AWS Interview Q&A For Top MNCs

CONTENTS

---

1. FUNDAMENTAL AWS CONCEPTS

1. What is AWS, and what are its primary services?

AWS (Amazon Web Services) is a comprehensive cloud computing platform provided by Amazon. It offers over 200 services including computing power, storage, databases, networking, security, analytics, and machine learning. The primary services include EC2 (compute), S3 (storage), RDS (relational databases), and VPC (networking). AWS operates on a pay-as-you-go model, eliminating upfront infrastructure costs.

2. What are the three types of cloud computing models, and provide examples of AWS services for each?

The three cloud computing models are:
- **IaaS (Infrastructure as a Service):** Provides virtualized computing resources over the internet. AWS Examples: EC2, VPC, AWS Direct Connect.
- **PaaS (Platform as a Service):** Provides a platform to develop, run, and manage applications. AWS Examples: Elastic Beanstalk, AWS AppSync, AWS Amplify.
- **SaaS (Software as a Service):** Provides fully managed software applications. AWS Examples: Amazon Chime, Amazon WorkMail, Amazon Connect.

3. Explain the difference between an AWS Region and an Availability Zone.

An **AWS Region** is a separate geographical area containing multiple data centers. There are currently 33 AWS regions worldwide. An **Availability Zone (AZ)** is a distinct data center within a region with independent power, cooling, and networking. Each region has multiple AZs (typically 3-4). AZs are isolated from each other to prevent correlated failures, while being connected via low-latency, high-bandwidth networks. This architecture ensures high availability and fault tolerance.

4. What is the AWS Well-Architected Framework, and what are its five pillars?

The AWS Well-Architected Framework provides best practices for building secure, high-performing, resilient, and efficient infrastructure. The five pillars are:

1. **Operational Excellence:** Focuses on running and monitoring systems to deliver business value and continuously improve processes. Emphasis on IaC, monitoring, automation, and incident response.
2. **Security:** Protects data, systems, and assets through risk management. Includes identity management, data encryption, network isolation, and compliance.

3. **Reliability:** Ensures workloads perform as intended and can recover quickly from failures. Involves distributed systems design, redundancy, and automated recovery.
4. **Performance Efficiency:** Optimizes computing resources and adapts to changing requirements. Includes right-sizing, caching, and using managed services.
5. **Cost Optimization:** Minimizes unnecessary expenses while maximizing business value. Involves resource optimization, reserved capacity, and waste elimination.

5. What is the AWS Shared Responsibility Model?

The Shared Responsibility Model divides security responsibilities between AWS and the customer:

AWS Responsibilities (Security OF the Cloud):
- Physical infrastructure security
- Network infrastructure
- Virtualization layer
- Host operating systems
- Service components

Customer Responsibilities (Security IN the Cloud):
- Application data
- Access management (IAM policies)
- Operating system configuration
- Network and firewall configuration
- Client-side encryption
- Application security

The balance shifts based on service type: with IaaS (EC2), customers manage more; with PaaS (Elastic Beanstalk), AWS manages more; with SaaS (Amazon Chime), AWS manages almost everything.

6. What is an Amazon Machine Image (AMI)?

An AMI is a pre-configured template containing all the necessary information to launch an EC2 instance. It includes:

- Operating system (Linux, Windows, macOS)
- Pre-installed applications and software
- Root volume configuration
- Block device mappings for additional storage
- Launch permissions (who can use it)

AMIs can be public (available to all), shared with specific accounts, or private (personal use). You can create custom AMIs from existing instances to standardize deployments across your organization.

7. Describe the AWS Global Infrastructure.

The AWS Global Infrastructure consists of:

Regions: 33 geographic regions worldwide, each containing multiple AZs. Regions are completely independent.

Availability Zones: Within each region, typically 3-4 AZs providing isolated data centers with independent infrastructure.

Edge Locations: Over 500 edge locations and 15 regional caches used by CloudFront (content delivery network) to serve content with low latency.

AWS Direct Connect Locations: Locations where you can establish dedicated network connections between your on-premises infrastructure and AWS.

This global infrastructure enables customers to deploy applications closer to users for lower latency, comply with data residency requirements, and design highly available systems.

8. What is a VPC, and why is it important?

A **Virtual Private Cloud (VPC)** is a logically isolated network within AWS that you fully control. Importance:
- **Network Isolation:** Resources are isolated from other customers' resources
- **Customization:** Define your own IP address range, subnets, route tables, and gateways
- **Security:** Implement security groups, network ACLs, and VPC endpoints
- **Hybrid Connectivity:** Connect to on-premises networks via VPN or AWS Direct Connect

- **Multi-tier Architecture:** Deploy public and private subnets for different application layers

9. Explain the difference between AWS Free Tier, Reserved Instances, and Spot Instances.

AWS Free Tier:
- Enables new customers to explore AWS services for free
- 12 months of free services (EC2 750 hours/month, S3 5GB, RDS 750 hours, Lambda 1 million free requests)
- Some services (CloudWatch, DynamoDB) have perpetual free tiers
- Ideal for learning and testing

Reserved Instances (RI):
- Commit to using instances for 1 or 3 years
- Provides up to 72% discount compared to On-Demand pricing
- Good for predictable, steady-state workloads
- Can be purchased per region or AZ

Spot Instances:
- Spare EC2 capacity available at up to 90% discount
- Can be interrupted with 2-minute notice
- Suitable for fault-tolerant, flexible, and batch workloads
- Requires bidding or using Savings Plans

10. What is Infrastructure as Code (IaC), and what AWS services support it?

Infrastructure as Code is the practice of managing infrastructure through machine-readable files rather than manual processes. Benefits include:
- Version control and tracking changes
- Reproducible deployments
- Reduced manual errors
- Faster provisioning
- Disaster recovery capability

AWS IaC Services:
- **AWS CloudFormation:** Declarative JSON/YAML templates for AWS resources
- **AWS CDK:** Programmatic IaC using Python, Java, TypeScript, etc.

- **AWS Terraform:** Third-party tool with AWS provider support
- **AWS SAM (Serverless Application Model):** Simplified CloudFormation for serverless applications

---

2. EC2 AND COMPUTE SERVICES

11. What is Amazon EC2, and what are the key instance types?

EC2 (Elastic Compute Cloud) provides scalable virtual servers. Key instance types:

General Purpose (T, M families):
- Balanced compute, memory, and networking
- Use: Web servers, small databases, development environments
- Examples: t3.micro, m5.large

Compute Optimized (C family):
- High-performance processors
- Use: Batch processing, media transcoding, high-traffic APIs
- Examples: c5.xlarge, c5.2xlarge

Memory Optimized (R, X families):
- Large in-memory caches and databases
- Use: In-memory databases, real-time analytics, SAP workloads
- Examples: r5.xlarge, x1.32xlarge

Storage Optimized (I, H, D families):
- High sequential read/write access
- Use: NoSQL databases, data warehousing, Elasticsearch
- Examples: i3.2xlarge, h1.8xlarge

Accelerated Computing (P, G, F families):
- GPU, FPGA, TPU support
- Use: Machine learning, graphics rendering, video processing
- Examples: p3.2xlarge (GPU), f1.2xlarge (FPGA)

