AWS Interview Preparation Guide

Cloud Computing Basics

Three Types of Cloud Services

Computing Services:

- **EC2** - Virtual servers in the cloud
- **Lambda** - Serverless compute (run code without managing servers)
- **Elastic Beanstalk** - Platform for deploying web applications

Storage Services:

- **S3** - Object storage (files, images, videos)
- **EBS** - Block storage for EC2 instances (like hard drives)
- **EFS** - File system storage (shared across multiple instances)

Networking Services:

- **VPC** - Virtual Private Cloud (your own network in AWS)
- **CloudFront** - Content delivery network (CDN)
- **Route53** - DNS service

Regions vs Availability Zones

- **Region** - Geographic area (like us-east-1, eu-west-1)
- **Availability Zone (AZ)** - Data centers within a region
- Each region has multiple AZs for high availability
- Deploy across multiple AZs to avoid single points of failure

AWS Shared Responsibility Model

- **AWS Responsibility** - Physical security, infrastructure, host OS
- **Customer Responsibility** - Guest OS, applications, data, network configuration
- Think of it as: AWS secures "OF" the cloud, you secure "IN" the cloud

IaaS, PaaS, SaaS

- **IaaS** - Infrastructure as a Service (EC2) - You manage OS and above

- **PaaS** - Platform as a Service (Elastic Beanstalk) - You manage applications only
- **SaaS** - Software as a Service (Gmail, Office 365) - Everything managed for you

Identity and Access Management (IAM)

Core Components

- **Users** - Individual people or services
- **Groups** - Collection of users (like "Developers", "Admins")
- **Roles** - Temporary permissions for AWS services or cross-account access
- **Policies** - JSON documents that define permissions

IAM Policy Structure

```json
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "arn:aws:iam::123456789012:user/Bob",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::mybucket/*",
      "Condition": {
        "StringEquals": {
          "aws:RequestedRegion": "us-east-1"
        }
      }
    }
  ]
}
```

Policy Evaluation Logic

1. **Explicit Deny** - Always wins (highest priority)
2. **Allow** - If explicitly allowed
3. **Implicit Deny** - Default (everything denied unless explicitly allowed)

Key IAM Concepts

- **Authentication** - Who you are (username/password)

- **Authorization** - What you can do (permissions)
- **Cross-account access** - Use roles to access resources in different AWS accounts
- **AWS STS** - Security Token Service provides temporary credentials
- **Least privilege** - Give minimum permissions needed

Compute Services

Amazon EC2

Instance Types:

- **General Purpose** (t3, m5) - Balanced CPU, memory, networking
- **Compute Optimized** (c5) - High-performance processors
- **Memory Optimized** (r5, x1) - Fast performance for memory-intensive workloads
- **Storage Optimized** (i3, d2) - High disk I/O
- **Accelerated Computing** (p3, g4) - GPU instances

Auto Scaling:

- **Auto Scaling Group (ASG)** - Automatically add/remove instances
- **Scaling Policies** - Scale based on metrics (CPU, memory, custom)
- **Target Tracking** - Maintain specific metric value
- **Simple Scaling** - Add/remove fixed number of instances

Load Balancers:

- **Application Load Balancer (ALB)** - HTTP/HTTPS traffic (Layer 7)
- **Network Load Balancer (NLB)** - TCP/UDP traffic (Layer 4), ultra-low latency
- **Classic Load Balancer (CLB)** - Legacy, supports both Layer 4 and 7

Security Groups vs NACLs:

- **Security Groups** - Firewall at instance level, stateful, allow rules only
- **NACLs** - Firewall at subnet level, stateless, allow and deny rules

EC2 Pricing:

- **On-Demand** - Pay by hour/second, no commitment
- **Reserved** - 1-3 year commitment, up to 75% discount
- **Spot** - Unused capacity, up to 90% discount, can be terminated

AWS Lambda

Key Concepts:

- **Serverless** - No server management, automatic scaling
- **Event-driven** - Triggered by events (S3 uploads, API calls, schedules)
- **Pay per execution** - Only pay when code runs

Function Structure:

```python
def lambda_handler(event, context):
    # Your code here
    return {
        'statusCode': 200,
        'body': json.dumps('Hello World!')
    }
```

Cold Starts vs Warm Starts:

- **Cold Start** - Function not used recently, takes longer to start
- **Warm Start** - Function container reused, faster execution
- **Optimization** - Keep functions small, reuse connections

Lambda Limitations:

- 15-minute maximum execution time
- 10GB memory maximum
- 6MB request/response payload
- 512MB /tmp directory

Common Integrations:

- **API Gateway** - Create REST APIs
- **S3** - Process file uploads
- **DynamoDB** - Process database changes
- **SQS** - Process messages from queues

Error Handling:

- **Retries** - Automatic retries for failed executions
- **Dead Letter Queue** - Send failed messages for investigation
- **Lambda Layers** - Share code and dependencies across functions

Storage Services

Amazon S3

Storage Classes:

- **Standard** - Frequently accessed data
- **Standard-IA** - Infrequently accessed, lower cost
- **One Zone-IA** - Single AZ, even lower cost
- **Glacier** - Long-term archive, minutes to hours retrieval
- **Glacier Deep Archive** - Lowest cost, 12+ hour retrieval

Key Features:

- **Versioning** - Keep multiple versions of objects
- **MFA Delete** - Require MFA to delete objects
- **Cross-Region Replication** - Automatically copy objects to different regions
- **Lifecycle Policies** - Automatically transition objects between storage classes

