Training Required** & **Pricing**
SageMaker & Full ML platform & Custom ML models, training, deployment & High (data scientist
Rekognition & Computer vision & Image/video analysis, face detection & Low (API calls) & No, p
Comprehend & NLP & Text analysis, sentiment, entities & Low (API calls) & No, pre-trained & P
Lex & Conversational AI & Chatbots, voice assistants & Medium & Yes, train intents & Per request
Polly & Text-to-speech & Convert text to lifelike speech & Low (API calls) & No, pre-trained & Per
Transcribe & Speech-to-text & Convert audio to text & Low (API calls) & No, pre-trained & Per s
Translate & Translation & Real-time text translation & Low (API calls) & No, pre-trained & Per c
Forecast & Time-series & Time-series forecasting & Medium & Yes, train on your data & Per foreca
Personalize & Recommendations & Recommendation engines & Medium & Yes, train on your data
Textract & Document analysis & Extract text from documents & Low (API calls) & No, pre-traine
Kendra & Intelligent search & Enterprise search with NLP & Low & Yes, index documents & Per h

AI/ML Service Details

SageMaker (Custom ML):

- Build, train, and deploy custom ML models
- Jupyter notebooks for development
- Built-in algorithms and frameworks
- AutoML with Autopilot
- Model monitoring and debugging
- For data scientists and ML engineers

Rekognition (Computer Vision):

- Object and scene detection
- Facial analysis and recognition
- Text in images (OCR)
- Content moderation
- Celebrity recognition
- Video analysis

Comprehend (Natural Language Processing):

- Sentiment analysis (positive, negative, neutral)
- Entity recognition (people, places, dates)
- Key phrase extraction
- Language detection
- Topic modeling
- PII detection

Lex (Conversational Interfaces):

- Build chatbots and voice assistants
- Powers Amazon Alexa
- Natural language understanding
- Multi-turn conversations
- Integration with Lambda for fulfillment
- Deploy to mobile, web, messaging platforms

Polly (Text-to-Speech):

- 60+ voices in 30+ languages
- Neural TTS for most natural voice
- SSML support for customization
- Speech marks for lip-syncing

- Lexicons for pronunciation
- Ideal for accessibility, IVR systems

Transcribe (Speech-to-Text):

- Real-time and batch transcription
- Speaker identification
- Custom vocabulary
- Automatic language identification
- Channel identification
- Medical and call analytics versions

AI/ML Selection Guide

Use SageMaker when:

- Building custom ML models
- Have data science team
- Need full control over training
- Complex ML requirements
- Want to use specific algorithms

Use Pre-trained AI Services when:

- No ML expertise required
- Quick integration via API
- Standard use cases covered
- Cost-effective for most scenarios
- Don't want to manage training

By Use Case:

- Analyze images/videos → Rekognition
- Analyze text sentiment → Comprehend
- Build chatbot → Lex
- Text to speech → Polly
- Speech to text → Transcribe
- Translate languages → Translate
- Extract from documents → Textract
- Product recommendations → Personalize
- Sales forecasting → Forecast
- Enterprise search → Kendra
- Custom ML → SageMaker

8.0.16 Enhanced Decision Trees

Networking Architecture Decision Tree

```
What do you need to connect?

AWS Resources to Internet
 From public subnet → Internet Gateway
 From private subnet → NAT Gateway (or NAT Instance)
 Private access to AWS services → VPC Endpoints

Multiple VPCs
 2 VPCs → VPC Peering
 3+ VPCs → Transit Gateway
 Shared services to many VPCs → PrivateLink

On-Premises to AWS
 Quick setup, encrypted → VPN Gateway
 Consistent bandwidth, private → Direct Connect
 Redundancy needed → Multiple VPN or Direct Connect + VPN
 Low latency required → Direct Connect

Content Delivery
 Static content globally → CloudFront + S3
 Dynamic content → CloudFront + ALB
 DNS routing → Route 53

Load Balancing
 HTTP/HTTPS → Application Load Balancer (ALB)
 TCP/UDP, extreme performance → Network Load Balancer (NLB)
 Third-party appliances → Gateway Load Balancer
 Legacy (avoid) → Classic Load Balancer

Security
 Web application protection → WAF + ALB/CloudFront
 DDoS protection → Shield (Standard/Advanced)
 Network ACLs → Subnet-level firewall (stateless)
 Security Groups → Instance-level firewall (stateful)
```

Disaster Recovery Strategy Selector

```
What's your RTO/RPO requirement?

RTO: Minutes, RPO: Seconds
 Multi-Site Active/Active
   - Hot standby in multiple regions
   - Highest cost, immediate failover
   - Use: Route 53 health checks, Aurora Global Database
```

```
   - Cost: 200\% of single site

RTO: < 1 hour, RPO: Minutes
 Warm Standby
   - Scaled-down version running in DR region
   - Can scale up quickly
   - Use: Scaled-down EC2, read replicas promoted
   - Cost: 50-100\% of single site

RTO: Hours, RPO: Minutes to Hours
 Pilot Light
   - Core components always running (e.g., databases)
   - Full environment provisioned on failover
   - Use: RDS standby, data replicated to S3
   - Cost: 20-50\% of single site

RTO: Days, RPO: Hours
 Backup and Restore
   - Regular backups to S3/Glacier
   - Restore from backup on disaster
   - Use: AWS Backup, S3 cross-region replication
   - Cost: < 10\% of single site (storage only)

Implementation Patterns:
- Database replication → RDS Multi-AZ, Aurora replicas
- Data replication → S3 Cross-Region Replication
- Infrastructure as Code → CloudFormation for quick rebuild
- Traffic management → Route 53 health checks and failover
- Continuous sync → DataSync, Storage Gateway
```

Data Analytics Platform Selector

```
What type of analytics?

Real-Time Analytics
 Streaming data → Kinesis Data Streams + Analytics
 Real-time dashboards → QuickSight with SPICE
 ML on streams → Kinesis + Lambda + SageMaker

Batch Analytics
 SQL on S3 data → Athena
 Complex ETL → Glue
 Large-scale processing → EMR (Spark/Hadoop)
 Data warehouse → Redshift

Interactive Exploration
 Ad-hoc queries → Athena
 Notebook-based → SageMaker notebooks or EMR notebooks
```

```
 Quick insights → QuickSight

Business Intelligence
 Dashboards and reports → QuickSight
 Complex queries → Redshift + QuickSight
 Self-service BI → QuickSight with datasets

Data Preparation
 Visual ETL → Glue DataBrew
 Code-based ETL → Glue (PySpark)
 Schema discovery → Glue Crawlers
 Data catalog → Glue Data Catalog

Machine Learning
 Feature engineering → Glue, SageMaker Data Wrangler
 Model training → SageMaker
 ML on big data → EMR with Spark MLlib
 AutoML → SageMaker Autopilot

Data Volume Considerations:
- < 1 TB → Athena, RDS with analytics
- 1-100 TB → Redshift
- 100+ TB → Redshift, EMR, or S3 data lake with Athena
- Petabyte scale → EMR, S3 data lake
```

Container Deployment Decision Flowchart

```
Do you need Kubernetes?
 Yes
   Want to manage nodes → EKS with EC2
   Serverless → EKS with Fargate

 No
    Want to manage hosts → ECS with EC2
    Serverless → ECS with Fargate or just Lambda

How much control do you need?
 Full OS control → EC2 with Docker (not orchestrated)
 Container orchestration → ECS or EKS
 Just run code → Lambda

How long does it run?
 < 15 minutes, event-driven → Lambda
 Continuous → ECS/EKS with Fargate or EC2
 Batch jobs → AWS Batch

Team expertise?
 Know Kubernetes well → EKS
```

```
 AWS experience → ECS
 Want simplest → Fargate with ECS
 Minimal DevOps → Lambda or Elastic Beanstalk

Cost optimization?
 Spot Instances → ECS/EKS with EC2 Spot
 Reserved Instances → ECS/EKS with EC2 RIs
 Pay per use → Fargate or Lambda
 Savings Plans → Fargate with Compute Savings Plans

Deployment pattern:
ECS with EC2:
- Register container instances to cluster
- Define task definitions
- Create services with desired count
- Use Application Load Balancer

ECS with Fargate:
- No instances to manage
- Define task definitions (CPU/memory)
- Create services
- AWS handles infrastructure

EKS with EC2:
- Managed Kubernetes control plane
- Self-managed worker nodes
- Standard kubectl commands
- Use Kubernetes services/ingress

EKS with Fargate:
- No nodes to manage
- Define Fargate profiles
- Deploy Kubernetes pods
- AWS provisions compute
```

Monitoring Solution Selector

```
What do you need to monitor?

Application Performance
 Metrics and alarms → CloudWatch
 Distributed tracing → X-Ray
 Real user monitoring → CloudWatch RUM
 Synthetic monitoring → CloudWatch Synthetics

Security and Compliance
 API activity → CloudTrail
 Configuration changes → Config
```

```
 Threat detection → GuardDuty
 Vulnerability scanning → Inspector
 Centralized security → Security Hub

Resource Changes
 Configuration history → Config
 Change notifications → EventBridge
 Drift detection → CloudFormation

Network Traffic
 VPC traffic → VPC Flow Logs
 DNS queries → Route 53 Resolver Query Logs
 CloudFront access → CloudFront Logs

Cost Monitoring
 Cost tracking → Cost Explorer
 Budget alerts → AWS Budgets
 Anomaly detection → Cost Anomaly Detection
 Rightsizing → Compute Optimizer

Logging Strategy:
Application Logs → CloudWatch Logs
Access Logs → S3, CloudWatch Logs
Audit Logs → CloudTrail to S3
Flow Logs → CloudWatch Logs or S3

Monitoring Architecture:
1. Collect: CloudWatch Agent, X-Ray SDK, CloudTrail
2. Store: CloudWatch Logs, S3, CloudWatch Metrics
3. Analyze: CloudWatch Insights, Athena on S3
4. Alert: CloudWatch Alarms, SNS, EventBridge
5. Visualize: CloudWatch Dashboards, QuickSight
```

8.0.17 Cost Comparison Matrices

Storage Cost Comparison (per GB-month)

Storage Type	Hot/Frequent	Warm/Infrequent	Cold/Archive	Use Case
S3 Standard	$0.023	-	-	Active data, frequent access
S3 Intelligent-Tiering	$0.023 + $0.0025 monitoring	Auto-moves	Auto-moves	Unknown a
S3 Standard-IA	-	$0.0125 + $0.01/GB retrieval	-	Infrequent but fast access
S3 One Zone-IA	-	$0.01 + $0.01/GB retrieval	-	Non-critical infrequent
S3 Glacier Instant	-	$0.004 + $0.03/GB retrieval	-	Archive, instant retrieval
S3 Glacier Flexible	-	-	$0.0036 + retrieval costs	Archive, 1-5 min retrieval
S3 Glacier Deep Archive	-	-	$0.00099 + retrieval costs	Long-term archive, 12 hrs
EBS gp3	$0.08	-	-	General purpose SSD

EBS io2 & $0.125 + IOPS cost & - & - & High performance SSD
EBS st1 & $0.045 & - & - & Throughput optimized HDD
EBS sc1 & $0.015 & - & - & Cold HDD
EFS Standard & $0.30 & - & - & Shared file system
EFS IA & - & $0.025 & - & Infrequent access files
FSx for Windows & $0.13 & - & $0.013 (backup) & Windows file shares
FSx for Lustre & $0.145-0.240 & - & - & HPC workloads

Storage Cost Optimization Tips:

- Use S3 Lifecycle policies to move to cheaper tiers
- Enable S3 Intelligent-Tiering for unpredictable access
- Use EBS snapshots to S3 for backups (cheaper than keeping unused volumes)
- Delete unused EBS volumes
- Use EFS IA for files not accessed in 30+ days

Database Cost Comparison (per hour)

Database	Small Instance	Medium Instance	Large Instance	Storage	Notes
RDS MySQL db.t3.micro	$0.017	-	-	$0.115/GB-month	General purpose
RDS MySQL db.t3.medium	-	$0.068	-	$0.115/GB-month	
RDS MySQL db.m5.large	-	-	$0.188	$0.115/GB-month	
Aurora MySQL	-	$0.082 (db.t3.medium)	$0.29 (db.r5.large)	$0.10/GB-month	Auto-sc
Aurora Serverless v2	-	-	-	$0.12/ACU-hour + $0.10/GB	Auto-scales, min 0.5 ACU
DynamoDB On-Demand	-	-	-	$1.25/million writes, $0.25/million reads	Pay per reque
DynamoDB Provisioned	-	-	-	$0.00065/WCU-hour, $0.00013/RCU-hour	Reserved cap
Redshift dc2.large	-	$0.25	-	160 GB SSD included	Min 1 node
Redshift ra3.xlplus	-	-	$1.086	Managed storage $0.024/GB	Scalable storage
ElastiCache t3.micro	$0.017	-	-	Included	Redis or Memcached
ElastiCache r5.large	-	-	$0.243	Included	Memory-optimized
DocumentDB r5.large	-	-	$0.277	$0.10/GB-month	MongoDB compatible
Neptune db.t3.medium	-	$0.073	-	$0.10/GB-month	Graph database

Database Cost Optimization:

- Use Aurora Serverless v2 for variable workloads
- Reserved Instances for RDS (up to 69% savings)
- DynamoDB On-Demand for unpredictable traffic
- DynamoDB Provisioned with auto-scaling for predictable
- Stop dev/test RDS instances when not in use
- Use read replicas instead of larger instances

Compute Pricing Comparison (per hour)

Instance Type	On-Demand	Spot (avg 70% off)	1-Year Reserved	3-Year Reserve
t3.micro	$0.0104	~$0.003	$0.0062 (40% off)	$0.0042 (60% off) & Similar to RI
t3.medium	$0.0416	~$0.012	$0.0250 (40% off)	$0.0166 (60% off) & Similar to RI
m5.large	$0.096	~$0.029	$0.058 (40% off)	$0.038 (60% off) & Similar to RI
m5.xlarge	$0.192	~$0.058	$0.115 (40% off)	$0.077 (60% off) & Similar to RI
c5.xlarge	$0.17	~$0.051	$0.102 (40% off)	$0.068 (60% off) & Similar to RI
r5.xlarge	$0.252	~$0.076	$0.151 (40% off)	$0.101 (60% off) & Similar to RI
Lambda	$0.20/1M requests + $0.0000166667/GB-sec	-	-	- & Compute Savings Plan
Fargate vCPU	$0.04048/vCPU-hour	-	-	- & Compute Savings Plan (up to 50% off)
Fargate Memory	$0.004445/GB-hour	-	-	- & Compute Savings Plan

Compute Cost Optimization:

- Spot Instances for fault-tolerant workloads (up to 90% savings)
- Reserved Instances for steady-state workloads (up to 72% savings)
- Savings Plans for flexible commitment (up to 72% savings)
- Right-size instances using Compute Optimizer
- Use Auto Scaling to match capacity to demand
- Lambda for event-driven (no idle costs)
- Schedule start/stop for dev/test environments

Data Transfer Cost Scenarios (per GB)

Scenario	Cost	Notes
Data IN to AWS (from internet)	FREE	All inbound data transfer
Data OUT from AWS to internet	$0.09 (first 10 TB)	Decreases with volume
Data OUT from AWS to internet	$0.085 (next 40 TB)	10-50 TB tier
Data OUT from AWS to internet	$0.07 (next 100 TB)	50-150 TB tier
Between S3 and EC2 (same region)	FREE	No data transfer charges
Between S3 and CloudFront	FREE	CloudFront to S3 in same region
CloudFront to internet	$0.085	Lower than direct from region
Between regions	$0.02	Inter-region data transfer
Cross-AZ within region	$0.01 (each direction)	$0.02 round trip
Between VPC peers (same region)	$0.01 (each direction)	Same as cross-AZ
Direct Connect (DX) data OUT	$0.02-0.03	Lower than internet
Data OUT via NAT Gateway	$0.045 + internet OUT costs	NAT Gateway processing fee

Data Transfer Cost Optimization:

- Use CloudFront for content delivery (lower egress costs)
- Keep resources in same AZ when possible

- Use VPC endpoints for S3/DynamoDB (no NAT Gateway or internet charges)
- Compress data before transfer
- Use Direct Connect for large data transfers
- Leverage S3 Transfer Acceleration for uploads
- Use AWS DataSync for migrations (no additional transfer fee)

Regional Cost Variations

Note: Prices vary by region. Examples above are for US East (N. Virginia).

Region & **Cost Multiplier** & **Notes**
US East (N. Virginia) & 1.0x & Baseline, usually cheapest
US West (Oregon) & 1.0x & Similar to US East
EU (Ireland) & ˜1.1x & Slightly higher
Asia Pacific (Singapore) & ˜1.15x & Higher than US
Asia Pacific (Sydney) & ˜1.2x & Higher data transfer costs
South America (S˜ao Paulo) & ˜1.5x & Highest costs generally

Cost Optimization by Region:

- Deploy in US East (N. Virginia) for lowest costs
- Choose region closest to users for performance
- Consider data sovereignty requirements
- Use S3 Intelligent-Tiering across regions

8.0.18 When to Use What - Detailed Scenarios

Compute Service Scenarios

Lambda - Perfect For:

- Image thumbnail generation when uploaded to S3
- Processing DynamoDB streams
- Scheduled tasks (every 5 minutes, hourly, daily)
- API backends with API Gateway
- Real-time file processing
- Chatbots and Alexa skills
- IoT backend processing
- ETL jobs under 15 minutes

Lambda - NOT Good For:

- Long-running processes (¿ 15 min)

- Applications with consistent traffic (better to use EC2/containers)
- Processes requiring local storage ¿ 10 GB
- Applications needing GPUs
- Legacy applications with complex dependencies

EC2 - Perfect For:

- Custom applications requiring specific OS
- Applications needing GPUs
- Lift-and-shift migrations
- Consistent workloads (use Reserved Instances)
- Applications with licensing restrictions
- High-performance computing clusters
- Windows-based applications

EC2 - NOT Good For:

- Simple event-driven functions (use Lambda)
- Unpredictable traffic spikes (unless using Auto Scaling)
- Microservices that could be containerized
- When you want zero administration

Fargate - Perfect For:

- Microservices architectures
- Long-running containers
- Batch jobs and scheduled tasks
- CI/CD agents
- Applications with variable load
- When you don't want to manage servers

Fargate - NOT Good For:

- Consistent high workloads (EC2 with containers cheaper)
- GPU workloads (use ECS with EC2)
- Windows containers requiring host customization
- Extremely latency-sensitive apps (cold starts)

Storage Service Scenarios

S3 Standard - Perfect For:

- Active website content and assets
- Frequently accessed data
- Content distribution with CloudFront
- Data lake hot tier
- Mobile and gaming applications
- Big data analytics (frequently queried)

S3 Standard-IA - Perfect For:

- Backups accessed occasionally
- Disaster recovery files
- Data retained for compliance
- Old data accessed few times per month
- Long-lived but infrequently accessed

S3 Glacier - Perfect For:

- Digital preservation
- Archive data rarely accessed
- Tape replacement
- Regulatory archives
- Long-term backups

S3 Glacier - NOT Good For:

- Frequently accessed data (expensive retrievals)
- Data needed immediately (retrieval time)
- Small files (minimum storage duration charges)

EFS - Perfect For:

- Web serving and content management
- Container persistent storage
- Shared data between instances
- Big data analytics needing shared access
- ML training data accessed by multiple instances
- Lift-and-shift applications expecting NFS

EFS - NOT Good For:

- Single-instance workloads (EBS cheaper)
- Windows applications (use FSx for Windows)
- Highest IOPS needs (use EBS io2)
- Object storage needs (use S3)

Database Service Scenarios

RDS - Perfect For:

- Traditional relational databases
- Existing applications using MySQL/PostgreSQL/Oracle/SQL Server
- OLTP workloads
- When you need Multi-AZ for HA
- Read replicas for read-heavy workloads
- Lift-and-shift of existing databases

RDS - NOT Good For:

- Massive scale (consider DynamoDB or Aurora)
- Need root access (use EC2 with self-managed DB)
- Simple key-value access (use DynamoDB)
- Frequent schema changes (consider NoSQL)

Aurora - Perfect For:

- High-performance OLTP
- Applications needing MySQL/PostgreSQL compatibility
- Global applications (Global Database)
- Variable workloads (Serverless v2)
- Read-heavy workloads (15 read replicas)
- When you need best RDS performance

DynamoDB - Perfect For:

- Serverless applications
- Mobile backends
- Gaming leaderboards
- Session stores
- Shopping carts
- IoT data
- Millisecond latency required
- Massive scale (millions of requests/sec)

DynamoDB - NOT Good For:

- Complex queries with joins
- Ad-hoc queries and analytics (use RDS or Athena)
- ACID transactions across items (use RDS)
- When you need SQL

- Applications with unpredictable query patterns

Redshift - Perfect For:

- Business intelligence
- OLAP workloads
- Complex queries across large datasets
- Data warehouse consolidation
- Historical data analysis
- Petabyte-scale analytics

Redshift - NOT Good For:

- OLTP workloads (use RDS or DynamoDB)
- Real-time data ingestion (use Kinesis)
- Small datasets ¡ 100 GB (use RDS or Athena)
- High-concurrency workloads (use RDS)

Networking Service Scenarios

CloudFront - Perfect For:

- Global content delivery
- Website acceleration
- Video streaming
- API acceleration
- Software distribution
- DDoS protection with Shield

CloudFront - NOT Good For:

- Same-region traffic (no benefit)
- Dynamic content that can't be cached
- Backend-to-backend communication

Direct Connect - Perfect For:

- Large data migrations to AWS
- Consistent bandwidth requirements
- Hybrid architectures with high throughput
- Compliance requiring private connectivity
- Cost reduction for high data transfer volumes

Direct Connect - NOT Good For:

- Quick setup needs (use VPN)
- Low data transfer volumes
- Temporary connections
- Budget-constrained projects

VPN - Perfect For:

- Quick hybrid connectivity
- Backup for Direct Connect
- Low to moderate traffic
- Encrypted connectivity
- Branch office connections

VPN - NOT Good For:

- High bandwidth requirements
- Latency-sensitive applications
- Very high data transfer volumes

Security Service Scenarios

IAM - Perfect For:

- Internal AWS access control
- Service-to-service permissions
- Cross-account access
- Temporary credentials with roles

IAM - NOT Good For:

- External user authentication (use Cognito)
- Thousands of users (use federation)

Cognito - Perfect For:

- Mobile app authentication
- Web app user sign-up/sign-in
- Social identity federation
- Guest user access
- User profile management

Cognito - NOT Good For:

- AWS resource access control (use IAM)
- Internal employee access (use SSO)

Secrets Manager - Perfect For:

- Database credentials
- API keys
- Automatic rotation of secrets
- Cross-region secret replication
- Applications using AWS SDKs

Secrets Manager - NOT Good For:

- Configuration data (use Parameter Store)
- Public data (use Parameter Store)
- When you don't need rotation (Parameter Store cheaper)

8.0.19 Service Anti-Patterns (When NOT to Use)

Common Mistakes and Anti-Patterns

Lambda Anti-Patterns:

- Using Lambda for long-running processes (use ECS/Fargate)
- Storing state in /tmp (use S3 or DynamoDB)
- Synchronous invocation chains (use Step Functions)
- Not using connection pooling for databases
- Ignoring cold start optimization

S3 Anti-Patterns:

- Using S3 as a database (use DynamoDB or RDS)
- Frequent small file updates (use EFS or database)
- Sequential file naming causing hot partitions
- Not using lifecycle policies for old data
- Public buckets without careful review

EC2 Anti-Patterns:

- Not using Auto Scaling for variable workloads
- On-Demand instances for predictable workloads (use RIs)
- Over-provisioning instance sizes
- Not using EBS snapshots for backups
- Forgetting to terminate unused instances

RDS Anti-Patterns:

- Not using Multi-AZ for production

- Single large instance instead of read replicas
- Not monitoring storage and IOPS
- Storing files/blobs in database (use S3)
- Not using parameter groups for tuning

DynamoDB Anti-Patterns:

- Using it as a relational database with complex joins
- Hot partition keys
- Storing large items ¿ 400 KB
- Not using DynamoDB Streams for replication
- Provisioned capacity with unpredictable traffic

VPC Anti-Patterns:

- Overlapping CIDR blocks preventing peering
- Too small CIDR ranges
- Not using multiple AZs for HA
- Public subnets for databases
- Not using VPC endpoints for S3/DynamoDB

ECS/EKS Anti-Patterns:

- Using Fargate for consistent high-volume (EC2 cheaper)
- Not using Application Load Balancer for HTTP services
- Ignoring container health checks
- Large monolithic containers
- Not using ECR for private images

8.0.20 Common Misconceptions

Clearing Up Confusion

S3 Misconceptions:

- "S3 is just for backups" - FALSE: It's for any object storage, including active applications
- "S3 is slow" - FALSE: Can handle thousands of requests per second per prefix
- "You need to manage S3 servers" - FALSE: Fully serverless and managed

Lambda Misconceptions:

- "Lambda is always cheaper" - FALSE: Consistent traffic makes EC2/Fargate cheaper

- "Lambda scales infinitely" - FALSE: Limited by concurrent execution limit (can be increased)
- "Lambda is only for simple functions" - FALSE: Can run complex applications

VPC Misconceptions:

- "All traffic between AZs is free" - FALSE: $0.01/GB each direction
- "Private subnet means secure" - FALSE: Still need Security Groups and NACLs
- "VPC peering is transitive" - FALSE: Must create direct peering between each VPC pair

RDS Misconceptions:

- "Multi-AZ means read scaling" - FALSE: It's only for HA, use read replicas for scaling
- "Automated backups are free" - FALSE: Charged for backup storage beyond DB size
- "All databases should use RDS" - FALSE: NoSQL workloads better on DynamoDB

IAM Misconceptions:

- "Root user should be used daily" - FALSE: Create IAM users and lock down root
- "IAM is free so roles don't matter" - FALSE: Proper roles crucial for security
- "Users need passwords" - FALSE: Service accounts should use roles, not users

8.0.21 Service Limits Extended

Soft Limits vs Hard Limits

Soft Limits (Can be increased by request):

- EC2 instance limits
- VPC limits
- RDS instance limits
- S3 bucket limit
- Lambda concurrent executions
- CloudFormation stacks
- Elastic IP addresses

Hard Limits (Cannot be increased):

- S3 object size (5 TB)
- Lambda execution timeout (15 minutes)

- Lambda /tmp storage (10 GB)
- Lambda deployment package size (250 MB)
- S3 bucket name length (63 characters)
- IAM policy size (6,144 characters for managed policies)

Extended Service Limits Table

Service	Limit Type	Default	Max After Increase	How to Request
EC2 On-Demand Instances	Soft	20 vCPUs	Unlimited	Service Quotas console
EC2 Spot Instances	Soft	20 vCPUs	Unlimited	Service Quotas console
VPCs per Region	Soft	5	100+	Service Quotas console
Subnets per VPC	Soft	200	200+	Service Quotas console
Security Groups per Region	Soft	2,500	10,000	Service Quotas console
Rules per Security Group	Soft	60	250	Service Quotas console
Route Tables per VPC	Soft	200	200+	Service Quotas console
S3 Buckets	Soft	100	1,000+	Support case
S3 Object Size	Hard	5 TB	Cannot increase	N/A
S3 PUT/COPY/POST/DELETE	Soft	3,500 req/sec	5,500+ req/sec	Automatic
S3 GET/HEAD	Soft	5,500 req/sec	Unlimited	Automatic
RDS DB Instances	Soft	40	100+	Service Quotas console
RDS Read Replicas per Master	Hard	15 (Aurora), 5 (others)	Cannot increase	N/A
RDS DB Snapshots	Soft	100	500+	Service Quotas console
DynamoDB Table Count	Soft	2,500	10,000+	Service Quotas console
DynamoDB Throughput	Soft	40,000 RCU, 40,000 WCU	Unlimited	Automatic with on-
Lambda Concurrent Executions	Soft	1,000	100,000+	Service Quotas console
Lambda Function Timeout	Hard	15 minutes	Cannot increase	N/A
Lambda /tmp Space	Hard	10 GB	Cannot increase	N/A
Lambda Deployment Package	Hard	250 MB (unzipped)	Cannot increase	N/A
Lambda Environment Variables	Hard	4 KB	Cannot increase	N/A
Lambda Layers	Hard	5 per function	Cannot increase	N/A
EBS Volumes per Instance	Soft	128	128	Cannot increase
EBS Volume Size (gp3/io2)	Hard	64 TB	Cannot increase	N/A
EBS Snapshots	Soft	10,000	100,000+	Service Quotas console
CloudFormation Stacks	Soft	2,000	10,000+	Service Quotas console
CloudFormation Stack Sets	Soft	100	500+	Service Quotas console
CloudWatch Alarms	Soft	5,000	15,000+	Service Quotas console
CloudWatch Log Groups	Soft	Unlimited	Unlimited	N/A
CloudTrail Trails	Soft	5	5	Cannot increase
IAM Users	Soft	5,000	5,000	Cannot increase
IAM Groups	Soft	300	500	Support case
IAM Roles	Soft	1,000	5,000+	Service Quotas console
IAM Policies per User/Group/Role	Hard	10 managed, 1 inline	Cannot increase	N/A
ELB (ALB/NLB) per Region	Soft	50	100+	Service Quotas console
Target Groups per Region	Soft	3,000	5,000+	Service Quotas console
Targets per Target Group	Soft	1,000	1,000+	Service Quotas console

VPN Connections per Region & Soft & 50 & 200+ & Service Quotas console
Direct Connect Connections & Soft & 10 & 50+ & Support case
Route 53 Hosted Zones & Soft & 500 & 5,000+ & Service Quotas console
Route 53 Records per Hosted Zone & Soft & 10,000 & 100,000+ & Service Quotas console

How to Request Service Limit Increases

Method 1: Service Quotas Console (Preferred)

1. Open Service Quotas console
2. Choose AWS service
3. Select quota to increase
4. Request quota increase
5. Provide use case justification

Method 2: Support Case

- For limits not in Service Quotas console
- Create support case under "Service Limit Increase"
- Provide detailed justification
- Include projected usage

Method 3: Automatic Increases

- Some services auto-increase based on usage (S3, DynamoDB On-Demand)
- No action required

Best Practices:

- Request increases proactively before hitting limits
- Monitor usage with CloudWatch and Service Quotas
- Set up CloudWatch alarms for approaching limits
- Design applications to handle throttling gracefully
- Use multiple accounts to multiply limits (via AWS Organizations)

Limits That Affect Architecture

Design Considerations: EC2 Instance Limits:

- Plan Auto Scaling max size within limits
- Use multiple instance types for diversity
- Consider Reserved Instances against limits

Lambda Concurrency:

- Reserve concurrency for critical functions

- Use SQS for buffering
- Design for throttling (exponential backoff)

DynamoDB Throughput:

- Use On-Demand for unpredictable traffic
- Provision with auto-scaling for predictable
- Partition data to avoid hot keys

S3 Request Rates:

- Use random prefixes for high request rates
- CloudFront for read-heavy workloads
- Transfer Acceleration for uploads

VPC Limits:

- Plan IP addressing carefully (CIDR sizing)
- Use Transit Gateway for many VPCs
- Monitor ENI usage (Lambda in VPC consumes ENIs)

IAM Limits:

- Use roles instead of users where possible
- Group permissions efficiently
- Consider AWS SSO for human users
- Federation for external identities

— ← Previous: Hands-On Labs — Back to Main — Next: Exam Scenarios →

Chapter 9

Chapter 9: Common Exam Scenarios and Real-World Solutions

9.0.1 Table of Contents

—

9.0.2 Scenario-Based Learning

Scenario 1: Cost Optimization for Predictable Workloads

Situation: A company runs a web application on EC2 instances that experiences predictable traffic Monday-Friday 9 AM-5 PM EST. Traffic is minimal on weekends and nights. **Current Setup**:

- 10 m5.large instances running 24/7
- On-Demand pricing
- Monthly cost: $1,200

Question: What's the MOST cost-effective solution? **Analysis**:

- Predictable schedule = opportunity for optimization
- Not running 24/7 = On-Demand might be wasteful
- Regular business hours = scheduled scaling
- Baseline capacity needed = Reserved Instances candidate

Recommended Solution:

1. **Purchase 3-year Standard Reserved Instances for 2-3 instances (baseline capacity)**

- Savings: Up to 75% on these instances

1. **Configure EC2 Auto Scaling with scheduled actions**:

- Scale up Monday-Friday 8:30 AM EST (before traffic starts)
- Scale down at 5:30 PM EST (after traffic ends)
- Minimum capacity on weekends: 2-3 instances

1. **Use On-Demand for peak periods during business hours**
2. **Store session data in ElastiCache or DynamoDB (not on instances)**

Expected Savings: 40-60% reduction in monthly costs —

Scenario 2: Designing for High Availability

Situation: An e-commerce company's application must remain available even if an entire Availability Zone fails. The application currently runs on a single EC2 instance with a MySQL database. **Question**: How should you architect this for high availability? **Current Problems**:

- Single point of failure (one EC2 instance)
- Database not redundant
- No automatic failover
- Session data tied to instance

Recommended Solution:

1. **Multi-AZ Application Tier**
   - Deploy **Application Load Balancer** spanning multiple AZs
   - Create **Auto Scaling group** with minimum 2 instances across different AZs
   - Set desired capacity based on traffic patterns
   - Configure health checks on ALB and Auto Scaling

2. **Multi-AZ Database**
   - Migrate MySQL to **Amazon RDS Multi-AZ**
   - Automatic failover to standby in different AZ
   - Synchronous replication
   - Minimal downtime during failover

3. **Stateless Application Design**
   - Store session data in **ElastiCache** (Redis with Multi-AZ)
   - Or use **DynamoDB** for session storage
   - Enable sticky sessions on ALB if needed (but prefer stateless)

4. **Static Assets**
   - Store in **S3** (automatically multi-AZ)
   - Use **CloudFront** for global distribution

5. **Monitoring**
   - Set up **CloudWatch alarms** for health checks
   - Configure **SNS notifications** for failures

Architecture Benefits:
   - Survives AZ failure
   - Automatic scaling for traffic spikes
   - Automatic failover for database
   - No single point of failure

—

Scenario 3: Large Data Migration

Situation: A healthcare company needs to migrate 80 TB of medical imaging data from on-premises storage to S3. Compliance requires data to be encrypted and migration completed within 2 weeks. **Constraints**:

- Internet connection: 100 Mbps
- Upload via internet would take: ˜74 days
- Deadline: 2 weeks
- Data must be encrypted
- HIPAA compliance required

Question: What's the best migration approach? **Analysis**:

- Data volume too large for internet upload
- Time constraint eliminates internet-based solutions
- Security and compliance requirements
- Need physical device for transfer

Recommended Solution:

1. Use AWS Snowball Edge Storage Optimized

- 80 TB usable capacity per device
- Order 1-2 devices (for redundancy)
- 256-bit encryption built-in
- HIPAA compliant

2. Migration Process

1. Order Snowball device via AWS Console
2. AWS ships device (2-3 days)
3. Connect to network, unlock with credentials
4. Copy data using Snowball client (2-4 days for 80 TB)
5. Ship device back to AWS (2-3 days)
6. AWS uploads to S3 (1-2 days)

3. S3 Configuration

- Enable **S3 server-side encryption (SSE-S3 or SSE-KMS)**
- Enable **versioning** for data protection
- Configure **lifecycle policies** to transition older data to Glacier
- Enable **S3 Object Lock** for compliance (WORM)

4. Compliance

- Use **AWS Artifact** to access HIPAA BAA
- Sign Business Associate Addendum (BAA)
- Enable **CloudTrail** for audit logging
- Use **AWS Config** for compliance monitoring

Timeline: 7-12 days (meets 2-week deadline)

Key Point

Tip: For data volumes ¿100 PB, use **AWS Snowmobile**

—

Scenario 4: Serverless Application Architecture

Situation: A startup wants to build a mobile app backend with REST API. They have limited DevOps resources and want to minimize operational overhead while paying only for actual usage. **Requirements**:

- REST API for mobile app
- User authentication
- Data storage
- Image storage
- Scalable to millions of users
- Minimal operational management
- Pay-per-use pricing

Question: What AWS services should they use? **Recommended Serverless Architecture**:

1. API Layer

- **Amazon API Gateway**: Create and manage REST API
- Features: Request throttling, API keys, caching, CORS
- Pay per million API calls

2. Compute Layer

- **AWS Lambda**: Run business logic without servers
- Languages: Node.js, Python, Java, Go, etc.
- Auto-scaling built-in
- Pay only for execution time

3. Authentication

- **Amazon Cognito**: User sign-up, sign-in, access control
- User pools for authentication
- Identity pools for AWS resource access
- Social identity providers (Facebook, Google)
- Free tier: 50,000 MAUs

4. Data Storage

- **Amazon DynamoDB**: NoSQL database
- Single-digit millisecond latency
- Automatic scaling
- On-demand or provisioned capacity
- Always-free tier: 25 GB storage

5. Image Storage

- **Amazon S3**: Store user-uploaded images
- Lifecycle policies to move old images to Glacier
- CloudFront for fast image delivery

6. Optional Enhancements

- **Amazon CloudFront**: CDN for API and static assets
- **AWS AppSync**: GraphQL API (alternative to API Gateway + Lambda)
- **Amazon SES**: Send transactional emails
- **Amazon SNS**: Push notifications to mobile devices

Benefits:

- Zero server management
- Automatic scaling from 0 to millions of users
- Pay only for actual usage
- High availability built-in
- Focus on application code, not infrastructure
- Fast deployment and iteration

Cost Example:

- 1 million API requests: ˜$3.50
- Lambda executions: ˜$0.20
- DynamoDB: ˜$1.25

- S3 storage (100 GB): ~$2.30
- **Total: ~$7.25/month for 1M requests**

—

Scenario 5: Compliance and Governance

Situation: A financial services company with 50 AWS accounts needs to ensure no S3 buckets are publicly accessible across the organization. They also need to track all changes and demonstrate compliance. **Requirements**:

- Enforce no public S3 buckets
- Apply to all accounts
- Monitor compliance continuously
- Audit all changes
- Automated remediation preferred

Question: How can they enforce and monitor this policy? **Recommended Solution**:

1. AWS Organizations Setup

- Group accounts using **Organizational Units (OUs)**
- Example structure: Production OU, Development OU, Test OU

2. Service Control Policies (SCPs)

- Create SCP denying `s3:PutBucketPublicAccessBlock` with value False
- Deny `s3:PutBucketPolicy` if it allows public access
- Apply to root or specific OUs
- SCPs define maximum permissions (even admins can't override)

3. S3 Block Public Access

- Enable **S3 Block Public Access** at organization level
- Applies to all accounts in organization
- Prevents accidental public exposure

4. Continuous Monitoring

- Enable **AWS Config** across all accounts
- Deploy **s3-bucket-public-read-prohibited** rule
- Deploy **s3-bucket-public-write-prohibited** rule
- Automatic compliance reporting

5. Automated Remediation

- Configure **AWS Config auto-remediation**
- Use AWS Systems Manager Automation documents
- Automatically disable public access when detected

6. Audit and Logging

- Enable **CloudTrail** in all accounts
- Centralize logs in dedicated security account
- Track all S3 API calls
- Set up **CloudWatch alarms** for policy violations

7. Centralized Security

- Use **AWS Security Hub** for centralized security view
- Aggregates findings from Config, GuardDuty, Inspector
- Compliance dashboards for standards (PCI DSS, CIS)

Additional Recommendations:

- Regular compliance reports using **AWS Artifact**
- Periodic access reviews
- Employee training on security best practices
- Implement least privilege IAM policies

—

Scenario 6: Disaster Recovery Strategy

Situation: An e-commerce company needs disaster recovery for their application. Their business requires:

- RPO (Recovery Point Objective): 1 hour
- RTO (Recovery Time Objective): 4 hours
- Currently running in us-east-1

Question: What DR strategy should they implement? **DR Strategy Options**:

Strategy	**RPO**	**RTO**	**Cost**	**Best For**
Backup and Restore	Hours to days	Hours to days	Lowest	Non-critical workloads
Pilot Light	Minutes to hours	Hours	Low-Medium	This scenario
Warm Standby	Seconds to minutes	Minutes	Medium-High	Critical applications
Multi-Site Active/Active	Near zero	Near zero	Highest	Mission-critical systems

Key Point

Recommendation: **Pilot Light** is the optimal strategy for this scenario, meeting the 1-hour RPO and 4-hour RTO requirements at a reasonable cost.