12. What is the difference between stopping, terminating, and rebooting an EC2 instance?

Stopping an Instance:
- Performs graceful shutdown
- Instance moves to “stopped” state
- EBS volumes remain attached (data preserved)
- No compute charges, but EBS storage charges continue
- Can be restarted, maintaining private IP (if in same subnet)
- Use when: Temporary pause or configuration changes

Terminating an Instance:
- Permanently deletes the instance
- EBS root volume is deleted by default
- All associated Elastic IPs are released
- Stopped instance charged for EBS, not compute
- Cannot be restarted
- Use when: No longer needed

Rebooting an Instance:
- Equivalent to restarting a computer
- Instance remains in “running” state
- IP addresses unchanged
- Takes 1-5 minutes typically
- No charge changes
- Use when: OS updates, software installation

13. Explain the different EC2 pricing models.

On-Demand Instances:
- Pay per hour/second used
- No long-term commitments
- Highest cost but maximum flexibility
- Good for: Development, testing, variable workloads
- Useful for: Unpredictable traffic patterns

Reserved Instances (RI):
- 1-year or 3-year commitment
- Up to 72% discount vs On-Demand (40% for 1-year)
- Upfront payment options: All Upfront, Partial Upfront, No Upfront

- Good for: Steady-state production workloads
- Can be sold in Reserved Instance Marketplace

Spot Instances:
- Up to 90% discount vs On-Demand
- Can be interrupted with 2-minute notice
- Good for: Batch jobs, data processing, testing
- Cannot be interrupted: Use Capacity Reservations or Savings Plans

Savings Plans:
- Commitment to usage (1 or 3 years)
- 20-40% discount vs On-Demand
- More flexible than RI (works across instance families)
- Good for: Organizations with variable workloads

14. What are Auto Scaling Groups, and how do they work?

An **Auto Scaling Group (ASG)** automatically adjusts the number of EC2 instances based on demand. Components:

Launch Template/Configuration:
- Specifies AMI, instance type, key pair, security groups, storage
- Can include user data scripts for initialization

Scaling Policies:
1. **Target Tracking:** Maintains a specific metric (e.g., 70% CPU)
2. **Step Scaling:** Scales based on CloudWatch alarm thresholds
3. **Simple Scaling:** Single threshold triggers scale-in/out
4. **Scheduled Scaling:** Scales at predetermined times

Capacity Configuration:
- Minimum capacity: Always maintain at least this many instances
- Maximum capacity: Never exceed this number
- Desired capacity: Target number to maintain

Health Checks:
- EC2 status checks (default)
- ELB health checks
- Custom health checks

Lifecycle Hooks:
- Add delay before scaling in/out
- Run scripts during scaling events

15. What is Elastic Load Balancing, and what are the different types?

Elastic Load Balancing (ELB) distributes incoming traffic across multiple targets for high availability and fault tolerance.

Application Load Balancer (ALB):
- Layer 7 (Application) routing
- Path-based routing (/images, /api)
- Host-based routing (domain routing)
- Hostname and header-based routing
- Best for: Microservices, REST APIs, web applications
- Supports: HTTP/HTTPS, WebSocket

Network Load Balancer (NLB):
- Layer 4 (Transport) ultra-high performance
- Millions of requests per second
- Extremely low latency (<100 microseconds)
- Support for static IPs and elastic IPs
- Best for: Gaming, IoT, non-HTTP protocols, extreme performance
- Supports: TCP, UDP, TLS

Gateway Load Balancer (GWLB):
- Layer 3 (Network) for third-party appliances
- Scales virtual appliances (firewalls, IDS/IPS)
- Best for: Deep packet inspection, network security
- Preserves source/destination IP

Classic Load Balancer (CLB):
- Legacy, Layer 4 and 7 support
- Connection pooling, sticky sessions
- Not recommended for new applications

16. What are Security Groups and how do they differ from Network ACLs?

Security Groups:
- Stateful firewall at instance level
- Allow rules only (default deny)
- Inbound rules specified, outbound rules inferred
- Applied to EC2 network interfaces
- Changes take effect immediately
- All rules evaluated before deciding to allow
- Up to 60 rules per security group

Network ACLs (Access Control Lists):
- Stateless firewall at subnet level
- Both allow and deny rules
- Process rules in order (top to bottom)
- Applied to entire subnets
- Changes require explicit configuration
- Changes take a few seconds
- Up to 20 rules per NACL

Key Differences:
- **Scope:** Security Groups are per-instance, NACLs are per-subnet
- **Statefulness:** Security Groups maintain state, NACLs don't
- **Rule Direction:** Security Groups infer return traffic, NACLs require explicit outbound rules
- **Default Behavior:** Security Groups deny by default, NACLs allow by default

17. Explain Elastic IP addresses and when to use them.

Elastic IP (EIP):
- Static public IPv4 address owned by your account
- Independent of instance lifetime
- Can be associated/disassociated on demand
- Remains associated when instance stops/starts
- Useful for DNS records (static endpoint)
- Charges apply when not associated with running instance

When to Use:
- Static website hosting endpoints
- DNS entries requiring fixed IPs

- Failover scenarios (quickly reassign to standby instance)
- Third-party integrations requiring fixed IPs

vs. Public IP:
- Public IP is dynamic (changes on stop/start)
- Public IP is released when instance terminates
- EIP remains with account across instance lifecycle

18. What is AWS Lambda, and what are its key characteristics?

AWS Lambda is a serverless compute service running code without managing servers.

Key Characteristics:
- **Event-driven:** Triggered by AWS services (S3, DynamoDB, API Gateway)
- **Stateless:** Each invocation is independent
- **Auto-scaling:** Automatically handles concurrent requests
- **Pay-per-use:** Charged for execution time (1ms granularity) and requests
- **Timeout:** Maximum 15 minutes (900 seconds) per invocation
- **Memory:** 128MB to 10,240MB (128MB increments)
- **Environment:** No direct server access
- **Languages:** Python, Node.js, Java, C#, Go, Ruby, custom runtimes

Advantages:
- No infrastructure management
- Lower costs for variable workloads
- Quick scaling
- Easy integration with AWS services

Limitations:
- Cold start latency
- Maximum execution time
- Temporary storage (/tmp) limited to 512MB
- Not suitable for long-running processes

19. What is AWS Elastic Beanstalk?

Elastic Beanstalk is a PaaS that simplifies application deployment and management.

Features:
- Automatic provisioning of EC2, RDS, S3, CloudFront
- Handles scaling, load balancing, monitoring
- Supports multiple languages and frameworks
- Application versioning and environment switching
- Rolling deployments with zero downtime
- Integration with AWS CodePipeline for CI/CD

Supported Platforms:
- Node.js, Python, Ruby, Java, PHP, Go, .NET
- Docker support for custom environments

Deployment Options:
- Single instance (for development)
- Load-balanced autoscaling (for production)
- Worker tier (for background jobs)

20. Explain the difference between EC2 and Lambda for a use case.

Scenario: Video thumbnail generation on image upload

Lambda Approach:
- S3 upload triggers Lambda function
- Function generates thumbnail in /tmp
- Uploads result to S3
- No servers to manage
- Cost: ~$0.20 per 1 million requests (assuming 1GB-sec memory)
- Cold start latency: 1-3 seconds

EC2 Approach:
- EC2 instance listening for S3 events
- Processes thumbnails
- Minimum: $20/month (t3.micro)
- Scales via Auto Scaling Group
- Full control but operational overhead

Best for Lambda:
- Unpredictable, sporadic workloads
- Short-duration tasks (<15 minutes)

- Event-driven workflows
- Budget-sensitive, variable traffic

Best for EC2:
- Long-running processes
- Continuous workloads
- Need for full OS access
- Complex dependencies

---

3. STORAGE SERVICES (S3, EBS, EFS)

21. What is Amazon S3 and its key features?

Amazon S3 (Simple Storage Service) is object storage for storing and retrieving unlimited data.