S3 Security:

- **Bucket Policies** - JSON policies for bucket-level permissions
- **ACLs** - Access Control Lists (legacy, use bucket policies instead)
- **Encryption** - Server-side (SSE-S3, SSE-KMS, SSE-C) and client-side

Database Services

RDS vs DynamoDB

RDS (Relational):

- Traditional databases (MySQL, PostgreSQL, Oracle)
- ACID transactions
- Complex queries with JOINs
- Vertical scaling

DynamoDB (NoSQL):

- Key-value and document database
- Millisecond latency at scale
- Automatic scaling
- No complex queries

RDS Features:

- **Multi-AZ** - Synchronous replication for high availability
- **Read Replicas** - Asynchronous replication for read scaling

DynamoDB Concepts:

- **Partition Key** - Primary key for data distribution
- **Sort Key** - Optional, creates composite primary key
- **GSI** - Global Secondary Index (different partition/sort key)
- **LSI** - Local Secondary Index (same partition key, different sort key)

Networking and Content Delivery

Amazon VPC

Core Components:

- **Subnets** - IP address ranges within VPC
- **Route Tables** - Control traffic routing
- **Internet Gateway** - Connect to internet
- **NAT Gateway** - Allow private subnets to access internet

Subnet Types:

- **Public Subnet** - Has route to Internet Gateway
- **Private Subnet** - No direct internet access

VPC Connectivity:

- **VPC Peering** - Connect two VPCs
- **Transit Gateway** - Connect multiple VPCs and on-premises
- **VPN** - Encrypted connection over internet

- **Direct Connect** - Dedicated network connection

CloudFront and Route53

CloudFront:

- **Edge Locations** - Cache content closer to users
- **Geo-targeting** - Serve different content by location
- **Geo-restriction** - Block content in specific countries

Route53 Routing Policies:

- **Simple** - Single resource
- **Weighted** - Distribute traffic by percentages
- **Latency** - Route to lowest latency endpoint
- **Failover** - Primary/secondary setup
- **Health Checks** - Monitor endpoint health

DevOps and CI/CD Services

AWS DevOps Tools

CodeCommit - Git repositories in AWS **CodeBuild** - Compile and test code **CodeDeploy** - Deploy to EC2, Lambda, on-premises **CodePipeline** - End-to-end CI/CD workflows

Blue/Green Deployment:

- **Blue Environment** - Current production
- **Green Environment** - New version
- Switch traffic after testing green environment

Infrastructure as Code

CloudFormation:

- JSON/YAML templates
- Declarative infrastructure
- Stack management (create, update, delete)

AWS CDK - Define infrastructure using programming languages

Container Services

ECS vs EKS:

- **ECS** - AWS managed container orchestration
- **EKS** - Managed Kubernetes service

Launch Types:

- **Fargate** - Serverless containers
- **EC2** - Containers on EC2 instances you manage

ECR - Elastic Container Registry for storing container images

Monitoring and Security

CloudWatch

- **Metrics** - Performance data from AWS services
- **Alarms** - Notifications based on metrics
- **Dashboards** - Visual monitoring
- **Logs** - Centralized logging
- **X-Ray** - Distributed tracing for troubleshooting

Security Services

KMS - Key Management Service for encryption **Secrets Manager** - Store and rotate secrets **GuardDuty** - Threat detection using ML **Config** - Monitor resource compliance **CloudTrail** - Log all API calls for auditing

Advanced Topics

High Availability and Disaster Recovery

Multi-AZ - Deploy across multiple availability zones **Cross-Region** - Replicate to different regions **RTO** - Recovery Time Objective (how long to recover) **RPO** - Recovery Point Objective (how much data loss acceptable)

Cost Optimization

Cost Explorer - Analyze spending patterns **Budgets** - Set spending alerts **Reserved Instances** - Commit to save money **Savings Plans** - Flexible pricing model **Cost Allocation Tags** - Track costs by project/department

Data Transfer and Migration

Snowball Family:

- **Snowball** - 80TB device for data transfer
- **Snowball Edge** - 100TB with compute capabilities
- **Snowmobile** - Exabyte-scale transfer truck

Migration Services:

- **DataSync** - Online data transfer
- **Storage Gateway** - Hybrid cloud storage
- **Database Migration Service** - Migrate databases with minimal downtime

Interview Tips

1. **Use the Well-Architected Framework** - Security, Reliability, Performance, Cost, Operational Excellence
2. **Think about trade-offs** - Cost vs performance, consistency vs availability
3. **Start simple, then add complexity** - Begin with basic architecture, then discuss improvements
4. **Ask clarifying questions** - Understand requirements before designing
5. **Mention alternatives** - Show you know multiple ways to solve problems

Common Scenario Questions

Q: Design a highly available web application

- Use multiple AZs, load balancer, auto scaling, RDS Multi-AZ

Q: How to handle sudden traffic spikes?

- Auto Scaling, CloudFront caching, Lambda for serverless scaling

Q: Secure a web application

- WAF, Security Groups, NACLs, encryption, IAM roles

Q: Cost optimization strategies

- Right-sizing instances, Reserved Instances, S3 lifecycle policies, CloudWatch monitoring

Remember: Focus on understanding concepts rather than memorizing details. Be ready to explain WHY you'd choose certain services and HOW they solve business problems. Good luck with your interview!