Recommended Pilot Light Implementation:

1. Data Replication

- Use **RDS cross-region read replicas**
- Replicate from us-east-1 to us-west-2
- Meets 1-hour RPO requirement

2. Application AMIs

- Regularly copy AMIs to DR region
- Keep AMIs up-to-date
- Automate with Lambda

3. Infrastructure as Code

- Use **CloudFormation templates**
- Pre-create VPC, subnets, security groups in DR region
- Keep Auto Scaling groups in DR region with 0 capacity

4. DNS Failover

- Use **Route 53 health checks**
- Configure failover routing policy
- Automatic DNS failover to DR region

5. Testing

- Quarterly DR drills
- Document runbooks
- Measure actual RTO/RPO

Failover Process:

1. Detect primary region failure (Route 53 health check)
2. Promote RDS read replica to master
3. Update CloudFormation stack to scale up Auto Scaling
4. Route 53 automatically redirects traffic
5. **Total time: ˜2-3 hours (meets 4-hour RTO)**

—

Scenario 7: Hybrid Cloud Connectivity

Situation: A manufacturing company wants to extend their on-premises data center to AWS while maintaining consistent network performance for their ERP system.
Requirements:

- Consistent network latency
- Private connection (no internet)
- Bandwidth: 1 Gbps
- Access to multiple VPCs

Connection Options Analysis:

Solution & **Pros** & **Cons** & **Best For**
Site-to-Site VPN & Quick setup (hours), low cost, encrypted & Variable latency, internet-based, li
AWS Direct Connect & Consistent performance, high bandwidth, private & Expensive, takes wee
Direct Connect + VPN & Best of both worlds & Most expensive, complex & Regulated industrie

Recommended Solution: AWS Direct Connect

1. Direct Connect Setup

- Order 1 Gbps Direct Connect port
- Work with AWS Direct Connect Partner
- Provision takes 2-4 weeks
- Set up cross-connect at colocation facility

2. Multiple VPC Access

- Use **Direct Connect Gateway**
- Connect to multiple VPCs across regions
- Single Direct Connect connection
- Simplifies connectivity

3. High Availability

- Order second Direct Connect connection (different location)
- Configure BGP for automatic failover
- Or use VPN as backup connection

4. Security

- Layer VPN over Direct Connect for encryption
- Or use **MACsec** encryption
- Private VIF for VPC access
- Public VIF for public AWS services

—

Scenario 8: Multi-Region Architecture for Global Application

Situation: A social media company is launching a new photo-sharing application that needs to serve users across North America, Europe, and Asia. They expect rapid growth and need to provide low-latency access to content while maintaining data consistency. **Current State**:

- Single-region deployment in us-east-1
- 200ms+ latency for users in Asia and Europe
- Customer complaints about slow image loading
- Growing user base: 100K users → 5M expected in 6 months

Requirements: **Functional Requirements**:

- Users can upload/view photos from any region
- Social features: likes, comments, follows
- User profile and settings
- Search functionality
- Mobile and web access

Non-Functional Requirements:

- Latency: ¡100ms for content delivery
- Availability: 99.95%
- Data residency compliance (GDPR for EU)
- RPO: 1 hour, RTO: 2 hours
- Support 10M concurrent users
- Cost-effective scaling

Question: How should they architect a multi-region solution? **Recommended Architecture**:

1. Global Content Delivery Amazon CloudFront:

- Deploy CloudFront distributions with edge locations worldwide
- Cache static assets (images, CSS, JavaScript)
- Regional edge caches for large files
- Configure custom origins pointing to regional endpoints
- Enable HTTP/2 and compression

Amazon S3:

- Create S3 buckets in each primary region (us-east-1, eu-west-1, ap-southeast-1)
- Enable S3 Transfer Acceleration for faster uploads
- Use S3 Intelligent-Tiering for automatic cost optimization
- Implement lifecycle policies for old content

Cross-Region Replication:

- Enable S3 Cross-Region Replication for disaster recovery
- Replicate photos bidirectionally between regions
- Use replication time control for predictable replication
- Replicate only active content (photos ¡30 days)

2. Database Architecture Amazon DynamoDB Global Tables:

- Deploy Global Tables across 3 regions
- Tables: Users, Posts, Likes, Comments, Follows
- Multi-master replication (writes to any region)
- Typical replication latency: ¡1 second
- Automatic conflict resolution (last writer wins)
- Use on-demand capacity for unpredictable traffic

Alternative: Amazon Aurora Global Database:

- If complex queries needed
- Primary region: us-east-1
- Read replicas in eu-west-1 and ap-southeast-1
- Lag: ¡1 second
- Failover: ¡1 minute
- Better for relational data and complex joins

Data Residency Compliance:

- Create separate DynamoDB tables for EU users
- Store EU user data only in eu-west-1
- Use IAM policies to enforce data boundaries
- Document data flow for GDPR compliance

3. API and Application Layer Amazon API Gateway:

- Deploy regional API Gateway endpoints
- Edge-optimized for CloudFront integration
- Custom domain names per region
- Request throttling and caching

AWS Lambda or ECS Fargate:

- Lambda for event-driven, sporadic workloads
- ECS Fargate for containerized applications
- Deploy in multiple AZs per region
- Auto Scaling based on request volume

4. Routing and Traffic Management Amazon Route 53:

- Create geolocation routing policy
- North America → us-east-1
- Europe → eu-west-1
- Asia → ap-southeast-1
- Configure health checks for failover
- Latency-based routing for optimal performance

Implementation:

```
User in Germany
→ Route 53 (geolocation: Europe)
→ CloudFront (Frankfurt edge)
→ API Gateway (eu-west-1)
→ Lambda/ECS (eu-west-1)
→ DynamoDB Global Table (eu-west-1)
→ S3 (eu-west-1) via CloudFront
```

5. Search Functionality Amazon OpenSearch Service:

- Deploy domain in each region
- Index user data and posts
- Cross-region snapshot for backup
- Or use Amazon CloudSearch

Alternative: Amazon Kendra:

- For intelligent search with ML
- Natural language queries

6. Monitoring and Operations Amazon CloudWatch:

- Cross-region dashboards
- Unified logging with CloudWatch Logs Insights
- Alarms for latency, errors, costs
- Custom metrics for business KPIs

AWS X-Ray:

- Distributed tracing across regions
- Identify bottlenecks
- Service map visualization

Step-by-Step Implementation: **Phase 1: Foundation (Weeks 1-2)**

1. Set up AWS Organizations and multi-account structure
2. Create VPCs in target regions
3. Deploy CloudFormation templates for infrastructure
4. Set up centralized logging and monitoring
5. Configure IAM roles and policies

Phase 2: Data Layer (Weeks 3-4)

1. Create DynamoDB Global Tables
2. Set up S3 buckets with replication
3. Configure Aurora Global Database (if chosen)
4. Test data replication and consistency
5. Implement backup strategies

Phase 3: Application Deployment (Weeks 5-6)

1. Deploy API Gateway in all regions
2. Deploy Lambda functions or ECS services
3. Configure Auto Scaling policies
4. Implement caching strategies
5. Set up CloudFront distributions

Phase 4: Routing and DNS (Week 7)

1. Configure Route 53 geolocation routing
2. Set up health checks and failover
3. Test routing from different regions
4. Configure SSL/TLS certificates

Phase 5: Testing and Optimization (Week 8)

1. Load testing from multiple regions
2. Latency measurements
3. Failover testing
4. Cost optimization
5. Security hardening

Cost Breakdown (Monthly Estimate for 5M users):

Service & **Configuration** & **Monthly Cost**
CloudFront & 10 TB data transfer, 100M requests & $850
S3 & 50 TB storage, Transfer Acceleration & $1,250
DynamoDB Global Tables & 1 billion requests, 500 GB & $1,800
Lambda & 500M requests, 1GB memory & $900
API Gateway & 500M requests & $1,750
Route 53 & Hosted zones, health checks & $100
CloudWatch & Logs, metrics, alarms & $250
Data Transfer & Cross-region replication & $450
Total & **˜$7,350/month** &

Cost Optimization Strategies:

1. Use S3 Intelligent-Tiering for automatic storage class transitions
2. Enable CloudFront compression to reduce data transfer
3. Implement DynamoDB on-demand pricing for variable workloads
4. Use reserved capacity for predictable base load
5. Set up AWS Budgets alerts
6. Archive old content to S3 Glacier

Benefits:

- **Performance**: ¡100ms latency worldwide
- **Availability**: 99.99% with multi-region failover
- **Scalability**: Seamlessly handles traffic spikes
- **Data Sovereignty**: GDPR compliance with regional data storage
- **User Experience**: Fast content delivery regardless of location
- **Business Continuity**: Automatic failover between regions

Trade-offs:

- **Complexity**: Managing multi-region infrastructure
- **Cost**: Higher than single-region deployment (3-4x)
- **Data Consistency**: Eventual consistency with Global Tables
- **Development**: More complex testing and deployment

- **Operational Overhead**: Multi-region monitoring and troubleshooting

Alternative Approaches: **Option 1: Hybrid Approach**

- Primary region with CloudFront for content delivery
- Lower cost but higher latency for writes
- Best for read-heavy applications

Option 2: Active-Passive Multi-Region

- Active region handles all traffic
- Passive region for disaster recovery only
- Lower cost, simpler but longer failover time

Option 3: Regional Isolation

- Completely separate deployments per region
- No data replication between regions
- Best for data residency requirements

Common Pitfalls to Avoid:

1. **Not testing failover**: Regularly practice region failover
2. **Ignoring data transfer costs**: Can be 30-40% of total costs
3. **Synchronous replication assumptions**: DynamoDB Global Tables are eventually consistent
4. **Over-engineering**: Start with 2 regions, expand as needed
5. **Neglecting monitoring**: Set up comprehensive CloudWatch dashboards early
6. **Hardcoded endpoints**: Use service discovery or configuration
7. **Ignoring data residency laws**: Consult legal team for compliance
8. **Not considering latency for writes**: Global Tables have ˜1s replication lag

—

Scenario 9: Security Incident Response and Prevention

Situation: A healthcare technology company experienced a security incident where an S3 bucket containing patient data was briefly exposed publicly. The CISO has mandated a comprehensive security overhaul to prevent future incidents and improve detection and response capabilities. **Current State**:

- 25 AWS accounts with inconsistent security practices
- No centralized security monitoring
- Manual security reviews

- Limited visibility into configuration changes
- Reactive security approach

Incident Impact:

- 10,000 patient records potentially exposed
- 4 hours until detection
- HIPAA violation investigation
- Reputation damage
- Potential fines up to $1.5M

Requirements: **Functional Requirements**:

- Detect security threats in real-time
- Prevent unauthorized access
- Automated incident response
- Continuous compliance monitoring
- Audit trail for all actions
- Encryption at rest and in transit

Non-Functional Requirements:

- Detection time: ¡5 minutes
- Automated response: ¡1 minute
- 100% configuration compliance
- 7-year log retention
- SOC 2, HIPAA compliance
- Zero trust architecture

Question: How should they implement comprehensive security controls? **Recommended Security Architecture**:

1. Detective Controls - Threat Detection Amazon GuardDuty:

- Enable in all accounts and regions
- Monitors VPC Flow Logs, CloudTrail, DNS logs
- ML-based anomaly detection
- Detects:
- Compromised EC2 instances (cryptocurrency mining)
- Reconnaissance activity
- Unauthorized access attempts
- Data exfiltration
- Malicious IP communications

AWS Security Hub:

- Centralized security dashboard
- Aggregates findings from:
- GuardDuty
- Amazon Inspector
- Amazon Macie
- IAM Access Analyzer
- AWS Config
- Third-party tools
- Compliance checks against:
- CIS AWS Foundations Benchmark
- PCI DSS
- HIPAA
- AWS Foundational Security Best Practices

Amazon Macie:

- Automated sensitive data discovery
- Scans S3 buckets for PII, PHI
- Machine learning classification
- Identifies:
- Credit card numbers
- Social Security numbers
- Patient health records
- API keys and secrets

AWS CloudTrail:

- Enable in all regions
- Record all API calls
- Multi-region trail
- Log file integrity validation
- Centralized logging to dedicated security account
- S3 bucket with MFA Delete enabled
- Lifecycle policy: 7-year retention

2. Preventive Controls - Access Management AWS Organizations with SCPs:

- Organizational hierarchy:
- Root
- Security OU
- Production OU
- Development OU
- Sandbox OU

Service Control Policies:

```
// Prevent disabling security services
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Deny",
      "Action": [
        "guardduty:DeleteDetector",
        "securityhub:DisableSecurityHub",
        "cloudtrail:StopLogging",
        "cloudtrail:DeleteTrail",
        "config:DeleteConfigRule",
        "config:StopConfigurationRecorder"
      ],
      "Resource": "*"
    \}
  ]
\}

// Enforce encryption
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Deny",
      "Action": "s3:PutObject",
      "Resource": "*",
      "Condition": \{
        "StringNotEquals": \{
          "s3:x-amz-server-side-encryption": [
            "AES256",
            "aws:kms"
          ]
        \}
      \}
    \}
  ]
\}
```

```
// Prevent public S3 access
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Deny",
      "Action": [
        "s3:PutAccountPublicAccessBlock"
      ],
      "Resource": "*",
      "Condition": \{
        "StringNotEquals": \{
          "s3:PublicAccessBlock": "true"
        \}
      \}
    \}
  ]
\}
```

IAM Access Analyzer:

- Continuously monitors IAM policies
- Identifies resources shared with external entities
- Validates policies against best practices
- Generates policy recommendations

AWS IAM Identity Center (SSO):

- Centralized user access management
- Multi-factor authentication mandatory
- Integration with corporate identity provider (Okta, Azure AD)
- Time-bound elevated access
- Attribute-based access control

3. Continuous Compliance Monitoring AWS Config:

- Enable in all regions and accounts
- Configuration recording for all resources
- Compliance rules:
- `s3-bucket-public-read-prohibited`
- `s3-bucket-public-write-prohibited`
- `s3-bucket-server-side-encryption-enabled`
- `rds-encryption-enabled`
- `ec2-encrypted-volumes`

- `cloudtrail-enabled`
- `multi-region-cloudtrail-enabled`
- `root-account-mfa-enabled`
- `iam-password-policy`
- `vpc-flow-logs-enabled`

AWS Config Aggregator:

- Centralized compliance view across all accounts
- Deployed in security account
- Cross-account access via IAM roles

4. Automated Incident Response AWS Lambda for Auto-Remediation: Scenario: S3 Bucket Made Public

```
\# Lambda function triggered by Config rule violation
import boto3

def lambda\_handler(event, context):
    s3 = boto3.client('s3')
    config = boto3.client('config')

    \# Extract bucket name from Config event
    bucket\_name = event['configRuleEvaluations'][0]['
       resourceId']

    \# Block all public access
    s3.put\_public\_access\_block(
        Bucket=bucket\_name,
        PublicAccessBlockConfiguration=\{
            'BlockPublicAcls': True,
            'IgnorePublicAcls': True,
            'BlockPublicPolicy': True,
            'RestrictPublicBuckets': True
        \}
    )

    \# Send SNS notification
    sns = boto3.client('sns')
    sns.publish(
        TopicArn='arn:aws:sns:us-east-1:123456789012:
           SecurityAlerts',
        Subject='SECURITY: Public S3 Bucket Auto-Remediated',
        Message=f'Bucket \{bucket\_name\} was made public and
           has been automatically secured.'
    )

    return \{
        'statusCode': 200,
```

```
        'body': f'Remediated public access for \{bucket\_name
            \}'
    \}
```

Amazon EventBridge Rules:

- Trigger Lambda on GuardDuty findings
- Automated responses:
- Isolate compromised EC2 instances (change security group)
- Revoke IAM user credentials
- Snapshot EBS volumes for forensics
- Block malicious IPs in NACLs

AWS Systems Manager Incident Manager:

- Automated incident response plans
- Escalation policies
- On-call schedules
- Post-incident analysis

5. Data Protection Encryption at Rest:

- S3: Default encryption with KMS
- EBS: Encrypted volumes mandatory
- RDS: Encryption enabled for all databases
- DynamoDB: Encryption enabled
- EFS: Encryption enabled

AWS Key Management Service (KMS):

- Customer-managed keys (CMKs)
- Automatic key rotation
- Key policies restricting access
- CloudTrail logging of key usage
- Separate keys per environment

Encryption in Transit:

- TLS 1.2+ for all communication
- AWS Certificate Manager for SSL/TLS
- VPC endpoints for private communication
- PrivateLink for service access

AWS Secrets Manager:

- Rotate database credentials automatically
- Store API keys and secrets
- Integration with RDS, Redshift, DocumentDB
- Audit secret access via CloudTrail

6. Network Security VPC Security:

- Private subnets for application and database tiers
- Public subnets only for load balancers
- VPC Flow Logs enabled (all VPCs)
- Network ACLs for subnet-level filtering

AWS Network Firewall:

- Stateful firewall at VPC level
- Intrusion prevention system (IPS)
- Block malicious domains
- Custom rule groups

AWS WAF (Web Application Firewall):

- Protect web applications from common exploits
- Managed rules:
- OWASP Top 10
- Known bad inputs
- SQL injection
- Cross-site scripting (XSS)
- Rate limiting
- Geo-blocking

Step-by-Step Implementation: **Phase 1: Foundation (Week 1)**

1. Enable CloudTrail in all accounts
2. Create security account
3. Set up centralized logging S3 bucket
4. Enable GuardDuty in all accounts/regions
5. Document current security posture

Phase 2: Detection and Monitoring (Week 2)

1. Enable Security Hub
2. Enable Macie for S3 scanning
3. Deploy Config with compliance rules

4. Set up Config Aggregator
5. Create CloudWatch dashboards

Phase 3: Preventive Controls (Week 3)

1. Implement SCPs in AWS Organizations
2. Enable S3 Block Public Access organization-wide
3. Deploy IAM Access Analyzer
4. Enforce MFA for all users
5. Implement IAM Identity Center

Phase 4: Automated Response (Week 4)

1. Create Lambda remediation functions
2. Set up EventBridge rules
3. Configure SNS topics for alerts
4. Deploy Systems Manager Incident Manager
5. Test automated responses

Phase 5: Data Protection (Week 5)

1. Enable default encryption on S3
2. Encrypt all EBS volumes
3. Deploy KMS CMKs with rotation
4. Migrate secrets to Secrets Manager
5. Implement AWS Backup

Phase 6: Testing and Validation (Week 6)

1. Conduct security drills
2. Penetration testing
3. Red team exercises
4. Update incident response playbooks
5. Train security team

Cost Breakdown (Monthly for 25 accounts):

Service & **Configuration** & **Monthly Cost**
GuardDuty & 25 accounts, 500GB VPC Flow Logs & $650
Security Hub & 25 accounts, 10K compliance checks & $200
Macie & 1TB S3 data scanned & $300
CloudTrail & Multi-region, 1M events & $50
Config & 25 accounts, 500 rules & $800
AWS WAF & 5 web ACLs, 100M requests & $150

KMS & 100 CMKs & $100
Lambda & Auto-remediation functions & $50
S3 & Log storage (1TB) & $25
Total & **˜$2,325/month** &

Benefits:

- **Rapid Detection**: Security threats detected in ¡5 minutes
- **Automated Response**: Incidents remediated in ¡1 minute
- **Compliance**: Continuous monitoring against standards
- **Visibility**: Centralized view of security posture
- **Prevention**: Proactive controls prevent incidents
- **Audit Trail**: Complete history for compliance
- **Cost of Prevention**: $2,325/month vs. potential $1.5M fine

Trade-offs:

- **Initial Complexity**: Setting up centralized security takes time
- **False Positives**: GuardDuty may flag legitimate activity
- **Operational Changes**: Teams must adapt to new security controls
- **Cost**: Ongoing security spend vs. risk mitigation

Alternative Approaches: **Option 1: Third-Party SIEM**

- Splunk, Sumo Logic, or Datadog
- More advanced analytics
- Higher cost
- Additional maintenance

Option 2: Manual Response

- Lower cost
- Slower response times
- Not recommended for compliance

Common Pitfalls to Avoid:

1. **Security Hub alert fatigue**: Start with critical findings only
2. **Not testing auto-remediation**: Test in dev first
3. **Overly restrictive SCPs**: Can block legitimate operations
4. **Ignoring GuardDuty findings**: Review and act on all findings
5. **No incident response plan**: Document procedures before incidents
6. **Single region deployment**: Enable security services in all regions
7. **No security training**: Educate developers on secure practices
8. **Forgetting about insider threats**: Monitor privileged user activity

—

Scenario 10: Modernizing Legacy Monolith Application

Situation: An insurance company runs a 15-year-old .NET Framework application on on-premises servers. The application handles policy management, claims processing, and customer portal. They want to migrate to AWS and modernize the architecture to improve scalability, reduce costs, and accelerate feature development. **Current State**:

- Monolithic .NET Framework 4.8 application
- SQL Server 2014 database (2TB)
- Windows Server 2012 R2
- 10 application servers behind hardware load balancer
- Peak load: 5,000 concurrent users
- Deployment: Manual, monthly releases, 4-hour downtime
- No automated testing
- Average response time: 2-3 seconds
- Annual infrastructure cost: $500K

Current Challenges:

- Slow development cycles
- Difficult to scale individual components
- High infrastructure costs
- Frequent production issues
- Aging technology stack
- Recruitment challenges (old tech)

Requirements: **Functional Requirements**:

- Migrate all functionality to AWS
- Maintain feature parity during migration
- Support existing integrations (SOAP, REST APIs)
- Preserve data integrity
- Windows authentication integration

Non-Functional Requirements:

- Zero downtime during migration
- Response time: ¡1 second
- 99.9% availability
- Support 10,000 concurrent users
- Reduce infrastructure costs by 40%
- Weekly deployments with zero downtime
- Automated testing and rollback

Question: How should they approach migration and modernization? **Recommended Migration Strategy: Strangler Fig Pattern**

Phase 1: Lift and Shift (Foundation) Step 1: Database Migration AWS Database Migration Service (DMS):

- Migrate SQL Server to Amazon RDS for SQL Server
- Minimal downtime using continuous replication
- Or migrate to RDS with Aurora PostgreSQL-Compatible (if licensing costs are high)

Database Configuration:

- RDS SQL Server Enterprise Edition
- Multi-AZ deployment for high availability
- db.r5.4xlarge (16 vCPU, 128 GB RAM)
- 3TB storage with Provisioned IOPS (10,000 IOPS)
- Automated backups (7-day retention)
- Automated patching in maintenance window

Migration Process:

1. Set up DMS replication instance
2. Create source endpoint (on-premises SQL Server)
3. Create target endpoint (RDS)
4. Create migration task with full load + CDC
5. Monitor replication lag
6. Perform cutover during low-traffic period

Step 2: Application Migration with App2Container AWS App2Container:

- Analyzes .NET Framework applications
- Creates container image
- Generates ECS task definitions
- Creates CloudFormation templates
- Minimal code changes required

Process:

1. Install App2Container on application server
2. Run inventory: `app2container inventory`
3. Analyze application: `app2container analyze --application-id <id>`
4. Customize deployment (app2container-config.json)
5. Generate artifacts: `app2container containerize`
6. Push to Amazon ECR

Step 3: Container Orchestration Amazon ECS on Fargate:

- Serverless compute for containers
- No EC2 instances to manage
- Automatic scaling
- Integrated with Application Load Balancer

ECS Configuration:

- Task definition:
- 4 vCPU, 8 GB memory per task
- Windows Server 2019 Core container
- Environment variables for configuration
- Secrets from AWS Secrets Manager
- Service:
- Desired count: 10 tasks
- Auto Scaling: 10-50 tasks based on CPU
- Spread across 3 AZs
- Health check grace period: 60 seconds