Key Features:
- **Scalability:** Unlimited storage capacity
- **Durability:** 99.999999999% (11 nines)
- **Availability:** 99.99% (standard)
- **Accessibility:** HTTP/HTTPS from anywhere
- **Versioning:** Multiple object versions in same bucket
- **Lifecycle policies:** Automatic transition/deletion
- **Server-side encryption:** SSE-S3, SSE-KMS, SSE-C
- **Static website hosting:** Host static websites
- **Event notifications:** Trigger Lambda/SNS/SQS on uploads
- **Cross-region replication:** Replicate to other regions
- **Transfer acceleration:** CloudFront-backed uploads

Components:
- **Bucket:** Container (namespace is global)
- **Key:** Object identifier within bucket
- **Object:** Data + metadata
- **Region:** Geographic location for bucket

22. Describe S3 storage classes and their use cases.

S3 Standard:
- Default storage class

- Frequent access, millisecond retrieval
- 99.99% availability
- Cost: Highest per GB
- Use: Active data, frequently accessed files

S3 Standard-IA (Infrequent Access):
- Infrequent access but rapid retrieval
- Minimum 30-day storage and 128KB object size
- 99.9% availability
- Lower storage cost, retrieval charges
- Use: Backup, disaster recovery

S3 One Zone-IA:
- Single AZ (lower durability)
- Infrequent access, rapid retrieval
- Lower cost than Standard-IA
- Use: Reproducible data, backup copies

S3 Intelligent-Tiering:
- Automatically moves objects between tiers
- Frequent → infrequent after 30 days
- Archive access: 90+ days
- Deep archive: 180+ days
- No retrieval charges
- Use: Unknown access patterns

S3 Glacier Instant Retrieval:
- Long-term archive (90+ days)
- Millisecond retrieval
- Use: Compliance, regulatory archives

S3 Glacier Flexible Retrieval:
- Long-term archive (90+ days)
- Retrieval: Minutes to hours (3 tiers)
- Lowest cost for archival
- Use: Long-term backup

S3 Glacier Deep Archive:
- Long-term archive (180+ days)
- Retrieval: 12 hours or more

- Lowest storage cost
- Use: Regulatory compliance, rarely accessed data

23. What are S3 lifecycle policies and how do you configure them?

Lifecycle Policies automate object transitions and deletions to optimize storage costs.

Transition Rules:
- Move objects to cheaper storage classes over time
- Example: Standard → Standard-IA after 30 days → Glacier after 90 days
- Objects must meet minimum storage period requirements

Expiration Rules:
- Delete objects after specified days
- Example: Delete incomplete multipart uploads after 7 days
- Example: Delete versions older than 90 days (with versioning)

Configuration Example (JSON):

```
{  "Rules": [    {      "Id": "Archive Rule",      "Status": "Enabled",      "Prefix": "log
s/",      "Transitions": [        {          "Days": 30,          "StorageClass": "STANDAR
D_IA"        },        {          "Days": 90,          "StorageClass": "GLACIER"        }
],      "Expiration": {        "Days": 365      }    }  ]}
```

Benefits:
- Automatic cost optimization
- Compliance with retention policies
- Reduced management overhead

24. Explain the difference between S3 and EBS.

Aspect	S3	EBS
Storage Type	Object storage	Block storage
Access	HTTP/HTTPS, REST API	Block-level, attached to EC2
Use Case	Backups, archives, static files	Databases, boot volumes
Durability	99.999999999%	99.9% (multi-AZ)

Aspect	S3	EBS
Scalability	Unlimited objects	Limited by instance attachment
Performance	Variable latency	Consistent, low latency
Pricing	Per GB stored	Per GB provisioned
Replication	Across AZs/regions	Single AZ or multi-AZ

25. What are EBS volumes and their types?

EBS (Elastic Block Store) provides persistent block-level storage for EC2 instances.

Volume Types:

General Purpose (gp3/gp2):
- Balanced performance/cost
- Up to 16,000 IOPS, 1,000 MB/s throughput (gp3)
- Use: Web servers, development, small databases

Provisioned IOPS (io1/io2):
- High-performance databases
- Up to 64,000 IOPS (io1) or 256,000 IOPS (io2)
- Highest cost, maximum durability (io2)
- Use: Databases, transaction-heavy workloads

Throughput Optimized (st1):
- High throughput sequential access
- Up to 500 MB/s
- Use: Hadoop, log processing, data warehouses

Cold HDD (sc1):
- Infrequent access, throughput
- Up to 250 MB/s
- Lowest cost
- Use: Cold storage, backups

Previous Generation (standard):
- Legacy, deprecated
- Not recommended for new deployments

Configuration:
- Size: 1GB to 64TB
- IOPS: Provisioned or auto-scaled
- Encryption: Default or AWS managed
- Snapshots: Point-in-time backups

26. What is Amazon EFS and when should you use it?

EFS (Elastic File System) is managed NFS file storage for EC2 instances.

Characteristics:
- Mountable across multiple EC2 instances
- Automatic scaling
- POSIX-compliant
- Performance modes: General (default), Max IO
- Throughput modes: Bursting (default), Provisioned

Benefits:
- Shared storage across instances
- Eliminates need for NAS appliances
- Highly available and durable
- Automatic backups available

vs. EBS:
- EBS: Single-instance block storage
- EFS: Multi-instance file storage

Use Cases:
- Shared application data
- Collaborative environments
- Content management systems
- Machine learning training data

27. How do you encrypt data in S3?

Server-Side Encryption (SSE):

SSE-S3 (Default):
- AWS manages encryption keys
- 256-bit AES encryption
- No key management overhead

- Lowest cost
- Transparent to application

SSE-KMS:
- User manages keys via AWS KMS
- Allows key rotation
- Enables audit logging (CloudTrail)
- Slightly higher cost
- Use when: Key control required, compliance

SSE-C:
- Customer provides encryption key
- AWS stores encrypted data only
- Customer manages key lifecycle
- Use when: External key management required

Client-Side Encryption:
- Encrypt before uploading to S3
- Application responsible for encryption/decryption
- Maximum control
- Use when: End-to-end encryption required

Enabling Encryption:
1. Default encryption via bucket settings
2. Per-object encryption via request headers
3. Bucket policy enforcement
4. IAM policies restricting unencrypted uploads

---

4. NETWORKING AND VPC

28. What is a VPC and its core components?

VPC (Virtual Private Cloud) is a logically isolated network within AWS.

Core Components:

Subnets:
- Segment of VPC IP address range
- Contained within single AZ
- Public (via internet gateway) or private

- CIDR block defines address range
- Reserved addresses: Network, gateway, broadcast, DNS

Internet Gateway (IGW):
- Enables internet connectivity for public subnets
- Allows resources to reach internet
- Allows inbound traffic from internet
- Attached to one VPC at a time

NAT Gateway:
- Network Address Translation for private subnets
- Enables outbound internet access
- Highly available, managed by AWS
- Requires Elastic IP address
- Placed in public subnet

Route Tables:
- Define traffic routing rules
- Associated with subnets
- Local routes created automatically
- Custom routes for internet/other resources
- Main route table for subnets without explicit association

Network ACLs:
- Stateless firewall at subnet level
- Allow and deny rules
- Processes top to bottom
- Default allows all traffic

Security Groups:
- Stateful firewall at instance level
- Allow rules only
- Per-instance or per-ENI configuration

29. How do you design a multi-tier VPC architecture?

Typical Three-Tier Design:

Web Tier (Public Subnet):
- Application load balancer

- Web servers with auto-scaling
- No direct database access
- Accessible from internet (port 80/443)
- Uses NAT Gateway for outbound access

Application Tier (Private Subnet):
- Application servers
- Cannot initiate outbound internet connections without NAT
- Security group allows traffic only from web tier
- Internal load balancer (optional)

Database Tier (Private Subnet):
- RDS Multi-AZ or Amazon Aurora
- No internet access
- Security group allows traffic only from app tier
- Backup to separate subnet/region

Additional Components:

Bastion Host (Jump Server):
- Public-facing EC2 for admin access
- SSH/RDP gateway to private instances
- Restricted security group (admin IPs only)

VPN Gateway:
- Connect to on-premises network
- Site-to-site VPN
- Managed failover

NAT Gateway Placement:
- One per AZ for high availability
- Placed in public subnet
- Associated Elastic IP for outbound traffic

VPC Endpoints:
- PrivateLink connections to AWS services
- No internet gateway required
- Lower latency, better security

30. Explain VPC Peering vs. Transit Gateway.

VPC Peering:
- Direct connection between two VPCs
- Non-transitive (A-B and B-C doesn't mean A-C)
- Requires explicit peering connections
- Routing updates needed for each peering relationship
- Simple for small number of VPCs
- Scalability issues: n VPCs require n(n-1)/2 connections

Transit Gateway (TGW):
- Central hub connecting multiple VPCs
- Transitive routing (all VPCs connected through hub)
- Simplifies multi-VPC architecture
- Scales efficiently
- Route control via Transit Gateway Route Tables
- Supports on-premises connections via Direct Connect
- Better for large environments (10+ VPCs)

Comparison Table:

Aspect	VPC Peering	Transit Gateway
Connections	One-to-one	Hub-and-spoke
Transitivity	Non-transitive	Transitive
Scalability	Limited	Large scale
Management	Complex (many peerings)	Centralized
Cost	Lower for few VPCs	Higher (per connection)

31. What is AWS Direct Connect?

AWS Direct Connect creates dedicated network connections between on-premises and AWS.

Benefits:
- Consistent network performance
- Lower latency than internet
- Dedicated bandwidth
- More secure than internet VPN
- Reduced internet bandwidth costs

Components:
- **Direct Connect Location:** AWS facility where connection is established
- **Virtual Interface:** Logical connection
- **Connection:** Physical network connection

Virtual Interface Types:

Private VIF:
- Connect to VPC via VGW
- Access to private EC2 instances
- Up to 100 VPCs per connection

Public VIF:
- Access to public AWS services (S3, DynamoDB)
- No internet traffic

Transit VIF:
- Connect to Transit Gateway
- Access multiple VPCs

Setup Process:
1. Create virtual interface
2. Download router configuration
3. Configure on-premises router
4. Create VGW or Transit Gateway
5. Create VPC connection

32. How do you configure Route 53 for DNS management?

Route 53 provides DNS and domain registration services.

Routing Policies:

Simple Routing:
- Single resource per name
- Default policy
- No special routing logic

Weighted Routing:
- Distribute traffic by percentage
- Example: 70% to version A, 30% to version B

- Useful for canary deployments
- Set weight values (0-255)

Latency-Based Routing:
- Route to resource with lowest latency
- Automatic based on source location
- Useful for global applications
- Requires health checks

Failover Routing (Active-Passive):
- Primary and secondary resources
- Route 53 monitors primary with health checks
- Automatic failover on primary failure
- Health check interval: 10 or 30 seconds

Geolocation Routing:
- Route based on geographic origin
- Country, continent, or state level
- Useful for content localization
- Compliance with data residency

Geoproximity Routing:
- Route based on geographic proximity plus bias
- More complex than geolocation
- Bias shifts traffic (positive or negative)

Multi-value Answer Routing:
- Multiple resources with health checks
- Random selection among healthy
- Similar to simple routing but with health checks

Health Checks:
- CloudWatch metrics
- Calculated health checks
- CloudWatch logs
- SNI for HTTPS health checks
- Automatic failover triggering

---

5. IAM AND SECURITY

33. What is IAM and what are its components?

IAM (Identity and Access Management) securely controls access to AWS services.

Components:

Users:
- Individual identities
- Long-term credentials (access keys, passwords)
- Not for services/applications
- Useful for: Developers, administrators, consultants

Groups:
- Collection of users
- Easier permission management
- Users inherit group permissions
- Useful for: Departments, teams

Roles:
- Assume-able identities
- Temporary security credentials
- Auto-rotated keys
- Trust policy defines who can assume
- Useful for: EC2 instances, Lambda, cross-account access

Policies:
- JSON documents defining permissions
- Attached to users, groups, or roles
- Allow and deny statements
- Resource-based or identity-based

Best Practices:
- Root account: Enable MFA, don't use for daily work
- Users: One per person, unique credentials
- Groups: Organize by role/department
- Roles: Prefer for applications/services
- Least privilege: Minimum necessary permissions
- MFA: Enforce for sensitive operations

34. What are IAM policies and policy types?

IAM Policies are JSON documents defining permissions.

Policy Structure:

```
{  "Version": "2012-10-17",  "Statement": [    {      "Effect": "Allow",      "Action": "s3:GetObject",      "Resource": "arn:aws:s3:::bucket-name/*",      "Condition": {        "IpAddress": {          "aws:SourceIp": "10.0.0.0/8"        }      }    }  ]}
```

Policy Types:

AWS Managed Policies:
- Pre-built by AWS
- Cannot modify
- Curated for common use cases
- Versioned, AWS maintains
- Examples: ReadOnlyAccess, PowerUserAccess

Customer Managed Policies:
- Created and maintained by you
- Full control
- Versioning support
- Share across users/roles
- Best for: Organization-specific permissions

Inline Policies:
- Directly attached to single user/group/role
- One-to-one relationship
- Deleted when entity deleted
- Not recommended for production

Resource-Based Policies:
- Attached to resources (S3, SQS, SNS)
- Define who can access resource
- Useful for: Cross-account access, public access
- Example: S3 bucket policy

35. Explain the principle of least privilege.

Principle of Least Privilege grants minimum necessary permissions for a task.