Phase 2: Modernization (Incremental) Strangler Fig Pattern Implementation:

1. Identify bounded contexts in monolith
2. Extract one service at a time
3. Route traffic to new service
4. Gradually replace monolith components

Priority Services to Extract: **1. Authentication Service**

- High reuse across features
- Extract first for shared use
- Technology: ASP.NET Core Web API
- Database: Amazon Aurora PostgreSQL
- Deployment: ECS Fargate

2. Claims Processing Service

- CPU-intensive
- Independent scaling needs
- Benefits from queue-based processing
- Technology: ASP.NET Core + AWS Lambda
- Queue: Amazon SQS
- Database: DynamoDB for claims status

3. Document Storage Service

- Large file uploads (claim documents, policy PDFs)
- Extract to reduce monolith load
- Technology: ASP.NET Core API
- Storage: Amazon S3
- OCR: Amazon Textract

4. Notification Service

- Email, SMS notifications
- High volume, sporadic
- Technology: AWS Lambda
- Email: Amazon SES
- SMS: Amazon SNS
- Queue: Amazon SQS

5. Reporting Service

- Resource-intensive queries
- Extract to dedicated read replica
- Technology: ASP.NET Core + Lambda
- Database: RDS read replica
- Caching: Amazon ElastiCache

Architecture Evolution:

```
Initial (Month 0-3):
Monolith (ECS) → RDS SQL Server

Phase 1 (Month 3-6):
ALB → Authentication Service (ECS)
    ↓
    → Monolith (ECS) → RDS SQL Server

Phase 2 (Month 6-9):
ALB → Authentication Service (ECS)
    → Claims Service (ECS + Lambda + SQS)
    → Monolith (ECS) → RDS SQL Server

Phase 3 (Month 9-12):
ALB → Authentication Service (ECS)
    → Claims Service (ECS + Lambda + SQS)
    → Document Service (ECS + S3 + Textract)
    → Notification Service (Lambda + SQS + SES/SNS)
    → Reporting Service (Lambda + ElastiCache)
    → Monolith (ECS) → RDS SQL Server (reduced functionality)
```

Phase 3: Supporting Infrastructure Caching Layer: **Amazon ElastiCache for Redis**:

- Cache frequently accessed data
- Session storage
- Reduce database load by 60%
- Configuration:
- cache.r5.large (2 nodes)
- Multi-AZ with automatic failover
- Encryption in transit and at rest

Application Load Balancer:

- Path-based routing
- Example routes:
- `/api/auth/*` → Authentication Service
- `/api/claims/*` → Claims Service
- `/api/documents/*` → Document Service
- `/*` → Monolith (default)
- Sticky sessions for monolith compatibility
- SSL/TLS termination
- WAF integration

API Gateway:

- For external partners accessing APIs
- Rate limiting and quotas
- API key management
- Request/response transformation
- CloudWatch logging

Observability: **AWS X-Ray**:

- Distributed tracing
- Identify performance bottlenecks
- Service map visualization
- Request flow analysis

Amazon CloudWatch:

- Centralized logging
- Custom metrics (business KPIs)
- Dashboards for each service

- Alarms for errors and latency

AWS CloudTrail:

- Audit trail for all API calls
- Compliance and security

Phase 4: CI/CD Pipeline AWS CodePipeline:

```
Source (CodeCommit)
  ↓
Build (CodeBuild)
  - Compile .NET Core
  - Run unit tests
  - Build Docker image
  - Push to ECR
  ↓
Test (CodeBuild)
  - Integration tests
  - Security scanning (Snyk, Aqua)
  ↓
Deploy to Dev (CodeDeploy + ECS)
  - Blue/green deployment
  - Smoke tests
  ↓
Manual Approval
  ↓
Deploy to Prod (CodeDeploy + ECS)
  - Blue/green deployment
  - Gradual traffic shifting (10\% → 50\% → 100\%)
  - Automatic rollback on errors
```

AWS CodeBuild buildspec.yml:

```
version: 0.2
phases:
  pre\_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region \$AWS\_DEFAULT\
        _REGION | docker login --username AWS --password-
        stdin \$AWS\_ACCOUNT\_ID.dkr.ecr.\$AWS\_DEFAULT\
        _REGION.amazonaws.com
  build:
    commands:
      - echo Build started on ‘date‘
      - echo Building the Docker image...
      - docker build -t \$IMAGE\_REPO\_NAME:\$IMAGE\_TAG .
```

```
      - docker tag \$IMAGE\_REPO\_NAME:\$IMAGE\_TAG \$AWS\
         _ACCOUNT\_ID.dkr.ecr.\$AWS\_DEFAULT\_REGION.
         amazonaws.com/\$IMAGE\_REPO\_NAME:\$IMAGE\_TAG
  post\_build:
    commands:
      - echo Build completed on ‘date‘
      - echo Pushing the Docker image...
      - docker push \$AWS\_ACCOUNT\_ID.dkr.ecr.\$AWS\_DEFAULT
         \_REGION.amazonaws.com/\$IMAGE\_REPO\_NAME:\$IMAGE\
         _TAG
      - echo Writing image definitions file...
      - printf '[\{"name":"app-container","imageUri":"\%s
         "\}]' \$AWS\_ACCOUNT\_ID.dkr.ecr.\$AWS\_DEFAULT\
         _REGION.amazonaws.com/\$IMAGE\_REPO\_NAME:\$IMAGE\
         _TAG > imagedefinitions.json
artifacts:
  files: imagedefinitions.json
```

Step-by-Step Implementation: **Phase 1: Preparation (Months 1-2)**

1. Assess application architecture
2. Identify dependencies and integrations
3. Set up AWS accounts and networking
4. Create migration plan
5. Train team on AWS services

Phase 2: Database Migration (Month 3)

1. Set up RDS instance
2. Test DMS replication
3. Migrate database with CDC
4. Verify data integrity
5. Update connection strings

Phase 3: Containerize Monolith (Month 4)

1. Use App2Container
2. Test containerized application
3. Deploy to ECS Fargate
4. Parallel run with on-premises
5. Gradual traffic shift (20% → 50% → 100%)

Phase 4: Extract Services (Months 5-12)

1. Extract authentication service (Month 5)
2. Extract claims service (Month 6-7)
3. Extract document service (Month 8-9)

4. Extract notification service (Month 10)
5. Extract reporting service (Month 11)
6. Decommission monolith components (Month 12)

Phase 5: Optimization (Ongoing)

1. Implement caching strategies
2. Optimize database queries
3. Right-size compute resources
4. Implement auto-scaling
5. Cost optimization

Cost Breakdown Comparison: **On-Premises (Annual)**:

- Hardware amortization: $200K
- Maintenance and support: $150K
- Datacenter costs: $100K
- Personnel (4 FTEs): $400K (partially allocated)
- **Total: $500K/year**

AWS Modernized Architecture (Annual):

Service & **Configuration** & **Monthly** & **Annual**
ECS Fargate & 30 tasks average, Windows & $3,600 & $43,200
RDS SQL Server & Multi-AZ, db.r5.4xlarge & $5,500 & $66,000
Application Load Balancer & 2 ALBs & $150 & $1,800
ElastiCache & Redis, 2 nodes & $250 & $3,000
S3 & 10 TB storage, requests & $300 & $3,600
Lambda & 10M requests & $200 & $2,400
CloudWatch & Logs, metrics & $400 & $4,800
Data Transfer & Outbound & $500 & $6,000
Total & **˜$10,900/month** & **˜$131K/year** &

Additional Costs:

- Migration tools and professional services: $50K (one-time)
- Training: $20K (one-time)

Total Year 1: $200K **Total Year 2+**: $131K/year **Savings**:

- Year 1: $300K (60% reduction)
- Year 2+: $369K (74% reduction)

Benefits:

- **Cost Reduction**: 74% infrastructure cost savings

- **Scalability**: Auto-scaling handles 2x traffic without manual intervention
- **Performance**: Response time reduced from 3s to ¡1s
- **Deployment Speed**: Monthly → weekly deployments
- **Availability**: 99.5% → 99.9%
- **Innovation**: Development team focuses on features, not infrastructure
- **Recruitment**: Modern tech stack attracts talent
- **Disaster Recovery**: Built-in with multi-AZ deployment

Trade-offs:

- **Migration Time**: 12-month project
- **Learning Curve**: Team must learn AWS, containers, microservices
- **Complexity**: Distributed systems more complex than monolith
- **Operational Changes**: New monitoring and deployment processes
- **Initial Investment**: Time and resources for migration

Alternative Approaches: **Option 1: Full Rewrite**

- Rebuild application from scratch
- Pros: Latest technology, clean architecture
- Cons: High risk, 2-3 years, expensive
- Recommendation: Avoid unless absolutely necessary

Option 2: Lift and Shift Only

- Migrate to EC2 without containerization
- Pros: Fastest migration (3 months)
- Cons: Limited benefits, still managing VMs
- Recommendation: Only if time-constrained

Option 3: Serverless-First

- Convert to Lambda + API Gateway + DynamoDB
- Pros: Maximum scalability, lowest operational overhead
- Cons: Requires significant rewrite, cold starts
- Recommendation: For new features, not existing monolith

Common Pitfalls to Avoid:

1. **Big Bang Migration**: Incremental migration reduces risk
2. **Ignoring Data Migration Complexity**: DMS testing is critical
3. **Not Modernizing Architecture**: Lift-and-shift alone provides limited benefits
4. **Underestimating Team Training**: Budget time for learning

5. **No Rollback Plan**: Always have a way to revert
6. **Skipping Load Testing**: Test at 2x expected peak load
7. **Not Involving Business Stakeholders**: Get buy-in early
8. **Ignoring Observability**: Implement monitoring from day one

—

Scenario 11: Big Data Analytics Platform

Situation: A retail company collects massive amounts of data from online transactions, mobile app usage, IoT sensors in stores, and social media. They want to build a comprehensive analytics platform to gain real-time insights into customer behavior, optimize inventory, and improve marketing effectiveness. **Current State**:

- Multiple data sources generating 5 TB/day
- Data scattered across different systems
- Manual reporting (takes 2-3 days)
- No real-time analytics
- Limited data science capabilities
- Expensive third-party analytics tools ($500K/year)

Data Sources:

- Web/mobile clickstream: 2 billion events/day
- Transaction logs: 10 million transactions/day
- IoT sensors (foot traffic, temperature): 50 million readings/day
- Social media mentions: APIs and web scraping
- CRM data: Customer profiles and interactions
- Inventory systems: Stock levels and shipments

Requirements: **Functional Requirements**:

- Ingest data from multiple sources
- Real-time dashboards for operations
- Batch processing for daily/weekly reports
- Ad-hoc SQL queries for analysts
- Machine learning for recommendations
- Data retention: Hot (90 days), Warm (1 year), Cold (7 years)

Non-Functional Requirements:

- Real-time latency: ¡1 minute
- Query performance: ¡5 seconds for interactive queries
- Scalability: Handle 10x data growth

- Cost-effective at scale
- Data governance and security
- Self-service analytics for business users

Question: How should they architect a big data analytics platform? **Recommended Architecture**:

1. Data Ingestion Layer Real-Time Streaming Data: **Amazon Kinesis Data Streams**:

- For clickstream, IoT sensors
- Shards: 50 (1 MB/s per shard = 50 MB/s total)
- Retention: 7 days for replay capability
- Producers: Web/mobile apps, IoT devices via Kinesis Agent

Amazon Kinesis Data Firehose:

- Deliver streams to S3, Redshift, OpenSearch
- Automatic batching and compression
- Transform data with Lambda
- Buffer size: 5 MB or 60 seconds

Batch Data Ingestion: **AWS Glue ETL Jobs**:

- Extract from source databases
- Transform and clean data
- Load to S3 data lake
- Schedule: Nightly for transaction logs, CRM data

AWS Database Migration Service (DMS):

- Continuous replication from transactional databases
- Change Data Capture (CDC)
- Minimal impact on source systems

API-Based Ingestion: **AWS Lambda**:

- Fetch data from social media APIs
- Parse and normalize
- Write to Kinesis or S3
- Schedule with EventBridge (hourly)

2. Storage Layer - Data Lake Amazon S3:

- Central data lake repository
- Organized by:
- Data source
- Date partitioning (year/month/day)
- File format (Parquet, ORC for analytics)

S3 Bucket Structure:

```
s3://retail-datalake-raw/
   clickstream/year=2025/month=01/day=15/
   transactions/year=2025/month=01/day=15/
   iot-sensors/year=2025/month=01/day=15/
   social-media/year=2025/month=01/day=15/

s3://retail-datalake-processed/
   customer-360/
   sales-analytics/
   inventory-metrics/

s3://retail-datalake-curated/
   marketing-reports/
   executive-dashboards/
```

S3 Storage Classes:

- Standard: Last 90 days (hot data)
- Standard-IA: 91 days - 1 year (warm data)
- Glacier Flexible Retrieval: 1-7 years (cold data)
- Lifecycle policies for automatic transitions

S3 Features:

- Versioning enabled for data protection
- Server-side encryption (SSE-S3 or SSE-KMS)
- S3 Object Lock for compliance
- S3 Access Points for different teams
- S3 Inventory for data catalog

3. Data Processing Layer Real-Time Processing: Amazon Kinesis Data Analytics:

- SQL queries on streaming data
- Tumbling/sliding windows

- Real-time aggregations
- Anomaly detection
- Output to Lambda, Kinesis, S3

AWS Lambda:

- Process individual events
- Enrich with reference data (DynamoDB)
- Real-time alerts via SNS
- Trigger downstream workflows

Batch Processing: **AWS Glue**:

- Serverless Spark-based ETL
- Discovers schema automatically
- Glue Data Catalog (metadata repository)
- Glue Studio for visual ETL
- Glue DataBrew for data preparation

Amazon EMR (Elastic MapReduce):

- For complex Spark, Hadoop jobs
- EMR on EKS for containerized workloads
- Spot Instances for cost savings (70% reduction)
- Cluster configuration:
- Master: m5.xlarge (1 instance)
- Core: r5.2xlarge (5 instances, On-Demand)
- Task: r5.2xlarge (20 instances, Spot)

Typical Glue ETL Job:

```
import sys
from awsglue.transforms import *
from awsglue.utils import getResolvedOptions
from pyspark.context import SparkContext
from awsglue.context import GlueContext
from awsglue.job import Job

args = getResolvedOptions(sys.argv, ['JOB\_NAME'])
sc = SparkContext()
glueContext = GlueContext(sc)
spark = glueContext.spark\_session
job = Job(glueContext)
job.init(args['JOB\_NAME'], args)

\# Read from Data Catalog
```

```
datasource0 = glueContext.create\_dynamic\_frame.from\
   _catalog(
    database = "retail\_raw",
    table\_name = "clickstream"
)

\# Transform
applymapping1 = ApplyMapping.apply(
    frame = datasource0,
    mappings = [
        ("user\_id", "string", "customer\_id", "string"),
        ("event\_timestamp", "long", "event\_time", "
           timestamp"),
        ("page\_url", "string", "page\_url", "string"),
        ("session\_id", "string", "session\_id", "string")
    ]
)

\# Filter out invalid records
filtered = Filter.apply(
    frame = applymapping1,
    f = lambda x: x["customer\_id"] is not None
)

\# Write to S3 in Parquet format
glueContext.write\_dynamic\_frame.from\_options(
    frame = filtered,
    connection\_type = "s3",
    connection\_options = \{
        "path": "s3://retail-datalake-processed/customer-
           sessions/",
        "partitionKeys": ["year", "month", "day"]
    \},
    format = "parquet"
)

job.commit()
```

4. Data Catalog and Governance AWS Glue Data Catalog:

- Centralized metadata repository
- Schema registry
- Integration with Athena, Redshift, EMR
- Glue Crawlers for automatic schema discovery

AWS Lake Formation:

- Fine-grained access control
- Column-level security

- Row-level security
- Data filtering
- Audit logging
- Governed tables for ACID transactions

Access Control Example:

```
Data Lake Administrator:
  - Full access to all tables

Marketing Team:
  - Read access to: customer\_360, campaign\_analytics
  - Column filtering: Hide PII (SSN, credit card)

Data Science Team:
  - Read access to: all tables
  - Write access to: ml\_models bucket

Finance Team:
  - Read access to: sales\_analytics, inventory\_metrics
  - Row filtering: Only their region's data
```

5. Analytics and Querying Amazon Athena:

- Interactive SQL queries on S3 data
- Serverless (no infrastructure)
- Pay per query ($5 per TB scanned)
- Integration with QuickSight
- Workgroups for cost control
- Query result caching

Optimization Techniques:

- Partition data by date
- Use columnar formats (Parquet, ORC)
- Compress data (Snappy, ZSTD)
- Limit columns in SELECT
- Use approximate functions (approx˙distinct vs COUNT DISTINCT)

Query Performance Comparison:

- CSV, uncompressed: $5/TB, 45 seconds
- Parquet, Snappy: $0.50/TB, 5 seconds
- **90% cost reduction, 9x faster**

Amazon Redshift:

- Data warehouse for complex queries
- Massively parallel processing
- Configuration:
- Node type: ra3.4xlarge
- Nodes: 5 (640 GB RAM, 128 TB storage)
- Redshift Spectrum for S3 queries
- Concurrency Scaling for peak loads
- Materialized views for aggregations

Use Cases:

- Athena: Ad-hoc queries, exploration, infrequent queries
- Redshift: Regular reports, complex joins, consistent performance

6. Business Intelligence and Visualization Amazon QuickSight:

- Serverless BI service
- Connect to Athena, Redshift, S3
- SPICE (in-memory engine) for fast visuals
- ML-powered insights
- Embedded analytics for applications
- Pricing: $5/author/month, $0.30/reader/session

Dashboards:

1. **Executive Dashboard**:

- Daily sales trends
- Revenue by region
- Top products
- Customer acquisition cost

1. **Operations Dashboard**:

- Real-time store foot traffic
- Inventory levels
- Stockout alerts
- Supply chain metrics

1. **Marketing Dashboard**:

- Campaign performance
- Customer segmentation

- Conversion funnels
- Social media sentiment

1. **Data Science Dashboard**:

- Model performance metrics
- A/B test results
- Recommendation effectiveness

7. Machine Learning Pipeline Amazon SageMaker:

- Train recommendation models
- Fraud detection
- Demand forecasting
- Customer churn prediction

ML Workflow:

1. **Data Preparation**: Glue DataBrew or SageMaker Data Wrangler
2. **Feature Engineering**: SageMaker Processing Jobs
3. **Model Training**: SageMaker Training Jobs (Spot Instances)
4. **Model Evaluation**: SageMaker Experiments
5. **Model Registry**: SageMaker Model Registry
6. **Deployment**: SageMaker Endpoints (real-time or batch)
7. **Monitoring**: SageMaker Model Monitor

Amazon Personalize:

- Pre-built recommendation engine
- No ML expertise required
- Real-time and batch recommendations
- Use cases:
- Product recommendations
- Personalized rankings
- Similar items

8. Orchestration and Workflow AWS Step Functions:

- Coordinate multi-step data pipelines
- Visual workflow designer
- Error handling and retry logic
- Integration with Lambda, Glue, EMR, SageMaker

Example Daily Pipeline:

```
1. Ingest data (Lambda, Glue)
   ↓
2. Data quality checks (Lambda)
   ↓
3. ETL processing (Glue or EMR)
   ↓
4. Load to Redshift (Glue)
   ↓
5. Refresh materialized views (Redshift)
   ↓
6. Update ML models (SageMaker)
   ↓
7. Refresh QuickSight datasets
   ↓
8. Send completion notification (SNS)
```

Amazon Managed Workflows for Apache Airflow (MWAA):

- Alternative to Step Functions
- For complex DAGs (Directed Acyclic Graphs)
- Python-based workflow definitions
- Better for data engineering teams familiar with Airflow

9. Monitoring and Optimization Amazon CloudWatch:

- Glue job metrics
- EMR cluster utilization
- Kinesis stream metrics
- Athena query performance
- Custom business metrics

AWS Cost Explorer:

- Analyze spending by service
- Identify optimization opportunities
- Reserved Instance recommendations

AWS Trusted Advisor:

- Cost optimization checks
- Security best practices

Step-by-Step Implementation: **Phase 1: Foundation (Months 1-2)**

1. Set up AWS accounts and networking

2. Create S3 data lake structure
3. Deploy AWS Glue Data Catalog
4. Set up Lake Formation permissions
5. Implement data governance policies

Phase 2: Ingestion (Month 3)

1. Deploy Kinesis streams for real-time data
2. Set up Glue ETL jobs for batch data
3. Implement Lambda for API ingestion
4. Test data flow end-to-end
5. Monitor data quality

Phase 3: Processing (Months 4-5)

1. Build Glue ETL pipelines
2. Deploy EMR clusters for complex processing
3. Implement data quality checks
4. Set up Step Functions orchestration
5. Optimize job performance

Phase 4: Analytics (Month 6)

1. Create Athena tables
2. Deploy Redshift cluster
3. Build initial dashboards in QuickSight
4. Train business users on self-service
5. Gather feedback and iterate

Phase 5: ML (Months 7-8)

1. Set up SageMaker environment
2. Build recommendation model
3. Deploy fraud detection
4. Implement demand forecasting
5. Monitor model performance

Phase 6: Optimization (Ongoing)

1. Right-size resources
2. Implement caching strategies
3. Optimize data formats
4. Use Spot Instances
5. Continuous cost monitoring

Cost Breakdown (Monthly):