Application:
- Users/roles: Only required permissions
- Services: Specific resources, not wildcards
- Conditions: IP restrictions, time-based access
- Regular audits: Remove unnecessary permissions

Example - Least Privilege for EC2 Service Role:

```
{  "Version": "2012-10-17",  "Statement": [    {      "Effect": "Allow",      "Action": ["s3:GetObject"],      "Resource": "arn:aws:s3:::app-bucket/data/*"    },    {      "Effect": "Allow",      "Action": ["logs:CreateLogStream", "logs:PutLogEvents"],      "Resource": "arn:aws:logs:us-east-1:123456789:log-group:/aws/ec2/app:*"    }  ]}
```

Benefits:
- Reduced blast radius on compromise
- Improved security posture
- Compliance with regulations
- Audit trail clarity

36. What is MFA and how do you enable it in AWS?

MFA (Multi-Factor Authentication) requires multiple verification factors.

MFA Factors:
1. Something you know (password/PIN)
2. Something you have (MFA device)
3. Something you are (biometric)

AWS MFA Support:

Virtual MFA Device:
- Google Authenticator, Authy, Microsoft Authenticator
- Time-based one-time passwords (TOTP)
- Free, convenient
- Supported per user

U2F Security Key:
- Physical device (YubiKey, etc.)
- Highest security
- Works across multiple services
- Browser-based authentication

Hardware MFA Device:
- AWS-provided hardware tokens
- Long-term credentials
- Suitable for sensitive roles
- VPN and console access

Enabling MFA for Root Account:
1. Sign in as root user
2. Account settings → Security Credentials
3. MFA devices → Activate MFA
4. Choose device type
5. Scan QR code/enter serial
6. Enter consecutive codes to verify

Benefits:
- Additional security layer
- Protects against password compromise
- Required for sensitive operations (delete, modify)
- Compliance requirement (many organizations)

37. What is AWS KMS and how does encryption work?

AWS KMS (Key Management Service) manages cryptographic keys for encryption.

Key Types:

Customer Master Keys (CMK):
- AWS managed (default for services)
- Customer managed (customer creates/controls)
- Can be imported from external sources
- Stored in HSM (Hardware Security Module)

Data Keys:
- Generated from CMK
- Used to encrypt data directly
- Sent in plaintext with encrypted data
- Customer never sees unencrypted CMK

Envelope Encryption Process:
1. Request data key from KMS: `GenerateDataKey()`
2. Receive plaintext and ciphertext versions
3. Use plaintext key to encrypt data locally
4. Discard plaintext key after encryption
5. Store ciphertext key alongside encrypted data
6. To decrypt: Use ciphertext key with KMS to get plaintext
7. Use plaintext key to decrypt data

Benefits:
- Centralized key management
- Encryption at rest for S3, RDS, EBS, DynamoDB
- Audit logging via CloudTrail
- Key rotation policies
- Compliance support (HSM in AWS CloudHSM)

38. What is AWS Secrets Manager?

Secrets Manager securely stores and rotates secrets (passwords, API keys, database credentials).

Features:
- Automatic secret rotation
- Encryption at rest (KMS)
- Audit logging (CloudTrail)
- Permission control (IAM policies)
- Versioning of secrets
- JSON or plaintext secrets

Use Cases:
- Database credentials
- API keys

- SSH keys (Secrets Manager + Systems Manager)
- Third-party integration credentials

Rotation:
- Built-in rotation for RDS, DocumentDB, Aurora
- Custom Lambda for other services
- Automatic or manual
- Credentials remain valid during rotation

vs. Systems Manager Parameter Store:
- Secrets Manager: Passwords, keys, automatic rotation
- Parameter Store: Configuration, less sophisticated

39. What is AWS WAF and how does it protect applications?

AWS WAF (Web Application Firewall) protects web applications from common web exploits.

How It Works:
- Monitors HTTP/HTTPS requests
- Applies rules to allow/block traffic
- Integrates with CloudFront, ALB, API Gateway
- Real-time metrics and logging

Rule Types:

Managed Rules:
- AWS-provided rule sets
- Common Vulnerabilities (OWASP top 10)
- No management overhead

Custom Rules:
- Define specific conditions
- IP addresses, headers, URIs, body content
- Rate limiting (DDoS protection)
- Geo-blocking

Protection Against:
- SQL Injection
- Cross-Site Scripting (XSS)
- DDoS attacks

- Malicious bots
- Brute force attacks

Configuration:
1. Create WAF ACL
2. Add rules (managed or custom)
3. Assign to CloudFront, ALB, or API Gateway
4. Monitor via CloudWatch

40. What is AWS CloudTrail?

CloudTrail logs all API calls for auditing and compliance.

Capabilities:
- Logs API calls from AWS Management Console, CLI, SDK
- Records: Who, what, when, where, result
- Stores logs in S3
- Multi-AZ trails for all regions
- Integration with CloudWatch Logs
- SNS notifications on API calls

Event Types:
- Management events (API calls)
- Data events (specific resources)
- Insight events (unusual activity)

Use Cases:
- Compliance auditing
- Security analysis
- Resource change tracking
- Troubleshooting
- Detecting unauthorized access

Best Practices:
- Enable on all accounts
- Multi-region trails
- Log file integrity validation
- CloudTrail Lake for long-term analysis
- S3 lifecycle policies for cost optimization

6. DATABASE SERVICES

41. What is Amazon RDS and which database engines are supported?

RDS (Relational Database Service) is a managed relational database service.

Supported Engines:
- MySQL
- PostgreSQL
- MariaDB
- Oracle Database
- Microsoft SQL Server
- Amazon Aurora (MySQL/PostgreSQL compatible)

Benefits:
- Automated backups (automated snapshots)
- Automatic failover (Multi-AZ)
- Software patching
- Hardware scaling
- Read replicas for read scaling
- Encryption at rest and in transit

Multi-AZ Deployment:
- Synchronous replication to standby
- Automatic failover (typically 1-2 minutes)
- Same database endpoint
- Higher availability, higher cost

Read Replicas:
- Asynchronous replication
- Read-heavy workload scaling
- Cross-region replicas possible
- Can be promoted to standalone DB

42. What is Amazon Aurora and how does it differ from RDS?

Aurora is AWS's next-generation managed relational database.

Differences:

Aspect	RDS	Aurora
Engine	Multiple (MySQL, PostgreSQL, etc.)	Aurora MySQL/PostgreSQL
Performance	Standard	3x MySQL, 3x PostgreSQL
Availability	Multi-AZ failover	Auto-healing, 3-way replication
Scaling	Vertical (resize instances)	Auto-scaling read replicas
Backup	Snapshots	Continuous incremental
Cost	Lower per instance	Higher per node, efficient
Data Replication	Single-master	Shared storage

Aurora Key Features:
- Aurora Global Database: Cross-region read replicas
- Serverless Aurora: Auto-scaling based on demand
- Compatible with MySQL/PostgreSQL
- Automatic scaling up to 15 read replicas
- Intelligent routing
- Parallel query execution

43. What is Amazon DynamoDB and when should you use it?

DynamoDB is a fully managed NoSQL database service.

Characteristics:
- Key-value and document store
- Millisecond latency
- Fully managed (no servers)
- Automatic scaling
- High availability (3-way replication)
- On-demand or provisioned billing

When to Use DynamoDB:
- High-concurrency reads/writes
- Predictable scaling needs
- Flexible schema
- Session data, real-time analytics
- IoT applications, gaming leaderboards
- Mobile backends

When NOT to Use:
- Complex joins (use RDS)
- Complex queries (use RDS)
- Transactional integrity across tables (use RDS)
- SQL requirements

DynamoDB Consistency Models:

Eventual Consistency (Default):
- Lower latency
- Lower cost
- Data eventually consistent (within milliseconds)
- Suitable for most use cases

Strong Consistency:
- Always fresh data
- Higher latency/cost
- Use when: Immediate consistency required

44. What are indexes in DynamoDB?

Indexes enable querying on non-key attributes.

Local Secondary Indexes (LSI):
- Same partition key as table
- Different sort key
- Maximum 10GB per partition key value
- Strong consistency option
- Created at table creation only

Global Secondary Indexes (GSI):
- Any partition and sort key
- No size limits per partition
- Eventually consistent only
- Can be added/removed anytime
- Separate throughput provisioning

Use Cases:
- Query by different attributes
- Different sort orders

- Performance optimization
- Access patterns

45. What is Amazon Redshift?

Redshift is a data warehouse service for large-scale analytics.

Characteristics:
- Columnar storage (compressed)
- Massive parallel processing (MPP)
- Suitable for gigabytes to petabytes
- SQL-based queries
- Much faster than traditional warehouses
- Cost-effective for infrequent queries (Spectrum)

Architecture:
- Leader node (coordination)
- Compute nodes (data storage/processing)
- Distributed queries

Use Cases:
- Business intelligence
- Data analysis
- Historical data analysis
- Time-series data analysis

vs. Other Services:
- vs. RDS: Large-scale analytics vs. transactional
- vs. Athena: Persistent data warehouse vs. ad-hoc queries
- vs. Elasticsearch: BI vs. full-text search

46. What is Amazon ElastiCache?

ElastiCache is an in-memory data store service.

Use Cases:
- Session caching
- Database query caching
- Real-time leaderboards
- Real-time analytics

Engines:
- **Memcached:** Simple, distributed memory caching
- **Redis:** Advanced data structures, persistence

Deployment:
- Single-node (development)
- Multi-node cluster (production)
- Automatic failover
- Encryption support

Best Practices:
- Cache frequently accessed data
- Appropriate TTL values
- Monitor cache hit rates
- Backup strategies for persistent data

---

7. ADVANCED ARCHITECTURE AND DESIGN

47. How do you design a highly available application on AWS?

Strategies for High Availability:

1. Multi-AZ Deployment:
- Distribute resources across at least 2 AZs
- Automatically handled: RDS Multi-AZ, Aurora, ALB
- Application-level: EC2 in ASG across AZs

2. Load Balancing:
- ALB/NLB distributes traffic
- Health checks detect unhealthy instances
- Automatic rerouting on failure

3. Auto Scaling:
- Maintain desired capacity
- Replace unhealthy instances
- Scale based on demand

4. Database Redundancy:
- RDS Multi-AZ (synchronous standby)

- Read replicas for read scaling
- Aurora (3-way replication across AZs)

5. Data Backup:
- Automated snapshots
- Cross-region backup copies
- Point-in-time recovery

6. Monitoring & Alerts:
- CloudWatch metrics
- Custom alarms
- SNS notifications
- Automatic recovery triggers

Example Architecture:
- Application Load Balancer across 2 AZs
- EC2 instances in ASG across 2 AZs
- RDS Multi-AZ
- S3 with cross-region replication

48. What is a disaster recovery strategy?

Disaster Recovery (DR) prepares for system failures/outages.

Recovery Metrics:
- **RTO (Recovery Time Objective):** Max acceptable downtime
- **RPO (Recovery Point Objective):** Max acceptable data loss

DR Strategies (ordered by cost/complexity):

1. Backup & Restore:
- Automated backups to S3
- Restore on failure
- RTO: Hours to days
- RPO: Hours
- Cost: Lowest
- Use: Non-critical applications

2. Pilot Light:
- Minimal resources running in DR region
- Replicate data asynchronously

- Scale up on failover
- RTO: 15-30 minutes
- RPO: 15-30 minutes
- Cost: Low

3. Warm Standby:
- Fully functional but scaled-down DR environment
- Continuous data replication
- Automatic failover (Route 53)
- RTO: 5-10 minutes
- RPO: 1-5 minutes
- Cost: Moderate

4. Multi-Site Active/Active:
- Full-scale systems in multiple regions
- Both handling production traffic
- RTO: Near zero
- RPO: Near zero
- Cost: Highest
- Use: Mission-critical applications

49. How do you implement auto-scaling for an application?

Auto Scaling Components:

1. Launch Template:

```
- AMI specification
- Instance type
- Security groups
- Key pair
- User data scripts
- Monitoring
```

2. Auto Scaling Group (ASG):

```
- Minimum capacity (always running)
- Maximum capacity (never exceed)
- Desired capacity (target)
- Subnets (across AZs)
- Load balancer attachment
- Health check configuration
```

3. Scaling Policies:

Target Tracking:
- Maintain metric at target value
- Example: 70% CPU utilization
- Automatically adjusts scaling
- Simplest to configure
- Most popular approach

Step Scaling:
- Multiple thresholds with different actions
- Example: >80% CPU: +2, >90% CPU: +4
- More granular control
- Better for bursty traffic

Simple Scaling:
- Single threshold
- Single action
- Less flexible

4. Metrics:
- CPU utilization (default)
- Network metrics (in/out)
- Custom CloudWatch metrics
- Target group request count

5. Lifecycle Hooks:
- Delay actions before termination
- Run scripts (deregistration, cleanup)
- Connection draining

Example Configuration:

```
Launch Template:
- Ubuntu 20.04 AMI
- t3.medium instance
- Auto-scaling SG
- User data: Install app

ASG:
- Min: 1, Max: 5, Desired: 2
- Subnets: 2 AZs
- Attach ALB
- Health check: ALB (300s grace period)

Scaling Policy:
- Target tracking
- Average CPU: 70%
- Scale-up cooldown: 300s
- Scale-down cooldown: 300s
```

50. What is AWS CloudFormation?

CloudFormation is Infrastructure as Code service for AWS resources.

Benefits:
- Declarative infrastructure definition
- Version control
- Repeatable deployments
- Stack rollback on failure
- Change sets preview changes
- Drift detection

Template Structure:

```
{  "AWSTemplateFormatVersion": "2010-09-09",  "Description": "Web applicat
ion template",  "Parameters": {    "InstanceType": {      "Type": "String",      "Def
ault": "t3.micro"    }  },  "Resources": {    "EC2Instance": {      "Type": "AWS::EC
2::Instance",      "Properties": {        "ImageId": "ami-12345678",        "Instance
Type": { "Ref": "InstanceType" }      }    }  },  "Outputs": {    "InstanceId": {
"Value": { "Ref": "EC2Instance" }    }  }}
```

Stack Operations:
- Create: Deploy new resources
- Update: Modify existing resources
- Delete: Remove resources and stack
- Drift Detection: Compare actual vs. expected

---

8. SERVERLESS COMPUTING

51. What are Lambda cold starts and how do you minimize them?

Cold Start: Delay when Lambda environment initializes before code execution.