Service & **Configuration** & **Monthly Cost**
Kinesis Data Streams & 50 shards, 5TB ingestion & $1,200
Kinesis Firehose & 5TB delivery & $125
S3 Storage & 100TB (tiered) & $2,000
AWS Glue & 200 DPU-hours ETL & $880
Amazon EMR & 25 nodes, 8 hrs/day, 70% Spot & $2,400
Redshift & 5 ra3.4xlarge nodes & $12,000
Athena & 10TB scanned/month & $50
QuickSight & 50 authors, 500 readers & $400
SageMaker & Training + endpoints & $1,500
Lambda & 50M invocations & $100
Data Transfer & Outbound & $500
CloudWatch & Logs and metrics & $300
Total & **˜$21,455/month** &

Cost Optimization Strategies:

1. Use Spot Instances for EMR (70% savings)
2. Convert data to Parquet (90% storage reduction)
3. Partition data effectively (80% query cost reduction)
4. Use S3 Intelligent-Tiering
5. Right-size Redshift with pause/resume
6. Use Athena for infrequent queries vs. Redshift
7. Implement S3 lifecycle policies

Optimized Cost: ˜$14,000/month (35% reduction) **Benefits**:

- **Real-Time Insights**: ¡1 minute latency for operational decisions
- **Cost Savings**: $500K/year (third-party tools) → $168K/year (50% savings)
- **Scalability**: Handle 10x data growth without architectural changes
- **Self-Service**: Business users run their own queries
- **Data-Driven Decisions**: ML-powered recommendations increase revenue 15%
- **Time to Insight**: 2-3 days → ¡1 hour for reports
- **Compliance**: Fine-grained access control and audit trails

Trade-offs:

- **Complexity**: Distributed systems require skilled team
- **Learning Curve**: Training required for Spark, SQL, ML
- **Initial Cost**: Higher upfront investment
- **Data Quality**: Garbage in, garbage out - need robust quality checks

Alternative Approaches: **Option 1: Redshift-Centric**

- Load all data into Redshift
- Simpler architecture
- Higher cost for storage
- Best for: Smaller datasets (¡10TB)

Option 2: EMR-Centric

- Use EMR for all processing
- More control and flexibility
- More operational overhead
- Best for: Teams with Hadoop/Spark expertise

Option 3: Third-Party (Snowflake, Databricks)

- Managed services
- Excellent performance
- Higher cost
- Less control
- Best for: Teams wanting minimal operational burden

Common Pitfalls to Avoid:

1. **Not Partitioning Data**: Results in slow queries and high costs
2. **Ignoring Data Formats**: CSV vs. Parquet makes 10x difference
3. **Over-Provisioning**: Start small, scale as needed
4. **No Data Governance**: Implement access controls from day one
5. **Ignoring Data Quality**: Build validation into pipelines
6. **Not Using Spot Instances**: 70% cost savings for EMR
7. **Storing Everything in Redshift**: Use S3 data lake + Redshift Spectrum
8. **No Monitoring**: Implement CloudWatch alarms and dashboards early

—

Scenario 12: DevOps CI/CD Pipeline Implementation

Situation: A SaaS company with 20 developers is struggling with manual deployment processes. Code is deployed to production once a month, with frequent rollbacks due to bugs. Deployments take 4-6 hours and require manual steps. The team wants to implement modern DevOps practices with automated CI/CD pipelines. **Current State**:

- Manual deployments via SSH and scripts
- No automated testing

- Deployment frequency: Monthly
- Deployment duration: 4-6 hours
- Rollback rate: 30%
- Production incidents: 2-3 per month
- Developer frustration: High
- Time to market: 4-6 weeks for features

Current Process:

1. Developers commit to shared Git branch
2. Manual code review (informal)
3. QA team tests for 1 week
4. Operations team deploys on weekends
5. Frequent production issues on Monday

Problems:

- Long feedback loops
- Manual error-prone deployments
- No deployment consistency
- Difficult rollbacks
- Fear of deploying
- Bottleneck at operations team

Requirements: **Functional Requirements**:

- Automated build and test on every commit
- Automated deployment to dev/staging/prod
- Code quality checks (linting, security scanning)
- Automated rollback capability
- Infrastructure as Code
- Secrets management
- Multi-environment support

Non-Functional Requirements:

- Deployment frequency: Multiple times per day
- Deployment duration: ¡15 minutes
- Automated rollback: ¡5 minutes
- Rollback rate: ¡5%
- Zero-downtime deployments
- Audit trail for compliance
- Cost-effective

Question: How should they implement a modern CI/CD pipeline? **Recommended CI/CD Architecture**:

1. Source Control and Branching Strategy AWS CodeCommit:

- Git-based source control
- Integration with AWS services
- Encryption at rest and in transit
- IAM-based access control
- Supports Git LFS for large files
- Pull request workflows

Alternative: GitHub, GitLab, Bitbucket

- If already using these platforms
- CodePipeline integrates with all

Branching Strategy (Trunk-Based Development):

```
main (production)
   feature/user-auth (short-lived)
   feature/payment-integration (short-lived)
   hotfix/critical-bug (short-lived)
```

Trunk-Based Guidelines:

- Small, frequent commits to main
- Feature flags for incomplete features
- Short-lived feature branches (¡2 days)
- Pull requests with automated checks
- Merge only if tests pass

2. Continuous Integration Pipeline AWS CodeBuild:

- Fully managed build service
- Docker-based build environments
- Pay per build minute
- Scales automatically
- Integration with security scanning tools

Build Specification (buildspec.yml):

```
version: 0.2

phases:
  install:
    runtime-versions:
      nodejs: 18
      docker: 20
    commands:
```

```
      - echo Installing dependencies...
      - npm install

  pre\_build:
    commands:
      - echo Running pre-build checks...
      - npm run lint
      - npm run security-check
      - echo Logging in to Amazon ECR...
      - aws ecr get-login-password --region \$AWS\_DEFAULT\
        _REGION | docker login --username AWS --password-
        stdin \$AWS\_ACCOUNT\_ID.dkr.ecr.\$AWS\_DEFAULT\
        _REGION.amazonaws.com

  build:
    commands:
      - echo Build started on ‘date‘
      - echo Running unit tests...
      - npm test -- --coverage
      - echo Building application...
      - npm run build
      - echo Building Docker image...
      - docker build -t \$IMAGE\_REPO\_NAME:\$CODEBUILD\
        _RESOLVED\_SOURCE\_VERSION .
      - docker tag \$IMAGE\_REPO\_NAME:\$CODEBUILD\_RESOLVED\
        _SOURCE\_VERSION \$AWS\_ACCOUNT\_ID.dkr.ecr.\$AWS\
        _DEFAULT\_REGION.amazonaws.com/\$IMAGE\_REPO\_NAME:\
        $CODEBUILD\_RESOLVED\_SOURCE\_VERSION
      - docker tag \$IMAGE\_REPO\_NAME:\$CODEBUILD\_RESOLVED\
        _SOURCE\_VERSION \$AWS\_ACCOUNT\_ID.dkr.ecr.\$AWS\
        _DEFAULT\_REGION.amazonaws.com/\$IMAGE\_REPO\_NAME:
        latest

  post\_build:
    commands:
      - echo Build completed on ‘date‘
      - echo Pushing Docker image...
      - docker push \$AWS\_ACCOUNT\_ID.dkr.ecr.\$AWS\_DEFAULT
        \_REGION.amazonaws.com/\$IMAGE\_REPO\_NAME:\
        $CODEBUILD\_RESOLVED\_SOURCE\_VERSION
      - docker push \$AWS\_ACCOUNT\_ID.dkr.ecr.\$AWS\_DEFAULT
        \_REGION.amazonaws.com/\$IMAGE\_REPO\_NAME:latest
      - echo Generating build artifacts...
      - printf '[\{"name":"app-container","imageUri":"\%s
        "\}]' \$AWS\_ACCOUNT\_ID.dkr.ecr.\$AWS\_DEFAULT\
        _REGION.amazonaws.com/\$IMAGE\_REPO\_NAME:\
        $CODEBUILD\_RESOLVED\_SOURCE\_VERSION >
        imagedefinitions.json

artifacts:
  files:
```

```
    - imagedefinitions.json
    - appspec.yml
    - taskdef.json
    - '**/*'
  discard-paths: no

reports:
  test-results:
    files:
      - 'test-results/**/*'
    file-format: 'JUNITXML'
  coverage-report:
    files:
      - 'coverage/clover.xml'
    file-format: 'CLOVERXML'

cache:
  paths:
    - '/root/.npm/**/*'
    - 'node\_modules/**/*'
```

Automated Checks in CI:

1. **Unit Tests**: Jest, Mocha, pytest
2. **Code Coverage**: Minimum 80% threshold
3. **Linting**: ESLint, Prettier, Black
4. **Security Scanning**:

- Snyk for dependency vulnerabilities
- OWASP Dependency-Check
- SonarQube for code quality

5. **Container Scanning**: Amazon ECR image scanning
6. **Infrastructure Validation**: cfn-lint, terraform validate

3. Continuous Deployment Pipeline AWS CodePipeline:

- Orchestrates CI/CD workflow
- Visual pipeline editor
- Integration with third-party tools
- Parallel and sequential stages
- Approval gates
- Automated rollback

Pipeline Stages:

```
                    SOURCE STAGE
- CodeCommit/GitHub trigger on push to main
- Fetch source code

                    BUILD STAGE
- CodeBuild compiles, tests, builds Docker image
- Push to ECR
- Generate artifacts

                  TEST STAGE (Dev)
- Deploy to Dev environment (ECS/EKS)
- CodeBuild: Integration tests
- CodeBuild: API tests (Postman/Newman)
- CodeBuild: Performance tests (k6, JMeter)

                DEPLOY STAGE (Staging)
- CodeDeploy to Staging environment
- Blue/green deployment
- Smoke tests

                 MANUAL APPROVAL
- SNS notification to approvers
- Review test results
- Approve or reject production deployment

              DEPLOY STAGE (Production)
- CodeDeploy to Production
- Blue/green deployment
- Traffic shifting: 10\% → 50\% → 100\%
- Automatic rollback on CloudWatch alarms
```

4. Deployment Strategy AWS CodeDeploy:

- Automated deployments
- Multiple deployment types:
- In-place
- Blue/green
- Canary
- Linear
- Automatic rollback
- Integration with ECS, Lambda, EC2, on-premises

Blue/Green Deployment (ECS): **AppSpec File (appspec.yml)**:

```
version: 0.0
Resources:
  - TargetService:
      Type: AWS::ECS::Service
      Properties:
        TaskDefinition: <TASK\_DEFINITION>
        LoadBalancerInfo:
          ContainerName: "app-container"
          ContainerPort: 8080
        PlatformVersion: "LATEST"
        NetworkConfiguration:
          AwsvpcConfiguration:
            Subnets:
              - subnet-12345678
              - subnet-87654321
            SecurityGroups:
              - sg-12345678
            AssignPublicIp: "DISABLED"

Hooks:
  - BeforeInstall: "LambdaFunctionToValidateBeforeInstall"
  - AfterInstall: "LambdaFunctionToValidateAfterInstall"
  - AfterAllowTestTraffic: "
     LambdaFunctionToRunIntegrationTests"
  - BeforeAllowTraffic: "LambdaFunctionToWarmUpCache"
  - AfterAllowTraffic: "LambdaFunctionToValidateProduction"
```

Traffic Shifting Strategy:

- **Canary**: 10% of traffic for 5 minutes, then 100%
- **Linear**: Increase by 10% every 5 minutes
- **All-at-once**: Immediate switch (not recommended for prod)

Automatic Rollback Triggers:

- CloudWatch Alarm: Error rate ¿5%
- CloudWatch Alarm: Response time ¿2 seconds
- CloudWatch Alarm: CPU utilization ¿80%
- Deployment failure

5. Infrastructure as Code AWS CloudFormation:

- Define infrastructure in YAML/JSON
- Version control infrastructure
- Stack updates with rollback
- Drift detection

Alternative: AWS CDK (Cloud Development Kit):

- Define infrastructure in programming languages
- Python, TypeScript, Java, C#
- Higher level abstractions
- Synthesizes to CloudFormation

Example CDK Stack (TypeScript):

```
import * as cdk from 'aws-cdk-lib';
import * as ec2 from 'aws-cdk-lib/aws-ec2';
import * as ecs from 'aws-cdk-lib/aws-ecs';
import * as ecsPatterns from 'aws-cdk-lib/aws-ecs-patterns';
import * as codedeploy from 'aws-cdk-lib/aws-codedeploy';

export class AppInfraStack extends cdk.Stack \{
  constructor(scope: cdk.App, id: string, props?: cdk.
     StackProps) \{
    super(scope, id, props);

    // VPC
    const vpc = new ec2.Vpc(this, 'AppVPC', \{
      maxAzs: 3,
      natGateways: 2
    \});

    // ECS Cluster
    const cluster = new ecs.Cluster(this, 'AppCluster', \{
      vpc: vpc,
      containerInsights: true
    \});

    // Fargate Service with ALB
    const fargateService = new ecsPatterns.
       ApplicationLoadBalancedFargateService(
      this,
      'AppService',
      \{
        cluster: cluster,
        cpu: 512,
        desiredCount: 3,
        taskImageOptions: \{
          image: ecs.ContainerImage.fromRegistry('amazon/
             amazon-ecs-sample'),
```

```
        containerPort: 8080,
        environment: \{
          ENVIRONMENT: 'production'
        \}
      \},
      memoryLimitMiB: 1024,
      publicLoadBalancer: true,
      deploymentController: \{
        type: ecs.DeploymentControllerType.CODE\_DEPLOY
      \}
    \}
  );

  // Auto Scaling
  const scaling = fargateService.service.autoScaleTaskCount
     (\{
    minCapacity: 3,
    maxCapacity: 20
  \});

  scaling.scaleOnCpuUtilization('CpuScaling', \{
    targetUtilizationPercent: 70
  \});

  scaling.scaleOnMemoryUtilization('MemoryScaling', \{
    targetUtilizationPercent: 80
  \});

  // CodeDeploy Deployment Group
  const deploymentGroup = new codedeploy.EcsDeploymentGroup
     (
    this,
    'AppDeploymentGroup',
    \{
      service: fargateService.service,
      blueGreenDeploymentConfig: \{
        blueTargetGroup: fargateService.targetGroup,
        greenTargetGroup: fargateService.targetGroup,
        listener: fargateService.listener,
        terminationWaitTime: cdk.Duration.minutes(5)
      \},
      deploymentConfig: codedeploy.EcsDeploymentConfig.
         CANARY\_10PERCENT\_5MINUTES,
      autoRollback: \{
        failedDeployment: true,
        stoppedDeployment: true,
        deploymentInAlarm: true
      \},
      alarms: [
        new cloudwatch.Alarm(this, 'ErrorAlarm', \{
          metric: fargateService.targetGroup.metrics.
```

```
                httpCodeTarget(
                elb.HttpCodeTarget.TARGET\_5XX\_COUNT
              ),
              threshold: 10,
              evaluationPeriods: 2
            \})
        ]
      \}
    );
  \}
\}
```

6. Secrets Management AWS Secrets Manager:

- Store database credentials, API keys
- Automatic rotation
- Encryption with KMS
- Fine-grained access control
- Integration with RDS, Redshift

Alternative: AWS Systems Manager Parameter Store:

- Free for standard parameters
- Hierarchical storage
- No automatic rotation
- Good for configuration values

Example Usage in ECS Task:

```
\{
  "containerDefinitions": [
    \{
      "name": "app-container",
      "secrets": [
        \{
          "name": "DB\_PASSWORD",
          "valueFrom": "arn:aws:secretsmanager:us-east
             -1:123456789012:secret:prod/db/password-AbCdEf"
        \},
        \{
          "name": "API\_KEY",
          "valueFrom": "arn:aws:secretsmanager:us-east
             -1:123456789012:secret:prod/api/key-XyZaBc"
        \}
      ]
    \}
  ]
\}
```

7. Monitoring and Observability Amazon CloudWatch:

- Unified logging from all environments
- Custom metrics
- Dashboards for pipeline health
- Alarms for deployment failures

AWS X-Ray:

- Distributed tracing
- Identify performance bottlenecks
- Request flow visualization

Key Metrics to Monitor:

- **Deployment Frequency**: Deploys per day
- **Lead Time**: Commit to production time
- **Mean Time to Recovery (MTTR)**: Time to fix production issue
- **Change Failure Rate**: % of deployments causing failure
- **Build Duration**: Time for build pipeline
- **Test Coverage**: % of code covered by tests

CloudWatch Dashboard:

```
\{
  "widgets": [
    \{
      "type": "metric",
      "properties": \{
        "metrics": [
          [ "AWS/CodePipeline", "PipelineExecutionSuccess",
             \{ "stat": "Sum" \} ],
          [ ".", "PipelineExecutionFailure", \{ "stat": "Sum"
              \} ]
        ],
        "period": 300,
        "stat": "Sum",
        "region": "us-east-1",
        "title": "Pipeline Executions"
      \}
    \},
    \{
      "type": "metric",
      "properties": \{
        "metrics": [
          [ "AWS/ECS", "CPUUtilization", \{ "stat": "Average"
              \} ],
          [ ".", "MemoryUtilization", \{ "stat": "Average" \}
              ]
```

```
        ],
        "period": 300,
        "stat": "Average",
        "region": "us-east-1",
        "title": "ECS Resource Utilization"
      \}
    \}
  ]
\}
```

8. Multi-Environment Strategy Account Structure:

- Dev Account: Frequent deployments, lower-cost resources
- Staging Account: Production-like environment
- Production Account: Strict change control

Environment-Specific Configuration:

```
config/
   dev.json
   staging.json
   production.json
```

Parameter Store Hierarchy:

```
/app/dev/database/host
/app/dev/database/port
/app/staging/database/host
/app/staging/database/port
/app/production/database/host
/app/production/database/port
```

9. Testing Strategy Test Pyramid:

```
          E2E Tests    ← Fewer, slower, expensive
           (Cypress)


       Integration Tests
         (API, Database)


         Unit Tests         ← Many, fast, cheap
    (Jest, pytest, JUnit)
```

Test Types:

1. **Unit Tests**: 80% of tests, ¡100ms each
2. **Integration Tests**: 15% of tests, ¡5s each
3. **E2E Tests**: 5% of tests, ¡30s each

CodeBuild Test Stage:

```
phases:
  build:
    commands:
      - echo Running unit tests...
      - npm test -- --coverage --maxWorkers=4
      - echo Running integration tests...
      - npm run test:integration
      - echo Running E2E tests...
      - npm run test:e2e

  post\_build:
    commands:
      - echo Checking test coverage threshold...
      - npm run coverage:check -- --lines 80 --functions 80
         --branches 75
```

10. Rollback Strategy Automatic Rollback:

– CloudWatch alarms trigger rollback
– CodeDeploy automatically reverts to previous version
– ¡5 minutes to rollback

Manual Rollback:

```
\# Rollback to previous deployment
aws deploy stop-deployment \textbackslash\{\}
  --deployment-id d-1234567890 \textbackslash\{\}
  --auto-rollback-enabled

\# Or redeploy previous version
aws ecs update-service \textbackslash\{\}
  --cluster my-cluster \textbackslash\{\}
  --service my-service \textbackslash\{\}
  --task-definition my-task:42  \# Previous version
```

Feature Flags:

– Use AWS AppConfig or Launch Darkly
– Toggle features without redeployment
– Gradual rollout to users
– Quick disable if issues arise

Step-by-Step Implementation: **Phase 1: Source Control (Week 1)**

1. Migrate code to CodeCommit/GitHub
2. Define branching strategy
3. Set up pull request workflows
4. Configure branch protection rules
5. Train team on Git best practices

Phase 2: CI Pipeline (Week 2)

1. Create buildspec.yml
2. Set up CodeBuild project
3. Integrate linting and testing
4. Add security scanning
5. Configure build notifications

Phase 3: Containerization (Week 3)

1. Create Dockerfile
2. Set up Amazon ECR
3. Build container images in pipeline
4. Test locally with Docker Compose
5. Document container configuration

Phase 4: Infrastructure as Code (Week 4)

1. Define infrastructure in CloudFormation/CDK
2. Create VPC, subnets, security groups
3. Deploy ECS cluster and services
4. Set up Application Load Balancer
5. Test infrastructure provisioning

Phase 5: CD Pipeline (Week 5)

1. Create CodePipeline
2. Add deployment stages (Dev, Staging, Prod)
3. Configure CodeDeploy
4. Set up approval gates
5. Test end-to-end deployment

Phase 6: Monitoring (Week 6)

1. Set up CloudWatch dashboards
2. Configure alarms
3. Integrate X-Ray tracing
4. Set up log aggregation

5. Define KPIs and metrics

Phase 7: Testing and Optimization (Weeks 7-8)

1. Test failure scenarios
2. Practice rollback procedures
3. Optimize build times
4. Tune auto-scaling parameters
5. Document runbooks

Cost Breakdown (Monthly):

Service & **Configuration** & **Monthly Cost**
CodeCommit & 5 active users, 10 GB & $2
CodeBuild & 500 build minutes (general1.small) & $25
CodePipeline & 10 pipelines, 200 executions & $10
CodeDeploy & Free for ECS, Lambda & $0
ECR & 50 GB storage & $5
ECS Fargate & 6 tasks (0.5 vCPU, 1 GB) & $140
ALB & 2 load balancers & $40
S3 & Artifact storage & $5
CloudWatch & Logs, metrics, dashboards & $50
Secrets Manager & 20 secrets & $8
Total & **˜$285/month** &

Benefits:

- **Deployment Frequency**: Monthly → Multiple times per day
- **Deployment Duration**: 4-6 hours → ¡15 minutes
- **Rollback Time**: Hours → ¡5 minutes
- **Rollback Rate**: 30% → ¡5%
- **Developer Productivity**: +40% (less time on deployments)
- **Mean Time to Recovery**: 4 hours → ¡30 minutes
- **Production Incidents**: 2-3/month → ¡1/month
- **Time to Market**: 4-6 weeks → 1-2 weeks
- **Developer Satisfaction**: Significantly improved

Trade-offs:

- **Initial Setup**: 6-8 weeks for full implementation
- **Learning Curve**: Team must learn new tools and practices
- **Cultural Change**: Shift from manual to automated processes
- **Responsibility Shift**: Developers more involved in operations

Alternative Approaches: **Option 1: Jenkins on EC2**

- Open-source CI/CD
- More plugins and flexibility
- Requires server management
- Higher operational overhead
- Best for: Teams with existing Jenkins expertise

Option 2: GitHub Actions

- Native GitHub integration
- Easy to set up
- Limited AWS integration compared to CodePipeline
- Best for: GitHub-centric workflows

Option 3: GitLab CI/CD

- All-in-one DevOps platform
- Built-in container registry
- Requires separate hosting
- Best for: Teams wanting single platform

Option 4: Third-Party (CircleCI, Travis CI)

- Easy to set up
- Great developer experience
- Additional cost
- Limited control
- Best for: Startups wanting quick setup

Common Pitfalls to Avoid:

1. **No Rollback Testing**: Practice rollbacks regularly
2. **Skipping Integration Tests**: Catch issues before production
3. **Manual Steps in Pipeline**: Automate everything
4. **Ignoring Build Times**: Optimize for ¡10 minute builds
5. **No Monitoring**: Implement comprehensive monitoring early
6. **Over-Engineering**: Start simple, add complexity as needed
7. **Ignoring Security**: Scan for vulnerabilities in pipeline
8. **No Documentation**: Document architecture and runbooks
9. **Forgetting Notifications**: Alert team on pipeline failures
10. **Not Measuring**: Track DORA metrics (deployment frequency, lead time, MTTR, change failure rate)

—

9.0.3 Common Troubleshooting Scenarios

Cannot Connect to EC2 Instance

Symptoms: SSH or RDP connection times out or refused **Troubleshooting Steps**:

1. Verify Instance Status

- Check instance state is "running"
- Check status checks are passing
- View system log for boot errors

2. Check Security Group

- Ensure inbound rule allows SSH (22) or RDP (3389)
- Verify source IP is allowed (0.0.0.0/0 or your IP)
- Check if security group changed recently

3. Check Network ACL

- Ensure NACL allows inbound traffic on port
- Ensure NACL allows ephemeral outbound ports (1024-65535)
- **Important**: NACLs are stateless!