Causes:
- Function not invoked for some time
- Auto-scaling new instances
- Large deployment package
- Heavy initialization code

Duration:
- Provisioned runtime: 1-10ms
- Interpreted languages (Node.js, Python): 100-300ms
- Java: 500-1000ms
- .NET: 300-500ms

Mitigation Strategies:

1. Provisioned Concurrency:
- Pre-warmed function instances
- Eliminates cold starts
- Higher cost
- Use for critical APIs

2. Reserved Concurrency:
- Reserved capacity
- Prevents throttling
- Doesn't eliminate cold start
- Cost-effective than provisioned

3. Lambda Layers:
- Separate dependencies from code

- Reduces package size
- Faster download/initialization

4. Lightweight Runtimes:
- Node.js/Python: Faster cold start than Java
- Go: Compiled, very fast
- Custom runtime: Minimal overhead

5. Initialization Optimization:
- Move initialization outside handler
- Use connection pools
- Lazy-load libraries
- Avoid large data structures

6. Keep Functions Warm:
- Periodic invocations (CloudWatch Events)
- Scheduled trigger every 5 minutes
- Prevents auto-termination

52. What is AWS API Gateway?

API Gateway manages REST and WebSocket APIs.

Features:
- RESTful and WebSocket APIs
- Request/response transformation
- Authorization (IAM, API keys, Cognito)
- Rate limiting and throttling
- CORS support
- Request validation
- Caching
- CloudWatch integration
- API versioning

Integration Types:
- Lambda (serverless)
- HTTP endpoint
- AWS service
- VPC link
- Mock (testing)

Authorization Methods:
- AWS IAM
- API keys
- Cognito user pools
- Custom authorizers (Lambda)
- OIDC
- OAuth 2.0

Stages:
- dev, test, prod
- Independent configurations
- Stage variables

Example Use Case:

```
Client → API Gateway (REST)
       ↓
   Lambda Function
       ↓
   DynamoDB
```

53. What is AWS SAM (Serverless Application Model)?

SAM simplifies serverless application development.

Features:
- Simplified CloudFormation syntax
- Local testing and debugging
- Built-in best practices
- CLI for deployment

Template Example:

```
AWSTemplateFormatVersion: '2010-09-09'Transform: AWS::Serverless-2016-10-31Resources:  MyApiFunction:    Type: AWS::Serverless::Function    Properties:      Handler: index.handler      Runtime: python3.9      Events:        ApiEvent:          Type: Api          Properties:            Path: /hello            Method: GET      Environm
```

```
ent:       Variables:         TABLE_NAME: MyTable      Policies:        - DynamoDB
CrudPolicy:           TableName: MyTable
```

Advantages:
- Less verbose than CloudFormation
- Quick prototyping
- Built-in Lambda best practices
- Easy local testing

54. What is AWS Amplify?

Amplify accelerates web and mobile app development.

Features:
- Frontend hosting
- Backend API creation
- Database/storage
- Authentication
- Real-time data sync
- CI/CD deployment

Components:
- Amplify CLI: Development
- Amplify Console: Hosting and CI/CD
- Amplify Libraries: Frontend integration
- Amplify Studio: Visual development

Use Cases:
- Full-stack web application development
- Mobile app backend
- Rapid prototyping
- Increasing development velocity

---

9. CONTAINER ORCHESTRATION

55. What is Amazon ECS?

ECS (Elastic Container Service) manages Docker containers at scale.

Components:
- **Container Definition:** Docker image, CPU, memory, port mappings
- **Task Definition:** Container blueprint (can run multiple containers)
- **Service:** Long-running task management with load balancing
- **Cluster:** Group of resources running tasks

Launch Types:

EC2 Launch Type:
- Manages EC2 instances
- Full control
- Cost-effective for continuous workloads
- More operational overhead

Fargate Launch Type:
- Serverless containers
- No EC2 management
- Pay per container
- Easier operations
- Good for: Variable workloads, startup environments

Task Placement Strategies:
- Spread (across instances)
- Binpack (minimize instances)
- Random

Deployment Strategies:
- Rolling update
- Blue/Green deployment
- Canary deployment

56. What is Amazon EKS?

EKS (Elastic Kubernetes Service) is managed Kubernetes.

Benefits:
- Managed Kubernetes control plane
- VPC networking
- IAM integration
- AWS resource compatibility

- Automatic patching
- High availability

vs. ECS:
- ECS: AWS-native, simpler, less learning curve
- EKS: Kubernetes, portable, larger ecosystem

Use Cases:
- Kubernetes migration from on-premises
- Multi-cloud strategy (Kubernetes on multiple clouds)
- Advanced container orchestration needs
- Team familiar with Kubernetes

57. What is AWS Fargate?

Fargate is serverless container compute.

Characteristics:
- No EC2 instance management
- Pay for container resources used
- Works with ECS and EKS
- Automatic scaling
- VPC networking

Pricing:
- Per vCPU and memory hour
- No idle charges if not running
- Supports 0.25 to 4 vCPU

Use Cases:
- Microservices
- Batch processing
- Scheduled tasks
- Variable workload websites

---

10. MONITORING, LOGGING, AND OPTIMIZATION

58. What is Amazon CloudWatch?

CloudWatch monitors AWS resources and applications.

Components:

Metrics:
- Time-series data from resources
- 1-minute to 1-hour granularity
- Custom metrics available
- 15 months retention

Logs:
- Application and system logs
- Centralized log storage
- Log groups and streams
- Retention policies
- CloudWatch Logs Insights (SQL-like queries)

Alarms:
- Threshold-based triggers
- Send notifications (SNS, EC2 action)
- Auto Scaling triggers
- State: OK, ALARM, INSUFFICIENT_DATA

Events (EventBridge):
- Real-time event processing
- Route to targets (Lambda, SNS, Kinesis)
- Schedule-based (cron)
- Resource-based (EC2 state change)

Dashboards:
- Visualize metrics
- Multiple metric types
- Real-time updates

59. What is AWS X-Ray?

X-Ray traces requests through distributed systems.

Capabilities:
- Request tracing
- Identify bottlenecks
- Service map visualization

- Performance analysis
- Error identification

Integration:
- Lambda, ECS, Elastic Beanstalk
- HTTP requests
- Database queries
- API calls

Benefits:
- End-to-end visibility
- Performance optimization
- Troubleshooting
- Application improvements

60. What is AWS CloudTrail?

CloudTrail logs AWS API calls for auditing.

Features:
- API call logging
- 90 days default retention
- S3 storage for long-term
- Encrypted logs
- CloudTrail Insights (unusual activity)
- Multi-region trails

Use Cases:
- Compliance auditing
- Security analysis
- Resource change tracking
- Troubleshooting

---

11. DISASTER RECOVERY AND BUSINESS CONTINUITY

61. What is AWS Backup?

Backup centralizes backup management.

Features:
- Centralized backup policies
- Cross-region backup copies
- On-demand and scheduled backups
- Compliance verification
- Backup compliance checks

Supported Services:
- RDS, DynamoDB, EBS, S3
- EC2, Neptune, DocumentDB
- EFS, FSx

62. What is Amazon S3 Cross-Region Replication?

CRR automatically replicates S3 objects to another region.

Requirements:
- Versioning enabled on both buckets
- IAM role for replication
- Source and destination regions

Characteristics:
- Asynchronous replication
- Separate storage costs
- Can replicate to different storage classes
- Delete markers can be replicated
- Good for: Compliance, disaster recovery

---

12. SCENARIO-BASED QUESTIONS

63. Design a system to handle a sudden spike in traffic.

Solution:

Immediate (0-5 minutes):
1. Auto Scaling Group increases instances based on CloudWatch metrics
2. ALB distributes traffic across instances
3. ElastiCache becomes active for frequently accessed data
4. CloudFront caches static content at edge

Short-term (5-30 minutes):
5. Monitor CloudWatch metrics
6. Add read replicas to RDS if database under load
7. Scale up application servers (vertical if needed)
8. Implement rate limiting via API Gateway

Long-term:
9. Analyze traffic patterns
10. Optimize code/database queries
11. Implement caching strategies
12. Review cost implications

Architecture Components:
- Route 53 with latency-based routing
- CloudFront for static content
- ALB for request distribution
- EC2 with ASG (min 2, max 20 instances)
- RDS with read replicas
- ElastiCache for caching
- S3 for static assets

64. A database is experiencing performance issues. How do you troubleshoot?

Investigation Steps:

1. Identify Bottleneck:
- CPU utilization (high query overhead)
- Memory (insufficient buffer pool)
- Disk I/O (slow reads/writes)
- Network (insufficient bandwidth)
- Connections (connection pool exhaustion)

Tools:
- CloudWatch metrics: CPU, Memory, Storage, Connections
- RDS Performance Insights
- Enhanced Monitoring
- Query execution plans

2. Common Issues & Solutions:

High CPU:
- Solution: Optimize queries, indexes, or scale up instance
- Tools: Slow query log, query optimizer

Memory Pressure:
- Solution: Increase instance size, tune buffer pool
- Tools: Memory utilization metrics

Disk I/O:
- Solution: Use SSD-backed instances, optimize queries
- Tools: I/O metrics, query plans

Slow Queries:
- Solution: Add indexes, optimize queries, partitioning
- Tools: Query analyzer, slow log

Connection Pool Exhaustion:
- Solution: Use RDS Proxy for connection pooling
- Benefits: Reduce connections, improve performance

3. RDS Proxy Solution:
- Connection pooling between application and database
- Transparent to application
- Reduces database load
- Improves connection efficiency

65. How do you migrate a 10TB database to AWS with minimal downtime?

Migration Strategy:

Phase 1: Preparation
1. Assess source database
2. Use AWS Database Migration Service (DMS)
3. Create DMS replication instance
4. Create target RDS instance (appropriately sized)

Phase 2: Full Load
1. Run full data load from source
2. Source database remains online

3. Verify data completeness
4. Measure initial load time

Phase 3: Change Data Capture (CDC)
1. Enable CDC on DMS
2. Continuous replication of changes
3. Minimal latency (seconds behind source)
4. Allows time for testing

Phase 4: Testing
1. Validate data integrity
2. Test application with target database
3. Run query validation
4. Performance testing

Phase 5: Cutover
1. Stop application (brief maintenance window: 15-30 minutes)
2. Final sync of CDC changes
3. Enable foreign keys, triggers
4. Update connection strings
5. Restart application
6. Monitor closely

Tools:
- AWS DMS
- AWS Schema Conversion Tool (SCT)
- AWS DataSync (for on-premises data)

Downtime: 15-30 minutes (cutover window)

---

13. COST OPTIMIZATION

66. What are strategies for reducing AWS costs?

Compute Optimization:

1. **Right-sizing:**
    - Use CloudWatch metrics to analyze utilization
    - Downsize over-provisioned instances

- Use AWS Compute Optimizer for recommendations
- Savings: 20-30%

2. **Instance Purchasing Options:**
   - Reserved Instances for baseline: 40-70% discount
   - Spot Instances for flexible workloads: 70-90% discount
   - On-Demand for variable workloads
   - Mix for optimal cost
3. **Stop/Delete Unused Resources:**
   - Identify stopped instances (still charged for EBS)
   - Delete unattached EBS volumes
   - Remove unused Elastic IPs
   - Cleanup old snapshots

Storage Optimization:

1. **S3 Lifecycle Policies:**
   - Transition to cheaper storage: Standard → Standard-IA → Glacier
   - Savings: 50-90% for archived data
2. **Intelligent-Tiering:**
   - Automatic tiering based on access patterns
   - No management overhead
3. **Delete Old Snapshots:**
   - Snapshots have ongoing costs
   - Delete after data is stored elsewhere

Database Optimization:

1. **Aurora:** Better cost-per-performance than RDS
2. **RDS Read Replicas:** Scale reads without scaling primary
3. **DynamoDB On-Demand:** For variable workloads

Data Transfer Optimization:

1. **Minimize Inter-AZ Traffic:**
    - Use resources in same AZ when possible
    - Data transfer between AZs is charged
2. **CloudFront for Distribution:**
    - Reduces outbound data transfer
    - Caches content at edge locations
3. **VPC Endpoints:**
    - Avoid data transfer charges for AWS services

Monitoring & Alerts:

1. **Cost Explorer:**
    - Analyze spending by service
    - Identify trends
    - Set budgets
2. **AWS Budgets:**
    - Set spending limits
    - Receive alerts on overage
    - Create custom filters
3. **Trusted Advisor:**
    - Identifies cost optimization opportunities
    - Underutilized resources
    - Reserve Instance opportunities

67. How do you estimate AWS costs?

Cost Estimation Tools:

1. **AWS Simple Calculator:**

    - Add services and configuration
    - Instant cost estimates
    - Scenario comparison
2. **AWS Pricing Calculator:**
    - More detailed estimates
    - Save configurations
    - Multiple scenarios
3. **Cost Explorer:**
    - Historical actual costs
    - Forecasts based on usage

Cost Components by Service:

EC2:
- Instance type and size
- Running hours
- Data transfer
- Elastic IPs
- Load Balancer

S3:
- Storage amount
- Request volume
- Data transfer
- Cross-region replication

RDS:
- Instance type
- Running hours
- Storage
- I/O operations

Estimation Process:
1. Identify required services
2. Estimate usage volumes

3. Calculate per-service costs
4. Add overhead (monitoring, logging, data transfer)
5. Apply discounts (RI, Savings Plans)
6. Factor in growth

---

14. DEVOPS AND INFRASTRUCTURE AS CODE

68. What is AWS CodePipeline?

CodePipeline automates application release processes.

Stages:

Source:
- Code repository (CodeCommit, GitHub, S3)
- Triggers on code changes
- Branch-based workflows

Build:
- CodeBuild compiles and tests code
- Docker image creation
- Security scanning

Deploy:
- CodeDeploy to EC2/on-premises
- CloudFormation for infrastructure
- Elastic Beanstalk for applications
- ECS/EKS for containers

Stages Can Include:
- Manual approval
- Lambda functions
- Third-party integrations
- Notifications

69. What is AWS CodeBuild?

CodeBuild compiles source code and runs tests.

Features:
- Managed build service

- Scales automatically
- Pre-built environments
- Custom Docker images
- Secrets management
- CloudWatch integration

Use Cases:
- Compile applications
- Run unit tests
- Build Docker images
- Security scanning

70. What is AWS CodeDeploy?

CodeDeploy automates application deployment.

Deployment Options:
- In-place deployment (replace on existing instances)
- Blue/Green deployment (maintain parallel environments)

Supported Targets:
- EC2 instances
- On-premises servers
- Lambda
- ECS (via CodePipeline)

Deployment Strategies:
- OneAtATime
- HalfAtATime
- AllAtOnce
- Custom percentage