4. Verify Network Configuration

- Instance has public IP (if connecting from internet)
- Instance in public subnet (has IGW route)
- Or using bastion host for private subnet

5. Check Key Pair

- Using correct .pem/.ppk file
- File permissions correct (`chmod 400` for .pem)
- Key pair matches instance

6. Check Route Table

- Subnet has route to IGW (0.0.0.0/0 → igw-xxx)
- Or route to NAT Gateway for private subnet

Key Point

Tip: Use EC2 Instance Connect or Systems Manager Session Manager as alternatives to SSH/RDP when troubleshooting connectivity issues.

—

S3 Access Denied Errors

Common Causes and Solutions:

1. IAM Permissions

- Verify IAM policy grants `s3:GetObject`, `s3:PutObject`
- Check for explicit Deny statements
- Verify resource ARN in policy matches bucket

2. Bucket Policy

- Check bucket policy doesn't deny access
- Verify Principal in policy
- Check for IP-based restrictions

3. Block Public Access

- If public access needed, disable Block Public Access
- Check both bucket-level and account-level settings

4. Encryption

- If using SSE-KMS, verify KMS key policy
- Ensure user has `kms:Decrypt` permission

5. Cross-Account Access

- Bucket policy must allow cross-account access
- Assume role with correct permissions

Key Point

Debugging Tip: Use AWS CloudTrail to review the API call and see the exact reason for access denial. Look for the `errorCode` and `errorMessage` fields in the CloudTrail logs.

—

Lambda Function Issues

Issue 1: Function Timing Out Solutions:

- Increase timeout (default 3 sec, max 15 min)
- Optimize code performance
- Check VPC configuration (can add latency)
- Increase memory (also increases CPU)
- Investigate cold start delays

Key Point

Best Practice: Set the timeout to slightly higher than your expected execution time, but not unnecessarily high to avoid long-running failed executions.

Issue 2: Insufficient Permissions Solutions:

- Check Lambda execution role has required permissions
- Review CloudWatch Logs for permission errors
- Add necessary IAM policies to execution role
- For VPC: Ensure role has VPC execution permissions

Common Required Permissions:

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Allow",
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents"
      ],
      "Resource": "arn:aws:logs:*:*:*"
    \}
  ]
\}
```

Issue 3: Throttling Solutions:

- Request concurrency limit increase
- Implement exponential backoff in calling application
- Use SQS to buffer requests
- Consider reserved concurrency for critical functions

Understanding Throttling:

- **Account-level limit**: 1,000 concurrent executions per region (default)
- **Function-level limit**: Can be set with reserved concurrency
- **Unreserved pool**: Shared across all functions without reserved concurrency

Key Point

Tip: Monitor the `ConcurrentExecutions` and `Throttles` metrics in CloudWatch to identify throttling issues early.

—

RDS Connection Problems

Symptoms: Cannot connect to RDS database from application **Troubleshooting Steps**:

1. Verify Endpoint and Port

- Check endpoint hostname is correct
- Default ports: MySQL (3306), PostgreSQL (5432), SQL Server (1433)
- Verify database is available (not stopped or in maintenance)

2. Security Group Configuration

- Inbound rule must allow traffic on database port
- Source should be application security group or IP range
- Example: MySQL on 3306 from application SG

3. Network Accessibility

- **Public Accessibility**: Set to Yes if connecting from internet
- **Private Subnet**: Application must be in same VPC or have connectivity
- **VPC Peering/VPN**: Required for cross-VPC or on-premises access

4. Database Credentials

- Verify username and password
- Check if password has special characters needing escaping
- Master user vs. database-specific users
- For Aurora, use cluster endpoint for writes, reader endpoint for reads

5. Network ACLs

- Check subnet NACL allows traffic on database port
- Both inbound and outbound rules needed (stateless)

6. SSL/TLS Requirements

- Some databases require SSL connections
- Download RDS certificate bundle
- Configure application to use SSL

7. Connection Limits

- RDS has maximum connections based on instance class
- MySQL: `DBInstanceClassMemory/12582880`
- Check CloudWatch `DatabaseConnections` metric
- If maxed out, scale up instance or optimize connection pooling

Testing Connection:

```
\# From EC2 instance in same VPC
\# MySQL
mysql -h mydb.abc123.us-east-1.rds.amazonaws.com -P 3306 -u
   admin -p

\# PostgreSQL
psql -h mydb.abc123.us-east-1.rds.amazonaws.com -p 5432 -U
   admin -d mydb

\# Test connectivity
telnet mydb.abc123.us-east-1.rds.amazonaws.com 3306
```

Common Solutions:

- Add application security group to RDS security group inbound rules
- Enable public accessibility (for testing only, not production)
- Check VPC routing and internet gateway configuration
- Verify database is in same VPC as application
- Use AWS Systems Manager Session Manager to connect to EC2, then test RDS connection

—

CloudFormation Stack Failures

Symptoms: CloudFormation stack creation or update fails, rolls back **Common Failure Reasons**:

1. Insufficient IAM Permissions **Error**: `User is not authorized to perform: [action]` **Solutions**:

- User/role needs permissions for all resources being created
- CloudFormation also needs permissions via service role
- Add required permissions to IAM policy
- Use CloudFormation service role with necessary permissions

Example IAM Policy:

```
\{
  "Version": "2012-10-17",
  "Statement": [
    \{
      "Effect": "Allow",
      "Action": [
        "cloudformation:*",
        "ec2:*",
        "iam:*",
        "s3:*"
      ],
      "Resource": "*"
    \}
  ]
\}
```

2. Resource Limits Exceeded **Error**: `LimitExceeded` or `ResourceLimitExceeded` **Solutions**:

- Check service quotas (VPCs, Elastic IPs, EC2 instances)
- Request limit increase via Service Quotas console
- Use existing resources instead of creating new ones
- Deploy to different region with available capacity

3. Parameter Validation Errors **Error**: `Parameters: [parameter] must match pattern [regex]` **Solutions**:

- Verify parameter values match constraints
- Check AllowedValues, MinLength, MaxLength
- Ensure CIDR blocks don't overlap
- Validate AMI IDs exist in target region

4. Resource Already Exists **Error**: `Resource already exists` **Solutions**:

- Delete existing resource or import it
- Use different resource names/logical IDs
- Check for resources from previous failed stacks
- Use DeletionPolicy: Retain to keep resources on stack deletion

5. Circular Dependencies **Error**: `Circular dependency between resources` **Solutions**:

- Review DependsOn attributes
- Remove unnecessary dependencies
- Restructure template to break circular references
- Use nested stacks to separate dependent resources

6. Insufficient Capacity **Error**: `Insufficient capacity` for EC2 instances **Solutions**:

- Try different availability zone
- Use different instance type
- Try multiple instance types with launch templates
- Deploy across multiple AZs

7. Timeout Issues **Error**: Resource creation timed out **Solutions**:

- Increase CreationPolicy timeout
- Check resource is actually being created (CloudWatch Logs)
- For ASG, verify instances can reach metadata service
- Use cfn-signal from instance user data

Troubleshooting Tools: **CloudFormation Events**:

- View stack events for detailed error messages
- Identify which resource failed
- Check status reason for failure cause

Change Sets:

- Preview changes before executing
- Identify resources that will be replaced
- Validate template before stack update

Stack Drift Detection:

- Detect if resources were manually modified

- Compare actual configuration vs. template
- Resolve drift before updating stack

Template Validation:

```
\# Validate template syntax
aws cloudformation validate-template --template-body file://
   template.yaml

\# Use cfn-lint for advanced validation
pip install cfn-lint
cfn-lint template.yaml
```

Best Practices:

1. Always validate templates before deployment
2. Use change sets for stack updates
3. Implement rollback triggers with CloudWatch alarms
4. Set appropriate timeouts for resource creation
5. Use DeletionPolicy: Retain for critical resources
6. Test templates in dev environment first
7. Use nested stacks for complex infrastructure
8. Enable termination protection for production stacks

—

Auto Scaling Not Working

Symptoms: Auto Scaling Group not launching or terminating instances as expected **Troubleshooting Steps**:

1. Verify Scaling Policies Check Policy Configuration:

- Target tracking vs. step scaling vs. simple scaling
- Metric being monitored (CPU, memory, custom)
- Target value or step adjustments
- Cooldown periods preventing rapid scaling

Example Issue:

- Target: 70% CPU utilization
- Current: 85% CPU
- But no scale-out occurring

Solutions:

- Check if in cooldown period (default 300 seconds)

- Verify CloudWatch alarm state is ALARM
- Check alarm has datapoints exceeding threshold
- Ensure policy is enabled

2. Check Auto Scaling Group Configuration Capacity Limits:

- Minimum capacity: Can't scale below this
- Maximum capacity: Can't scale above this
- Desired capacity: Current target

Common Issue: Max capacity reached

```
Current: 10 instances
Max capacity: 10
Result: Cannot scale out, even if CPU is high
```

Solutions:

- Increase max capacity
- Review if capacity limits are appropriate
- Check service quotas for EC2 instances

3. Launch Template/Configuration Issues Invalid AMI:

- AMI deleted or not available in region
- AMI shared from another account no longer accessible

Insufficient IAM Permissions:

- Instance profile missing required permissions
- Cannot access S3, Parameter Store, Secrets Manager

Invalid User Data:

- Syntax errors in user data script
- Script fails causing instance initialization to fail

Solutions:

- Check AMI exists: `aws ec2 describe-images --image-ids ami-xxx`
- Review CloudWatch Logs for user data script output
- Test launch template manually by launching instance
- Verify security groups and key pairs are valid

4. Availability Zone Issues No Capacity:

- EC2 capacity not available in specified AZs
- Only some AZs have capacity

Solutions:

- Distribute across multiple AZs
- Use multiple instance types (mixed instances policy)
- Enable capacity rebalancing

5. Health Check Failures Instances Terminating Immediately:

- Health check type: EC2 vs. ELB
- Health check grace period too short
- Instances failing health checks

Symptoms:

- Instances launch, then terminate repeatedly
- CloudWatch shows instances unhealthy

Solutions:

- Increase health check grace period (300-600 seconds)
- Fix application issues causing health check failures
- Verify ELB target group health check settings
- Check security groups allow health check traffic

6. Service Quotas EC2 Instance Limits:

- On-Demand vCPU limits
- Spot Instance limits
- Per-region limits

Check Current Usage:

```
\# Check service quotas
aws service-quotas get-service-quota \textbackslash\{\}
  --service-code ec2 \textbackslash\{\}
  --quota-code L-1216C47A  \# Running On-Demand Standard
     instances
```

Solutions:

- Request quota increase
- Use different instance types
- Deploy to different region

7. Scaling Suspended Check Suspended Processes:

- ReplaceUnhealthy
- Launch
- Terminate
- AddToLoadBalancer

Resume Processes:

```
aws autoscaling resume-processes \textbackslash\{\}
  --auto-scaling-group-name my-asg
```

8. CloudWatch Alarm Issues Alarm Not Triggering:

- Insufficient data
- Metric not published
- Threshold not exceeded for required evaluation periods
- Alarm in INSUFFICIENT˙DATA state

Solutions:

- Check CloudWatch metrics are being published
- Verify alarm configuration (threshold, periods)
- Review alarm history
- Test with lower threshold temporarily

Debugging Commands:

```
\# Describe Auto Scaling Group
aws autoscaling describe-auto-scaling-groups \textbackslash
   \{\}
  --auto-scaling-group-names my-asg

\# View scaling activities
aws autoscaling describe-scaling-activities \textbackslash
   \{\}
  --auto-scaling-group-name my-asg \textbackslash\{\}
  --max-records 20

\# Check scaling policies
aws autoscaling describe-policies \textbackslash\{\}
  --auto-scaling-group-name my-asg

\# View CloudWatch alarms
aws cloudwatch describe-alarms \textbackslash\{\}
  --alarm-names my-cpu-alarm
```

Common Solutions:

1. Ensure min/max/desired capacity are appropriate
2. Verify CloudWatch alarms are in ALARM state
3. Check for suspended processes
4. Increase health check grace period
5. Fix launch template issues (AMI, security groups, user data)
6. Distribute across multiple AZs for availability
7. Use multiple instance types to improve capacity availability
8. Monitor with CloudWatch and set up alerting

—

High AWS Bill Unexpectedly

Symptoms: AWS bill significantly higher than expected or usual **Common Cost Culprits**:

1. Untagged or Orphaned Resources EC2 Instances Running:

- Instances left running after testing
- Auto Scaling not scaling down
- Spot requests creating instances

Check:

```
\# List all running instances
aws ec2 describe-instances \textbackslash\{\}
  --filters "Name=instance-state-name,Values=running" \
     textbackslash\{\}
  --query 'Reservations[].Instances[].[InstanceId,
     InstanceType,Tags[?Key==`Name`].Value|[0]]'
```

EBS Volumes:

- Volumes detached from terminated instances
- Snapshots accumulating over time
- Volumes larger than needed

Check:

```
\# Find unattached volumes
aws ec2 describe-volumes \textbackslash\{\}
  --filters "Name=status,Values=available" \textbackslash\{\}
  --query 'Volumes[].[VolumeId,Size,CreateTime]'
```

Solutions:

- Terminate unused EC2 instances
- Delete unattached EBS volumes
- Set up lifecycle policies for snapshots
- Use AWS Resource Groups Tag Editor to find untagged resources

2. Data Transfer Costs Cross-Region Transfer:

- Data transfer between regions ($0.02/GB)
- Not using same-region resources

Internet Data Transfer:

- Data transfer out to internet ($0.09/GB for first 10TB)
- Large file downloads
- Streaming video/audio

Solutions:

- Keep resources in same region
- Use CloudFront for content delivery (cheaper egress)
- Compress data before transfer
- Use VPC endpoints to avoid NAT Gateway data processing charges
- Review CloudFront, S3, and EC2 data transfer in Cost Explorer

3. NAT Gateway Costs High Data Processing:

- NAT Gateway charges for data processed ($0.045/GB)
- Instances in private subnets accessing internet frequently

Solutions:

- Use VPC endpoints for AWS services (S3, DynamoDB)
- Consolidate NAT Gateways (one per AZ sufficient)
- Review what traffic is going through NAT Gateway
- Consider switching to NAT instances for high-volume use cases

4. CloudWatch Logs Large Log Ingestion:

- Application logging too verbosely
- Retention period too long
- Many log groups

Check Costs:

- Ingestion: $0.50/GB
- Storage: $0.03/GB/month
- Insights queries: $0.005/GB scanned

Solutions:

- Reduce log verbosity

- Set retention policies (7-30 days typical)
- Export old logs to S3 (cheaper storage)
- Use sampling for high-volume logs
- Delete unnecessary log groups

5. Elastic Load Balancers Idle Load Balancers:

- Load balancer running with no traffic
- Using multiple load balancers when one suffices

Costs:

- ALB/NLB: ˜$0.0225/hour (˜$16/month) + data processing
- Classic LB: ˜$0.025/hour (˜$18/month)

Solutions:

- Delete unused load balancers
- Consolidate applications behind fewer load balancers
- Use path-based routing on ALB

6. RDS Instances Over-Provisioned:

- Database instance too large
- Multi-AZ when not needed for dev/test
- Not using Reserved Instances

Solutions:

- Right-size instance based on CloudWatch metrics
- Use Single-AZ for non-production
- Stop RDS instances when not in use (dev/test)
- Purchase Reserved Instances for production (save up to 72%)

7. S3 Storage Costs Incorrect Storage Class:

- Using Standard for infrequently accessed data
- Not using Intelligent-Tiering

Many Small Objects:

- S3 charges per request
- Millions of tiny files more expensive

Solutions:

- Use lifecycle policies to transition to cheaper storage classes
- Enable S3 Intelligent-Tiering for unknown access patterns
- Consolidate small objects
- Delete incomplete multipart uploads
- Use S3 Storage Lens for insights

8. Lambda Costs High Invocations:

- Infinite loop or recursive calls
- Too frequent CloudWatch Events triggers
- Over-allocated memory

Solutions:

- Review CloudWatch Logs for errors causing retries
- Optimize function execution time
- Right-size memory allocation
- Use reserved concurrency to limit costs
- Implement exponential backoff for retries

Cost Analysis Tools: **AWS Cost Explorer**:

- View costs by service, region, tag
- Identify trends and anomalies
- Filter by time period

AWS Budgets:

- Set budget alerts
- Get notified when exceeding threshold
- Forecast spending

AWS Cost Anomaly Detection:

- ML-based anomaly detection
- Automatic alerts for unusual spending
- Root cause analysis

AWS Trusted Advisor:

- Cost optimization recommendations
- Identify idle resources
- Right-sizing suggestions (with Business/Enterprise support)

Tag-Based Cost Allocation:

- Tag resources by: Project, Environment, Owner
- Enable tag-based cost allocation reports
- Identify costs by business unit

Investigation Steps:

1. **Open Cost Explorer**:

- Group by service
- Identify top cost services
- Compare to previous month

1. **Check for Anomalies**:

- Look for sudden spikes
- Identify specific days/hours

1. **Review Top Services**:

- EC2: Running instances, EBS volumes
- S3: Storage, requests, data transfer
- Data Transfer: Cross-region, internet egress
- RDS: Running databases

1. **Tag Analysis**:

- Identify untagged resources
- Track costs by project/team

1. **Enable Detailed Billing**:

- Resource-level granularity
- Understand what's driving costs

Prevention:

1. Set up AWS Budgets with email alerts
2. Tag all resources appropriately
3. Set up Cost Anomaly Detection
4. Review Cost Explorer monthly
5. Implement least-privilege IAM policies (prevent accidental expensive resource creation)
6. Use CloudFormation with budget constraints
7. Enable AWS Cost Optimization Hub
8. Regular cost review meetings with stakeholders

—

API Gateway 502/504 Errors

Symptoms: API Gateway returns 502 Bad Gateway or 504 Gateway Timeout
Error Types:

1. 502 Bad Gateway Causes:

- Backend endpoint (Lambda, HTTP) returning invalid response
- Lambda function error/exception
- Malformed response from integration
- Certificate validation failure (for HTTP integration)

Solutions: **Lambda Integration**:

- Check CloudWatch Logs for Lambda errors
- Ensure Lambda returns proper response format:

```
\{
  "statusCode": 200,
  "headers": \{
    "Content-Type": "application/json"
  \},
  "body": "\{\textbackslash\{\}"message\textbackslash\{\}":\
     textbackslash\{\}"Success\textbackslash\{\}"\}"
\}
```

- Verify Lambda execution role has required permissions
- Check if Lambda is in VPC and can reach required resources

HTTP Integration:

- Verify backend endpoint is accessible
- Check SSL certificate is valid
- Test endpoint directly from EC2 in same VPC
- Verify security groups allow API Gateway to reach backend
- Check if using correct HTTP method

VPC Link Issues:

- Network Load Balancer health checks failing
- Security groups blocking traffic
- Target group has no healthy targets

2. 504 Gateway Timeout Causes:

- Backend taking longer than API Gateway timeout (29 seconds maximum)
- Lambda function timeout
- HTTP endpoint not responding
- Network connectivity issues

Solutions: **Lambda Timeout**:

- Check Lambda timeout setting (max 15 minutes, but API Gateway limit is 29 seconds)
- Set Lambda timeout to ¡29 seconds for synchronous invocations
- For long-running tasks, use asynchronous invocation or Step Functions
- Optimize Lambda performance

HTTP Endpoint Timeout:

- Reduce backend processing time
- Implement caching at backend
- Use asynchronous processing for long operations
- Return immediate response, process in background

VPC Configuration:

- If Lambda in VPC, check it can reach endpoints (NAT Gateway for internet)
- Verify DNS resolution working
- Check VPC Flow Logs for dropped packets

3. Integration Response Issues Invalid Response Transformation:

- VTL (Velocity Template Language) mapping error
- Headers not properly formatted
- Response body invalid JSON

Solutions:

- Test mapping templates in API Gateway console
- Verify response structure matches defined model
- Check for syntax errors in VTL templates
- Enable CloudWatch logging for API Gateway

4. Resource Policy or Authorization 403 Forbidden disguised as 502:

- Resource policy denying request
- Lambda authorizer denying access but not returning proper response

Solutions:

- Review resource policy
- Check Lambda authorizer CloudWatch Logs
- Ensure authorizer returns proper policy document

5. Throttling TooManyRequestsException:

- Account-level throttle (10,000 RPS default)
- Burst limit exceeded (5,000 default)
- Stage-level or method-level throttles

Solutions:

- Implement client-side retry with exponential backoff
- Request limit increase
- Use usage plans to control access
- Implement caching to reduce backend calls

Debugging Steps: **1. Enable CloudWatch Logs**:

```
\# Enable execution logging
aws apigateway update-stage \textbackslash\{\}
  --rest-api-id abc123 \textbackslash\{\}
  --stage-name prod \textbackslash\{\}
  --patch-operations \textbackslash\{\}
    op=replace,path=/*/logging/loglevel,value=INFO \
       textbackslash\{\}
    op=replace,path=/*/logging/dataTrace,value=true
```

2. Check CloudWatch Metrics:

- 4XXError: Client errors
- 5XXError: Server errors
- IntegrationLatency: Backend response time
- Latency: Total request latency
- Count: Number of requests

3. Test Endpoint:

```
\# Test API directly
curl -X POST https://api-id.execute-api.region.amazonaws.com/
   stage/path \textbackslash\{\}
  -H "Content-Type: application/json" \textbackslash\{\}
  -d '\{"key":"value"\}' \textbackslash\{\}
  -v

\# Check for specific error codes
\# 502: Bad Gateway (integration error)
\# 504: Gateway Timeout (backend timeout)
```

4. Review Lambda Logs (if Lambda integration):

```
\# Get latest log stream
aws logs describe-log-streams \textbackslash\{\}
  --log-group-name /aws/lambda/my-function \textbackslash\{\}
  --order-by LastEventTime \textbackslash\{\}
  --descending \textbackslash\{\}
  --max-items 1

\# View logs
aws logs tail /aws/lambda/my-function --follow
```

5. Test Lambda Directly:

```
\# Invoke Lambda with test event
aws lambda invoke \textbackslash\{\}
  --function-name my-function \textbackslash\{\}
  --payload '\{"key":"value"\}' \textbackslash\{\}
  response.json
```

Common Solutions:

1. **Lambda Response Format**:

- Use proxy integration for simple cases
- Ensure statusCode, headers, body are properly formatted
- Stringify JSON body

1. **Timeout Configuration**:

- Lambda timeout: ¡29 seconds
- Use async invocation for long-running tasks
- Implement caching

1. **VPC Configuration**:

- Add NAT Gateway for internet access
- Use VPC endpoints for AWS services

- Verify security groups

1. **Error Handling**:

- Implement try-catch in Lambda
- Return proper error responses
- Log errors to CloudWatch

1. **Monitoring**:

- Set up CloudWatch alarms for 5XX errors
- Enable X-Ray tracing for detailed analysis
- Regular review of CloudWatch Logs

Best Practices:

1. Always enable CloudWatch Logs (at least for errors)
2. Implement proper error handling in backend
3. Set appropriate timeouts (Lambda ¡ 29 seconds)
4. Use X-Ray for distributed tracing
5. Test API thoroughly before production
6. Monitor latency and error rates
7. Implement caching to reduce backend load
8. Use usage plans to control access and prevent abuse

—

9.0.4 Key Takeaways

1. **Cost Optimization**: Combine Reserved Instances, Spot Instances, and On-Demand based on workload patterns
2. **High Availability**: Always design across multiple Availability Zones
3. **Data Migration**: Use AWS physical devices (Snowball/Snowmobile) for large datasets
4. **Serverless**: Ideal for unpredictable workloads and minimal operational overhead
5. **Compliance**: Use AWS Organizations, SCPs, and AWS Config for governance at scale
6. **Disaster Recovery**: Choose DR strategy based on RPO/RTO requirements and budget
7. **Hybrid Connectivity**: Direct Connect for production, VPN for dev/test
8. **Multi-Region**: Use Global Tables, CloudFront, and Route 53 for low-latency global access

9. **Security**: Implement defense in depth with GuardDuty, Security Hub, Config, and automated remediation
10. **Modernization**: Use strangler fig pattern for incremental migration from monoliths
11. **Big Data**: Build data lakes with S3, process with Glue/EMR, analyze with Athena/Redshift
12. **CI/CD**: Automate deployments with CodePipeline, implement blue/green deployments, enable fast rollbacks
13. **Troubleshooting**: Follow systematic approaches for connectivity, security, and performance issues
14. **Cost Management**: Use Cost Explorer, Budgets, and tagging to track and optimize spending

— ← Previous: Exam Preparation — Next: Additional Resources →

Chapter 10

Additional AWS Services and Advanced Topics

10.1 Developer Tools and CI/CD

10.1.1 AWS CodeCommit

- Git-based source control repository
- Secure, highly available
- No size limits on repositories
- Integrates with existing Git tools
- Encrypted at rest and in transit
- Free tier: 5 active users per month

10.1.2 AWS CodeBuild

- Fully managed build service
- Compiles source code, runs tests, produces packages
- Scales automatically
- Pay only for build time
- Pre-configured environments or custom Docker images
- Integrates with CodeCommit, GitHub, Bitbucket

10.1.3 AWS CodeDeploy

- Automated deployment service
- Deploy to EC2, Lambda, on-premises servers
- Deployment strategies: In-place, blue/green
- Automatic rollback on failure
- Integration with existing CI/CD tools
- No additional charge (pay for resources)

10.1.4 AWS CodePipeline

- Continuous delivery service
- Automates release pipeline
- Integrates with CodeCommit, CodeBuild, CodeDeploy
- Third-party integrations (GitHub, Jenkins)
- Visual workflow builder
- $1 per active pipeline per month

10.1.5 AWS CodeStar

- Unified user interface for development activities
- Quickly develop, build, and deploy applications
- Project templates for various languages and platforms
- Integrated dashboard for monitoring
- Team collaboration features
- No additional charge

10.1.6 AWS Cloud9

- Cloud-based IDE (Integrated Development Environment)
- Write, run, and debug code in browser
- Supports 40+ programming languages
- Built-in terminal with AWS CLI
- Collaborative coding features
- Pay only for underlying EC2 instance

10.2 Application Integration Services

10.2.1 Amazon EventBridge

- Serverless event bus service
- Connect applications using events
- Formerly CloudWatch Events
- Event sources: AWS services, custom applications, SaaS apps
- Event patterns for filtering
- Multiple targets per rule
- Pay per event

10.2.2 Amazon MQ

- Managed message broker service
- Supports Apache ActiveMQ and RabbitMQ
- For migrating existing applications using message brokers
- Alternative to SNS/SQS for specific protocols
- Industry-standard APIs and protocols (MQTT, AMQP, STOMP)
- Single-instance or active/standby deployment

10.2.3 AWS App Mesh

- Service mesh for microservices
- Monitor and control communications
- Works with ECS, EKS, EC2
- Provides observability, traffic management
- Based on Envoy proxy
- No additional charge (pay for resources)

10.3 End User Computing

10.3.1 Amazon WorkSpaces

- Managed Desktop-as-a-Service (DaaS)
- Virtual Windows or Linux desktops
- Access from any device
- Persistent desktop storage
- Integrated with Active Directory
- Pricing: Monthly or hourly
- Use cases: Remote work, contractors, BYOD

10.3.2 Amazon AppStream 2.0

- Application streaming service
- Stream desktop applications to browser
- No need to install applications locally
- Scales automatically
- Pay as you go
- Use cases: Training, POC, software trials

10.3.3 Amazon WorkDocs

- Secure document storage and collaboration
- Similar to Dropbox or Google Drive
- 1 TB storage per user
- File comments and feedback
- Active Directory integration
- Mobile and desktop apps

10.3.4 Amazon WorkLink

- Secure mobile access to internal websites
- No VPN required
- Renders content in browser on AWS
- Only pixels sent to device
- Protects corporate network
- Per user per month pricing

10.4 IoT Services

10.4.1 AWS IoT Core

- Connect IoT devices to cloud
- Supports billions of devices
- MQTT, HTTPS, WebSockets protocols
- Device shadow for state management
- Rules engine for data processing
- Integration with other AWS services

10.4.2 AWS IoT Greengrass

- Extend AWS to edge devices
- Local compute, messaging, and data caching
- Run Lambda functions at the edge
- Operate offline
- Secure communication with cloud
- ML inference at edge

10.4.3 AWS IoT Analytics

- Analytics for IoT data
- Process and analyze IoT data
- Pre-built analytics templates
- Integration with QuickSight
- SQL queries on time-series data
- Machine learning integration

10.5 Media Services

10.5.1 Amazon Elastic Transcoder

- Convert media files between formats
- Scalable media transcoding
- Pre-configured presets
- Pay per minute of transcoding
- Integration with S3, CloudFront

10.5.2 AWS Elemental MediaConvert

- File-based video transcoding
- Broadcast-grade features
- Supports various formats and codecs
- On-demand or reserved pricing
- More advanced than Elastic Transcoder

10.5.3 Amazon Kinesis Video Streams

- Capture, process, and store video streams
- Millions of devices
- Playback, analytics, ML integration
- Use cases: Security cameras, live streaming
- Pay for data ingested and consumed

10.6 Additional AI/ML Services

10.6.1 Amazon Polly

- Text-to-speech service
- Natural sounding speech
- Multiple languages and voices
- Neural TTS for more natural sound
- Pay per character
- Use cases: E-learning, accessibility

10.6.2 Amazon Transcribe

- Automatic speech recognition (ASR)
- Convert speech to text
- Real-time and batch processing
- Speaker identification
- Custom vocabulary
- Pay per second of audio

10.6.3 Amazon Translate

- Neural machine translation
- Translate text between languages
- 75+ languages supported
- Custom terminology
- Pay per character
- Use cases: Localization, content creation

10.6.4 Amazon Forecast

- Time-series forecasting service
- Uses machine learning
- No ML expertise required
- More accurate than traditional methods
- Use cases: Demand forecasting, resource planning

10.6.5 Amazon Kendra

- Intelligent search service
- ML-powered enterprise search
- Natural language queries
- Learns from user interactions
- Connects to various data sources

10.6.6 Amazon Personalize

- Real-time recommendations
- Same technology as Amazon.com
- No ML expertise required
- Real-time and batch recommendations
- Use cases: Product recommendations, personalized content

10.6.7 Amazon Textract

- Extract text and data from documents
- OCR and form recognition
- Preserves structure and relationships
- Works with PDFs, images
- Pay per page

10.7 Business Applications

10.7.1 Amazon Connect

- Cloud-based contact center
- Omnichannel customer service
- AI-powered chatbots
- Pay as you go
- Integration with other AWS services
- Use cases: Customer support, helpdesk

10.7.2 Amazon Simple Email Service (SES)

- Email sending and receiving
- Transactional and marketing emails
- High deliverability
- Pay per email sent
- Free tier: 62,000 emails/month (from EC2)
- Email validation and filtering

10.7.3 Amazon Pinpoint

- Marketing communication service
- Email, SMS, push notifications
- Customer segmentation
- Campaign management
- Analytics and engagement metrics
- Pay for messages sent

10.7.4 AWS WorkMail

- Managed email and calendar service
- Alternative to Exchange/Gmail
- Web-based access
- Mobile and desktop clients
- Active Directory integration
- $4 per user per month

10.7.5 Amazon Chime

- Video conferencing and communication
- Meetings, chat, business calling
- Screen sharing
- Per user per day pricing
- Alternative to Zoom, Teams

10.8 Management and Governance (Advanced)

10.8.1 AWS License Manager

- Manage software licenses
- Track license usage
- Set usage limits
- Prevent license violations
- Works with Microsoft, Oracle, SAP, etc.

10.8.2 AWS Service Catalog

- Create and manage catalogs of IT services
- Standardized products
- Control which services users can deploy
- Version control for products
- Governance and compliance
- Self-service portal for end users

10.8.3 AWS Well-Architected Tool

- Review workload architecture
- Compare against best practices
- Six pillars assessment
- Get improvement plan
- Free service
- Periodic reviews recommended

10.8.4 AWS Personal Health Dashboard

- Personalized view of AWS service health
- Alerts for service events affecting you
- Proactive notifications
- Remediation guidance
- Available to all customers
- Different from Service Health Dashboard (global status)

10.8.5 AWS Compute Optimizer

- Recommends optimal AWS resources
- ML-based recommendations
- Analyzes historical utilization
- Suggests EC2 instance types, EBS volumes, Lambda memory
- Cost savings opportunities
- No additional charge

10.9 Migration and Transfer Services

10.9.1 AWS Application Discovery Service

- Discover on-premises applications
- Plan migrations
- Collect server utilization and dependency data
- Agentless or agent-based discovery
- Export data for analysis
- Integration with Migration Hub

10.9.2 AWS Migration Hub

- Track application migrations
- Single location to monitor migrations
- Works with migration tools
- Visualize migration progress
- No additional cost

10.9.3 AWS Server Migration Service (SMS)

- Migrate on-premises servers to AWS
- Incremental replication
- Minimal downtime
- Supports VMware, Hyper-V, Azure
- No additional charge
- Being replaced by Application Migration Service

10.9.4 AWS DataSync

- Transfer data between on-premises and AWS
- Up to 10x faster than open-source tools
- Automated data transfer
- Supports NFS, SMB protocols
- Transfers to S3, EFS, FSx
- Pay per GB transferred

10.9.5 AWS Transfer Family

- Fully managed SFTP, FTPS, FTP
- Transfer files to/from S3 or EFS
- No infrastructure to manage
- Integration with existing auth systems
- Pay per protocol enabled + data transfer

10.10 Networking (Advanced)

10.10.1 AWS Global Accelerator

- Improve availability and performance
- Uses AWS global network
- Static anycast IP addresses
- Automatic failover
- Health checks
- Use cases: Gaming, IoT, VoIP
- Different from CloudFront (not caching)

10.10.2 AWS App Mesh

- Service mesh for microservices
- Monitor and control communications
- Works with ECS, EKS, EC2
- Based on Envoy proxy
- Traffic management, observability

10.10.3 AWS Cloud Map

- Service discovery
- Register application resources
- Discover services via API or DNS
- Health checking
- Integration with ECS, EKS

10.11 Additional Storage Services

10.11.1 Amazon FSx

- Fully managed file systems
- **FSx for Windows File Server**:
  - Windows native file system
  - SMB protocol
  - Active Directory integration
  - For Windows applications
- **FSx for Lustre**:
  - High-performance file system
  - ML, HPC, video processing
  - Integration with S3
  - Sub-millisecond latencies

10.11.2 AWS Backup

- Centralized backup service
- Automate and manage backups
- Backup across AWS services
- Backup policies and retention
- Compliance reporting
- Pay for storage used

10.12 Blockchain and Quantum

10.12.1 Amazon Managed Blockchain

- Create and manage blockchain networks
- Supports Hyperledger Fabric and Ethereum

- Fully managed
- Scales automatically
- Use cases: Supply chain, finance

10.12.2 Amazon Braket

- Quantum computing service
- Develop quantum algorithms
- Test on quantum simulators
- Run on quantum hardware
- Pay for simulation and quantum tasks

10.13 Exam Tips for Service Selection

10.13.1 Database Selection Decision Tree

Choose Database Based on Requirements:

- **Need SQL and ACID transactions?**
  - Traditional app, existing database → **RDS**
  - Need extreme performance → **Aurora**
  - Specific needs: Oracle/SQL Server → **RDS** with that engine
- **Need NoSQL?**
  - Key-value, scale to millions of requests/sec → **DynamoDB**
  - Document database, MongoDB compatible → **DocumentDB**
  - Graph relationships → **Neptune**
- **Need caching?**
  - In-memory cache → **ElastiCache**
  - Redis features needed → ElastiCache for Redis
  - Simple caching → ElastiCache for Memcached
- **Need data warehouse/analytics?**
  - OLAP, BI, big data → **Redshift**
  - Query data in S3 → **Athena**
  - Real-time analytics → **Kinesis Data Analytics**
- **Time-series data?**
  - IoT, metrics, logs → **Timestream**

Requirement & **Best Choice** & **Why**
Full OS control & EC2 & Complete flexibility
Serverless, event-driven & Lambda & No servers, pay per use
Deploy app quickly & Elastic Beanstalk & PaaS, managed
Containers, Kubernetes & EKS & Kubernetes compatible
Containers, AWS native & ECS & Simpler than EKS
Containers, no servers & Fargate & Serverless containers
Batch processing & AWS Batch & Optimized for batch
Simple website, blog & Lightsail & Predictable pricing

Table 10.1: Compute Service Selection

10.13.2 Compute Selection Flowchart

10.13.3 Storage Selection Guide

Use Case & **Service** & **Reason**
Backups, archives & S3 Glacier & Lowest cost
Website content & S3 + CloudFront & Scalable, global
EC2 boot volume & EBS & Block storage
Shared file system & EFS & Multi-attach

Use Case & **Service** & **Reason**
Windows file shares & FSx for Windows & SMB, AD integration
HPC workloads & FSx for Lustre & High performance
Temporary data & Instance Store & Highest IOPS
Offline transfer & Snow Family & Large data volumes

Table 10.2: Storage Selection Guide

% Study Plan and Exam Preparation - Expanded content

Chapter 11

Study Plan and Exam Preparation

11.1 Recommended Study Timeline

11.1.1 4-Week Study Plan

This comprehensive plan is ideal for beginners with no prior AWS experience.

Week 1: Cloud Concepts and Foundations

Focus Areas:

- Study Domain 1: Cloud Concepts (24% of exam)
- Understand cloud computing fundamentals
- Learn AWS Well-Architected Framework
- Complete AWS Cloud Practitioner Essentials course
- **Hands-on Practice:** Create AWS account, explore console

Daily Breakdown:

- **Days 1-2:** What is cloud computing, deployment models, cloud advantages
- **Days 3-4:** AWS Well-Architected Framework pillars
- **Days 5-6:** AWS global infrastructure, regions, AZs, edge locations
- **Day 7:** Review and practice quiz

Week 2: Security and Compliance

Focus Areas:

- Study Domain 2: Security and Compliance (30% of exam)
- Master Shared Responsibility Model
- Learn IAM thoroughly (users, groups, roles, policies)
- Review security services (Shield, WAF, GuardDuty, Inspector)

- **Hands-on Practice:** Set up IAM users, groups, roles, enable MFA

Daily Breakdown:

- **Days 1-2:** Shared Responsibility Model, IAM basics
- **Days 3-4:** IAM policies, roles, best practices
- **Days 5-6:** Security services, compliance programs, data protection
- **Day 7:** Review and practice quiz

Week 3: Technology and Services

Focus Areas:

- Study Domain 3: Cloud Technology and Services (34% of exam)
- Focus on core services: EC2, S3, RDS, VPC
- Learn other services at high level
- **Hands-on Practice:** Launch EC2 instance, create S3 bucket, set up VPC
- Take mid-point practice exam

Daily Breakdown:

- **Days 1-2:** Compute services (EC2, Lambda, ECS, Elastic Beanstalk)
- **Day 3:** Storage services (S3, EBS, EFS, Storage Gateway)
- **Day 4:** Database services (RDS, DynamoDB, Redshift)
- **Day 5:** Networking services (VPC, CloudFront, Route 53, ELB)
- **Day 6:** Additional services (CloudWatch, SNS, SQS, CloudFormation)
- **Day 7:** Practice exam and review weak areas

Week 4: Billing and Review

Focus Areas:

- Study Domain 4: Billing, Pricing, and Support (12% of exam)
- Memorize support plans and response times
- Review all domains
- Take multiple practice exams
- Review weak areas
- Final review day before exam

Daily Breakdown:

- **Days 1-2:** Pricing models, cost optimization, Free Tier
- **Day 3:** Support plans, Trusted Advisor, AWS Organizations
- **Days 4-5:** Take 2-3 full practice exams
- **Day 6:** Review all weak areas and flagged topics
- **Day 7:** Light review only, rest before exam

11.1.2 2-Week Intensive Plan

For those with IT background or time constraints.

Week 1: Core Content

Days 1-2: Cloud Concepts and Security

- Cloud computing fundamentals
- AWS Well-Architected Framework
- Shared Responsibility Model
- IAM complete coverage

Days 3-5: Core Services

- Compute: EC2, Lambda, Elastic Beanstalk
- Storage: S3, EBS, EFS
- Database: RDS, DynamoDB
- Networking: VPC, CloudFront, Route 53, ELB

Days 6-7: Remaining Services and Practice Exam

- CloudWatch, CloudTrail, Config
- SNS, SQS, Lambda
- CloudFormation, Elastic Beanstalk
- Take first practice exam

Week 2: Billing and Review

Days 8-9: Billing, Pricing, and Support

- All pricing models (On-Demand, Reserved, Spot, Savings Plans)
- Support plans and TAM
- Cost optimization strategies
- Billing tools (Cost Explorer, Budgets, Pricing Calculator)

Days 10-12: Practice Exams and Review

- Take 3-4 practice exams
- Review all weak areas
- Focus on commonly confused topics

Days 13-14: Final Review and Rest

- Light review of key concepts
- No new material
- Rest and prepare mentally

11.2 Exam Registration Process

11.2.1 Step 1: Create AWS Training and Certification Account

1. Go to https://www.aws.training/
2. Click "Sign In" or "Create Account"
3. Provide email address and create password
4. Complete account profile information
5. Verify email address

11.2.2 Step 2: Schedule Your Exam

1. Log in to AWS Certification Account
2. Go to "Certifications" → "AWS Certification Account"
3. Click "Schedule New Exam"
4. Select "AWS Certified Cloud Practitioner (CLF-C02)"
5. Choose exam delivery method:
   - **Pearson VUE Testing Center:** In-person at authorized location
   - **Online Proctored:** Take from home/office with live proctor

11.2.3 Step 3: Select Date, Time, and Location

For Testing Center:

1. Enter your location/zip code
2. Select preferred testing center
3. Choose available date and time slot
4. Review and confirm

For Online Proctored:

1. Select available date and time
2. Review system requirements
3. Run system test to verify your computer meets requirements
4. Confirm workspace requirements (quiet, private room)

11.2.4 Step 4: Pay for Exam

- **Cost:** $100 USD
- **Payment methods:** Credit/debit card
- **Cancellation policy:** Free cancellation/reschedule up to 24 hours before exam

11.2.5 Step 5: Prepare for Exam Day

- Receive confirmation email
- Note exam date, time, and location (or online link)
- Review exam policies
- Prepare required identification

Note: AWS offers exam vouchers through partner programs and AWS Training events. Check for available discounts.

11.3 Test-Taking Strategies

11.3.1 Before the Exam

1. **Review all exam objectives** – Ensure you've covered each domain thoroughly
2. **Take practice exams** – Identify weak areas and improve
3. **Get hands-on experience** – Create AWS account, experiment with services
4. **Review AWS documentation** – Especially FAQs for key services
5. **Get adequate rest** – Sleep well before exam day
6. **Light review only** – Don't cram new material the night before

11.3.2 During the Exam

Reading Questions Carefully

Pay special attention to keywords:

- **"MOST cost-effective"** – Focus on pricing and efficiency
- **"BEST"** – Look for AWS-recommended practices
- **"LEAST amount of effort"** – Simplest solution, often managed services
- **"NOT" or "EXCEPT"** – Looking for the wrong answer
- **"Select TWO/THREE"** – Multiple correct answers required

Answer Strategies

1. **Eliminate wrong answers** – Narrow down choices systematically
2. **Watch for absolutes** – "Always," "never," "all," "none" are often wrong
3. **Flag difficult questions** – Return to them later with fresh perspective
4. **Use process of elimination** – Remove obviously incorrect answers first
5. **Trust your preparation** – Your first instinct is often correct
6. **Don't overthink** – AWS prefers simple, managed solutions

Review Process

1. **Review flagged questions** – Revisit difficult questions
2. **Check "Select TWO/THREE" questions** – Verify you selected correct number
3. **Verify "EXCEPT" questions** – Easy to misread these
4. **Use remaining time wisely** – Review all answers if time permits

11.4 Day-of-Exam Tips

11.4.1 For Testing Center Exam

What to Bring

Required:

- **Two forms of valid ID** (both must include name, one with signature, one with photo)
  - Government-issued photo ID (driver's license, passport)
  - Secondary ID (credit card, student ID)

Not Allowed:

- Mobile phones, watches, jewelry
- Bags, books, notes
- Food and drinks
- Electronic devices

Arrival

1. **Arrive 15-30 minutes early** – Allow time for check-in
2. **Check in at reception** – Present IDs
3. **Store belongings** – Use provided locker
4. **Read and sign policies** – Review exam rules
5. **Get settled** – Testing staff will guide you to workstation

During Test

- Scratch paper and pen/pencil provided
- Raise hand for restroom breaks (time continues)
- Remain quiet and professional
- Follow all proctor instructions

11.4.2 For Online Proctored Exam

Before Test Day

1. **Run system test** – Verify computer compatibility (72 hours before)
2. **Check internet connection** – Stable, high-speed connection required
3. **Prepare workspace** – Clear desk, quiet room, no other people
4. **Test webcam and microphone** – Must be working properly

Test Day Setup (30 minutes before)

1. **Close all applications** – Only exam software should run
2. **Remove items from desk** – Only computer, keyboard, mouse allowed
3. **Clear walls around you** – No posters, whiteboards, notes visible
4. **Ensure good lighting** – Face must be clearly visible
5. **Have ID ready** – Government-issued photo ID

During Online Exam

- Proctor will verify your ID via webcam
- Proctor will ask you to show room via webcam
- Must remain in view of camera at all times
- No talking allowed (except to proctor via chat)
- No leaving camera view during exam
- Follow all proctor instructions immediately

11.4.3 What to Expect at Testing Center

1. **Check-in process** – 5-10 minutes
2. **ID verification** – Both forms of ID checked
3. **Photo taken** – Digital photo for records
4. **Rules review** – Brief overview of policies
5. **Locker assignment** – Store all personal items
6. **Escort to workstation** – Staff will guide you
7. **Begin exam** – Launch exam when ready

11.5 Time Management

11.5.1 Exam Time Overview

- **Total duration:** 90 minutes
- **Number of questions:** 65
- **Time per question:** ∼1 minute 23 seconds average
- **Recommended pace:** 60 questions in 70 minutes, 20 minutes for review

11.5.2 Time Management Strategy

First Pass (60-70 minutes)

1. **Easy questions (30-40 seconds each)** – Answer immediately
2. **Medium questions (60-90 seconds each)** – Think through and answer
3. **Difficult questions (flag for later)** – Read, eliminate options, flag, move on

Review Pass (15-20 minutes)

1. **Return to flagged questions** – Fresh perspective often helps
2. **Use process of elimination** – Narrow down to best answer
3. **Make educated guess** – No penalty for wrong answers
4. **Check "Select TWO/THREE" questions** – Verify correct number selected

Final Minutes (5-10 minutes)

1. **Review all answers** – Quick scan if time permits
2. **Verify no unanswered questions** – Must answer all
3. **Check flagged questions one more time** – Final review
4. **Submit with confidence** – You've prepared well

11.5.3 Pacing Tips

- Don't spend more than 2 minutes on any single question
- If stuck, flag and move on
- Keep track of time (displayed on screen)
- Aim to complete first pass with 20 minutes remaining
- Remember: No penalty for guessing

11.6 Question Types and Approaches

11.6.1 Scenario-Based Questions

Format: Describe a situation, ask for best solution **Example:**

> A company needs to store frequently accessed data that requires millisecond retrieval times. Which AWS service should they use?

Approach:

1. Identify key requirements (frequently accessed, millisecond retrieval)
2. Match requirements to service characteristics
3. Eliminate services that don't fit
4. Choose best match (DynamoDB or ElastiCache)

11.6.2 Multiple Response Questions

Format: "Select TWO" or "Select THREE" correct answers **Approach:**

1. Read carefully – note exact number required
2. Evaluate each option independently
3. Use checkboxes (not radio buttons)
4. Verify you selected correct number before moving on

11.6.3 Best Practice Questions

Format: "What is the AWS recommended approach?" **Approach:**

1. Think about AWS Well-Architected Framework
2. Choose managed services over manual solutions
3. Select options that enhance security, reliability, cost-optimization
4. Favor automation and scalability

11.6.4 Cost Optimization Questions

Format: "Which option is MOST cost-effective?" **Approach:**

1. Compare pricing models (On-Demand vs Reserved vs Spot)
2. Consider Free Tier eligibility
3. Look for pay-as-you-go vs upfront costs
4. Choose serverless/managed when appropriate

11.6.5 Security Questions

Format: "Which option is MOST secure?" **Approach:**

1. Apply principle of least privilege
2. Choose encryption options (at rest and in transit)
3. Prefer IAM roles over access keys
4. Select options with MFA and audit trails

11.6.6 High Availability Questions

Format: "Which design ensures high availability?" **Approach:**

1. Look for multi-AZ deployments
2. Consider load balancing
3. Check for redundancy and failover
4. Verify auto-scaling capabilities

11.6.7 "EXCEPT" or "NOT" Questions

Format: "Which is NOT a benefit of..." or "All of the following EXCEPT..." **Approach:**

1. **Read very carefully** – These are reverse questions
2. Mark the question mentally as "find the wrong answer"
3. Evaluate each option
4. Select the one that doesn't fit

11.7 Common Pitfalls to Avoid

11.7.1 Confusing Service Names

Problem: Similar names, different purposes **Common Confusions:**

- **EC2 vs ECS vs EKS vs EBS**
  - EC2: Virtual servers
  - ECS: Container orchestration
  - EKS: Kubernetes management
  - EBS: Block storage for EC2
- **S3 vs EBS vs EFS**
  - S3: Object storage, internet-accessible
  - EBS: Block storage, attached to single EC2

    - EFS: Network file system, multiple EC2 instances
- **CloudWatch vs CloudTrail vs Config**
    - CloudWatch: Monitoring and metrics
    - CloudTrail: API call logging and auditing
    - Config: Resource configuration tracking

11.7.2 Not Understanding Shared Responsibility Model

Problem: Confusion about AWS vs customer responsibilities **Remember:**

- **AWS:** Responsible for "security OF the cloud" (infrastructure, hardware, facilities)
- **Customer:** Responsible for "security IN the cloud" (data, applications, IAM, encryption)

11.7.3 Mixing Up Support Plans

Problem: Confusing response times and features **Key Differences:**

- **Basic:** Free, no technical support
- **Developer:** Email support, 12-24 hour response
- **Business:** 24/7 phone/chat, 1-hour response for production down
- **Enterprise:** TAM, 15-minute response for business-critical down

11.7.4 Forgetting S3 Storage Classes

Problem: Not matching use case to storage class **Remember:**

- **S3 Standard:** Frequently accessed, high durability
- **S3 Standard-IA:** Infrequent access, lower cost
- **S3 One Zone-IA:** Infrequent, single AZ, lowest cost
- **S3 Glacier:** Long-term archive, retrieval times vary
- **S3 Intelligent-Tiering:** Automatic cost optimization

11.7.5 Confusing Pricing Models

Problem: Not knowing when to use which model **Remember:**

- **On-Demand:** Pay per hour/second, no commitment
- **Reserved Instances:** 1-3 year commitment, up to 72% savings
- **Spot Instances:** Bid on spare capacity, up to 90% savings, can be terminated
- **Savings Plans:** Flexible commitment, similar savings to RIs

11.7.6 Not Knowing AWS Global Infrastructure

Problem: Confusing regions, AZs, and edge locations **Remember:**

- **Regions:** Geographic areas with multiple AZs (e.g., us-east-1)
- **Availability Zones:** Isolated data centers within a region (minimum 3 per region)
- **Edge Locations:** CDN endpoints for CloudFront (400+ globally)

11.7.7 Overlooking "EXCEPT" Questions

Problem: Missing the "NOT" or "EXCEPT" keyword **Solution:**

- Read questions twice
- Highlight or mentally note "EXCEPT" questions
- Remember you're looking for the WRONG answer

11.7.8 Assuming Real-World Complexity

Problem: Overthinking solutions **Solution:**

- Choose AWS-recommended simple solutions
- Prefer managed services over DIY
- Don't add unnecessary complexity
- Trust AWS best practices

11.8 Study Resources

11.8.1 Official AWS Resources (Free)

AWS Cloud Practitioner Essentials

- **Type:** Free digital course on AWS Skill Builder
- **Duration:** Approximately 6 hours
- **Content:** Covers all exam domains
- **Link:** https://aws.amazon.com/training/digital/

AWS Exam Guide

- **Type:** Official exam content outline
- **Content:** Lists all topics tested, sample questions
- **Source:** Download from AWS Training and Certification
- **Importance:** Critical – shows exact exam objectives

AWS Whitepapers

Must-read whitepapers:

- Overview of Amazon Web Services
- AWS Well-Architected Framework
- AWS Pricing Overview
- **Link:** https://aws.amazon.com/whitepapers/

AWS Documentation and FAQs

- **Content:** Comprehensive service documentation
- **Focus on:** FAQs for key services (EC2, S3, RDS, VPC, IAM)
- **Link:** https://docs.aws.amazon.com/

AWS Free Tier

- **Purpose:** Hands-on practice at no cost
- **Duration:** 12 months free for many services
- **Always Free:** Some services always free within limits
- **Link:** https://aws.amazon.com/free/

11.8.2 Official AWS Resources (Paid)

AWS Official Practice Exam

- **Cost:** $20 USD
- **Questions:** 20 questions
- **Format:** Same format as real exam
- **Value:** Best indicator of readiness
- **Purchase:** Through AWS Training and Certification portal

AWS Classroom Training

- **Type:** Instructor-led courses
- **Format:** Virtual or in-person
- **Cost:** Varies by location and provider
- **Value:** Comprehensive, interactive learning

11.8.3 Third-Party Resources

Online Courses

- **A Cloud Guru / Pluralsight:** Comprehensive video courses
- **Udemy:** AWS Certified Cloud Practitioner courses (check ratings)
- **Coursera:** AWS Cloud Practitioner specializations
- **LinkedIn Learning:** AWS fundamentals courses

Practice Exams

- **Tutorials Dojo:** Highly recommended, detailed explanations
- **Whizlabs:** Multiple practice tests
- **ExamTopics:** Free community questions (use with caution)

Books

- AWS Certified Cloud Practitioner Study Guide (Sybex)
- AWS Certified Cloud Practitioner Exam Guide
- AWS Certified Cloud Practitioner All-in-One Exam Guide

YouTube Channels

- AWS Online Tech Talks
- FreeCodeCamp AWS courses
- ExamPro (Andrew Brown)
- Stephane Maarek courses

11.9 Final Preparation Checklist

11.9.1 One Week Before Exam

Complete all study materials
Take at least 3 practice exams
Score consistently above 80%
Review all flagged topics
Revisit Shared Responsibility Model
Memorize support plans and response times
Review service comparisons (S3 vs EBS vs EFS, etc.)
Understand pricing models thoroughly
Review AWS global infrastructure
Practice hands-on labs

11.9.2 One Day Before Exam

Light review only – no new material
Don't study intensively – avoid burnout
Review exam logistics (time, location, or online setup)
Prepare identification documents
Test system (if online proctored)
Prepare workspace (if online proctored)
Get good sleep (7-8 hours)
Eat well and stay hydrated
Relax and stay confident

11.9.3 Exam Day Morning

Eat a good breakfast
Arrive early (testing center) or log in early (online)
Bring two forms of ID (testing center)
Test connection and equipment (online proctored)
Clear workspace (online proctored)
Take deep breaths and stay calm
Review key concepts mentally (optional light review)

11.9.4 During Exam

Read questions carefully
Watch for keywords (MOST, BEST, EXCEPT)
Flag difficult questions
Manage time effectively ($\sim$1.5 minutes per question)
Use process of elimination
Answer all questions (no penalty for guessing)
Review flagged questions
Verify "Select TWO/THREE" questions
Submit with confidence

11.10 After the Exam

11.10.1 Immediate Results

- **Preliminary result:** Pass/fail shown immediately on screen
- **Emotional response:** Normal to feel uncertain even if you passed
- **Exit survey:** Optional feedback about exam experience

11.10.2 Official Score Report

- **Timeline:** Within 5 business days
- **Access:** Available in AWS Certification Account
- **Content:**
  - Final score (100-1000 scale, 700 to pass)
  - Performance by domain
  - Pass/fail status

11.10.3 Digital Badge and Certificate

Digital Badge: Available via Credly/Acclaim

- Shareable on LinkedIn, email signature, resume
- Includes verification link
- Available within 1-2 weeks

Certificate: Downloadable from AWS Certification Account

- PDF format
- Official AWS certification logo
- Valid for 3 years

11.10.4 Recertification

- **Validity:** Certification valid for 3 years from exam date
- **Recertification options:**
  - Retake CLF-C02 exam
  - Pass higher-level associate certification (automatically renews Cloud Practitioner)
- **Reminder:** AWS will email reminders before expiration

11.10.5 Next Steps

Continue Learning

1. **Hands-on practice:** Continue experimenting with AWS services
2. **Real-world projects:** Apply AWS to actual problems
3. **Stay updated:** AWS releases new services regularly
4. **Join communities:** AWS user groups, forums, subreddits

Pursue Advanced Certifications

Associate Level:

- AWS Certified Solutions Architect – Associate
- AWS Certified Developer – Associate
- AWS Certified SysOps Administrator – Associate

Specialty Certifications:

- AWS Certified Advanced Networking – Specialty
- AWS Certified Security – Specialty
- AWS Certified Machine Learning – Specialty

Share Your Achievement

- Update LinkedIn profile with certification
- Add badge to email signature
- Share on social media (if desired)
- Include on resume/CV
- Display certificate at workspace

11.10.6 If You Don't Pass

Don't be discouraged:

- Many successful professionals don't pass on first attempt
- Review score report for weak domains
- Focus study on low-scoring areas
- Take more practice exams
- Schedule retake when ready

Retake Policy:

- Must wait 14 days before retaking
- No limit on number of attempts
- Each attempt requires full exam fee

11.11 Key Concepts to Memorize

11.11.1 Numbers to Remember

- **S3 durability:** 99.999999999% (11 nines)
- **Minimum AZs per Region:** 3
- **Edge Locations:** 400+ globally
- **Lambda max execution:** 15 minutes
- **RDS read replicas:** Up to 5
- **Free Tier:** 12 months for EC2, S3, RDS (some services always free)
- **Support response times:** Memorize all tiers

11.11.2 Service Comparisons to Master

- S3 vs EBS vs EFS (storage types)
- RDS vs DynamoDB vs Redshift (databases)
- EC2 vs Lambda vs Elastic Beanstalk (compute)
- CloudWatch vs CloudTrail vs Config (monitoring/logging)
- SNS vs SQS (messaging)
- Security Groups vs NACLs (network security)
- ALB vs NLB vs GLB (load balancers)

Chapter 12

Quick Reference

12.1 Service Cheat Sheet

12.1.1 Compute

- **EC2**: Virtual servers
- **Lambda**: Serverless functions
- **Elastic Beanstalk**: PaaS for web apps
- **Lightsail**: Simple VPS
- **ECS/EKS**: Container orchestration
- **Fargate**: Serverless containers

12.1.2 Storage

- **S3**: Object storage
- **EBS**: Block storage for EC2
- **EFS**: Network file system
- **Storage Gateway**: Hybrid cloud storage
- **Snow Family**: Physical data migration

12.1.3 Database

- **RDS**: Managed relational database
- **Aurora**: High-performance MySQL/PostgreSQL
- **DynamoDB**: NoSQL database
- **ElastiCache**: In-memory cache
- **Redshift**: Data warehouse

12.1.4 Networking

- **VPC**: Virtual private network
- **CloudFront**: CDN
- **Route 53**: DNS
- **ELB**: Load balancing
- **Direct Connect**: Dedicated network connection

12.1.5 Security

- **IAM**: Identity and access management
- **Organizations**: Multi-account management
- **KMS**: Key management
- **Shield**: DDoS protection
- **WAF**: Web application firewall
- **GuardDuty**: Threat detection
- **Macie**: Data privacy

12.1.6 Management

- **CloudWatch**: Monitoring
- **CloudTrail**: API logging
- **Config**: Resource configuration tracking
- **CloudFormation**: Infrastructure as Code
- **Trusted Advisor**: Best practices
- **Systems Manager**: Operations hub

12.2 Acronym Guide

- **AZ**: Availability Zone
- **IAM**: Identity and Access Management
- **VPC**: Virtual Private Cloud
- **EC2**: Elastic Compute Cloud
- **S3**: Simple Storage Service
- **RDS**: Relational Database Service
- **EBS**: Elastic Block Store
- **EFS**: Elastic File System
- **ELB**: Elastic Load Balancing

- **ALB**: Application Load Balancer
- **NLB**: Network Load Balancer
- **CDN**: Content Delivery Network
- **DNS**: Domain Name System
- **NAT**: Network Address Translation
- **ACL**: Access Control List
- **CIDR**: Classless Inter-Domain Routing
- **SLA**: Service Level Agreement
- **RPO**: Recovery Point Objective
- **RTO**: Recovery Time Objective
- **HA**: High Availability
- **DR**: Disaster Recovery

12.3 Exam Day Reminders

Exam Tip

- 90 minutes, 65 questions
- Passing score: 700/1000
- Flag difficult questions for review
- Eliminate obviously wrong answers
- Trust your preparation
- Stay calm and focused

Good luck on your AWS Cloud Practitioner exam!