AWS Zero to Hero - Complete Study Notes

Comprehensive AWS & DevOps Learning Guide

From Cloud Computing Fundamentals to Advanced Container Orchestration & DNS Configuration

---

📚 Complete Table of Contents

- Attaching EBS Volumes to Running Instances
- EBS Volume Modification (Resize)
- Storage Operations on EC2 (Linux)
- EBS Snapshots Deep Dive
- Cross-Region Snapshot Copy
- Creating EBS from Snapshots
- Multiple EBS Attachment to Single Instance

Section 4: ELB, ALB, ASG, Launch Templates

- Load Balancing Fundamentals
- Elastic Load Balancer (ELB) Overview
- Application Load Balancer (ALB)
- Network Load Balancer (NLB)
- Gateway Load Balancer (GWLB)
- Target Groups
- Health Checks
- Auto Scaling Groups (ASG)
- Launch Templates
- Scaling Policies

Section 5: S3, Storage Classes, Lifecycle, Snow Family, RDS

- Amazon S3 (Simple Storage Service)
- S3 Storage Classes
- S3 Versioning
- S3 Lifecycle Policies
- S3 Replication
- S3 Security
- S3 Encryption
- AWS Snow Family
- Amazon RDS (Relational Database Service)
- RDS Multi-AZ vs Read Replicas

Section 6: DynamoDB, Lambda, CloudWatch

- Amazon DynamoDB
- DynamoDB Core Concepts
- DynamoDB Read/Write Capacity
- DynamoDB Indexes

- AWS Lambda
- Lambda Execution Model
- Lambda Triggers
- Amazon CloudWatch
- CloudWatch Logs
- CloudWatch Alarms

Section 7: Route53, DNS, CloudFront, CDN

- Amazon Route 53
- DNS Record Types
- Route 53 Routing Policies
- Health Checks
- Domain Registration
- Amazon CloudFront
- CloudFront Origins
- CloudFront Caching
- CloudFront Security
- Lambda@Edge

Section 8: VPC, Networking Complete

- VPC Overview
- VPC Components
- Subnets
- Internet Gateway (IGW)
- NAT Gateway vs NAT Instance
- Route Tables
- Security Groups vs NACLs
- VPC Peering
- VPC Endpoints
- VPN and Direct Connect

Section 9: ECS Container Management

- Container Fundamentals
- Docker Basics
- Amazon ECS Overview
- ECS Launch Types: EC2 vs Fargate
- Task Definitions
- ECS Services

- ECS Clusters
- IAM Roles for ECS
- ECS Service Auto Scaling
- AWS Amplify
- Amazon Cognito
- Sandbox Environment
- ECR (Elastic Container Registry)
- Port Mapping & Networking

Section 10: DNS Server Configuration

- DNS Overview
- What is DNS (Domain Name System)
- How DNS Works
- DNS Server Components
- Setting Up Your Own DNS Server
- Installing BIND9 DNS Server
- DNS Configuration Files
- Zone Files
- Forward Zone Configuration
- Reverse Zone Configuration
- DNS Records Types
- Testing DNS Configuration
- Web Server Setup with Apache
- Firewall Configuration
- Terraform Cloud Introduction

SECTION 1: Introduction, Virtualization, Cloud Computing & IAM

Course Introduction

What You'll Learn:

- **IT Fundamentals** - Basic understanding required for cloud
- **Website Development & Deployment** - How websites work in real world
- **Networking Basics** - IP, Ports, Routing, Internet Gateway
- **Cloud Computing** - What it is and how it works
- **AWS Services** - Hands-on with various AWS services
- **Infrastructure as Code (IaC)** - Terraform basics
- **Containers** - Docker and Kubernetes (EKS)
- **Full Stack Development** - Complete deployment on cloud

Who Should Watch:

- IT professionals wanting to learn cloud
- Developers wanting AWS knowledge
- Anyone in any IT role
- Support engineers
- Those preparing for AWS certifications

Virtualization Concepts

🔷 What is Virtualization?

Definition: Virtualization is the process of creating virtual (software-based) versions of physical hardware resources like servers, storage, networks, and operating systems.

In-Depth Explanation:

Think of virtualization like a building with multiple apartments. The building (physical server) has fixed resources - electricity, water, space. Instead of one family using the entire building, multiple families (virtual machines) share these resources efficiently, each with their own private space.

Real-World Analogy:

- **Without Virtualization:** One company buys 10 servers, each running at 10% capacity = 90% waste
- **With Virtualization:** One powerful server runs 10 VMs, each getting resources they need = 90% efficiency

Technical Deep Dive:

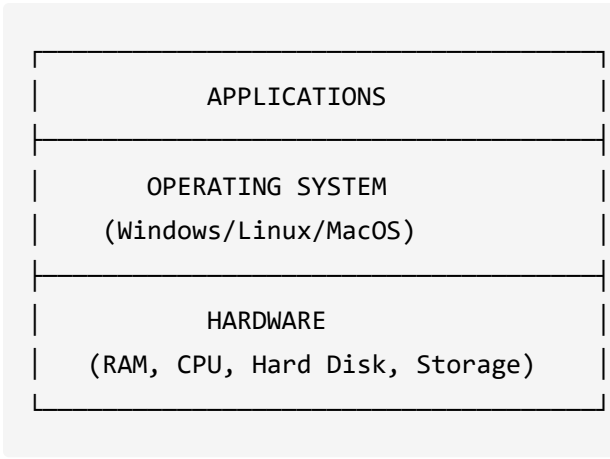
Virtualization works by inserting a **Hypervisor** (Virtual Machine Monitor) between hardware and operating systems. The hypervisor:

1. Abstracts physical hardware into virtual resources
2. Allocates CPU, memory, storage to each VM
3. Ensures isolation between VMs
4. Manages resource scheduling

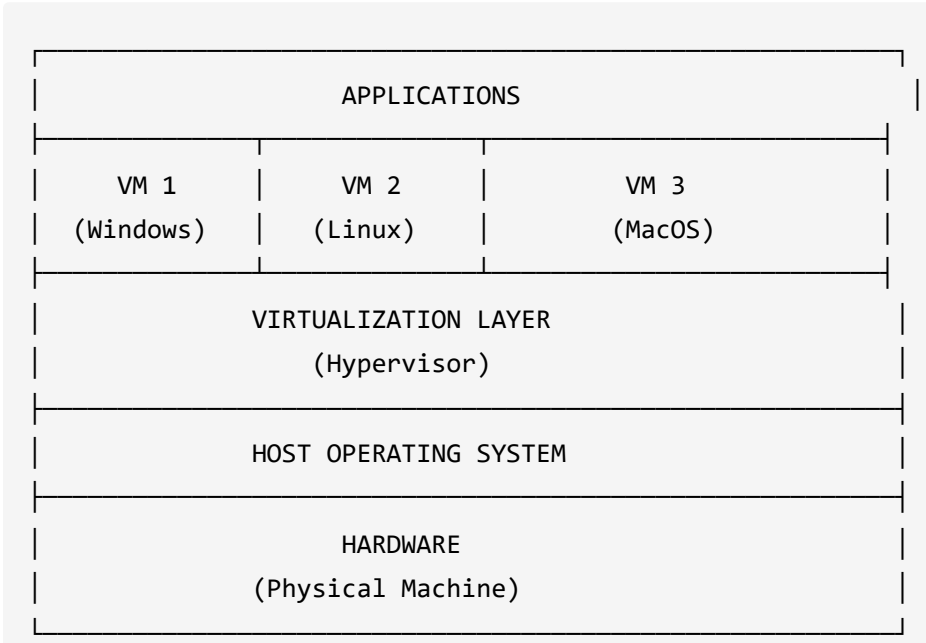
Types of Hypervisors:

Type	Description	Examples	Use Case
Type 1 (Bare Metal)	Runs directly on hardware	VMware ESXi, Microsoft Hyper-V, Xen	Enterprise data centers
Type 2 (Hosted)	Runs on top of OS	VirtualBox, VMware Workstation	Development, testing

Physical Machine Components:



How Virtualization Works:



Key Points:

- **Hypervisor** - Software that creates and manages virtual machines
- Multiple OS can run on single physical hardware
- Each VM gets allocated resources (RAM, CPU, Storage)
- VMs are isolated from each other

Benefits of Virtualization:

1. **Resource Optimization** - Better utilization of hardware (from 10-15% to 70-80%)
2. **Cost Savings** - Less physical hardware, reduced power and cooling costs
3. **Flexibility** - Run different OS on same hardware simultaneously
4. **Testing** - Test applications in different environments safely
5. **Isolation** - Problems in one VM don't affect others

6. **Disaster Recovery** - Easy backup and restoration of entire VMs
7. **Rapid Deployment** - Create new servers in minutes, not weeks

Virtualization vs Containerization:

Aspect	Virtualization	Containerization
Isolation	Complete OS isolation	Process-level isolation
Resource Usage	Heavy (full OS per VM)	Lightweight (shared OS kernel)
Boot Time	Minutes	Seconds
Use Case	Different OS environments	Microservices, same OS
Example	VMware, VirtualBox	Docker, Kubernetes

Cloud Computing Fundamentals

🔷 What is Cloud Computing?

Definition: Cloud computing is the delivery of computing services (servers, storage, databases, networking, software) over the internet ("the cloud") on a pay-as-you-go basis.

In-Depth Explanation:

Cloud computing is like using electricity from the power grid instead of running your own generator:

- **Before Cloud:** Companies bought servers, hired IT staff, maintained data centers, planned capacity years in advance
- **With Cloud:** Rent computing power by the hour/minute, scale instantly, pay only for what you use

The Five Essential Characteristics of Cloud (NIST Definition):

1. **On-Demand Self-Service:** Provision resources automatically without human interaction
2. **Broad Network Access:** Available over the network from any device
3. **Resource Pooling:** Provider's resources serve multiple customers (multi-tenancy)
4. **Rapid Elasticity:** Scale up/down quickly based on demand
5. **Measured Service:** Pay only for what you consume (metering)

Cloud Deployment Models:

Model	Description	Use Case	Example
Public Cloud	Resources owned by third-party	Startups, variable workloads	AWS, Azure, GCP
Private Cloud	Dedicated to single organization	Banks, healthcare, government	On-premise OpenStack
Hybrid Cloud	Mix of public and private	Sensitive + variable workloads	AWS Outposts
Multi-Cloud	Using multiple cloud providers	Avoid vendor lock-in	AWS + Azure together

Traditional IT vs Cloud Computing:

Traditional IT	Cloud Computing
Buy hardware upfront	Pay as you go
Manage your own servers	Provider manages infrastructure
Fixed capacity	Scalable capacity
High CAPEX	OPEX model
Long setup time	Quick provisioning

Types of Cloud Services:

1. IaaS (Infrastructure as a Service)

- **What:** Raw computing resources (VMs, Storage, Networks)
- **Example:** AWS EC2, Azure VMs
- **You Manage:** OS, Applications, Data
- **Provider Manages:** Hardware, Networking, Storage

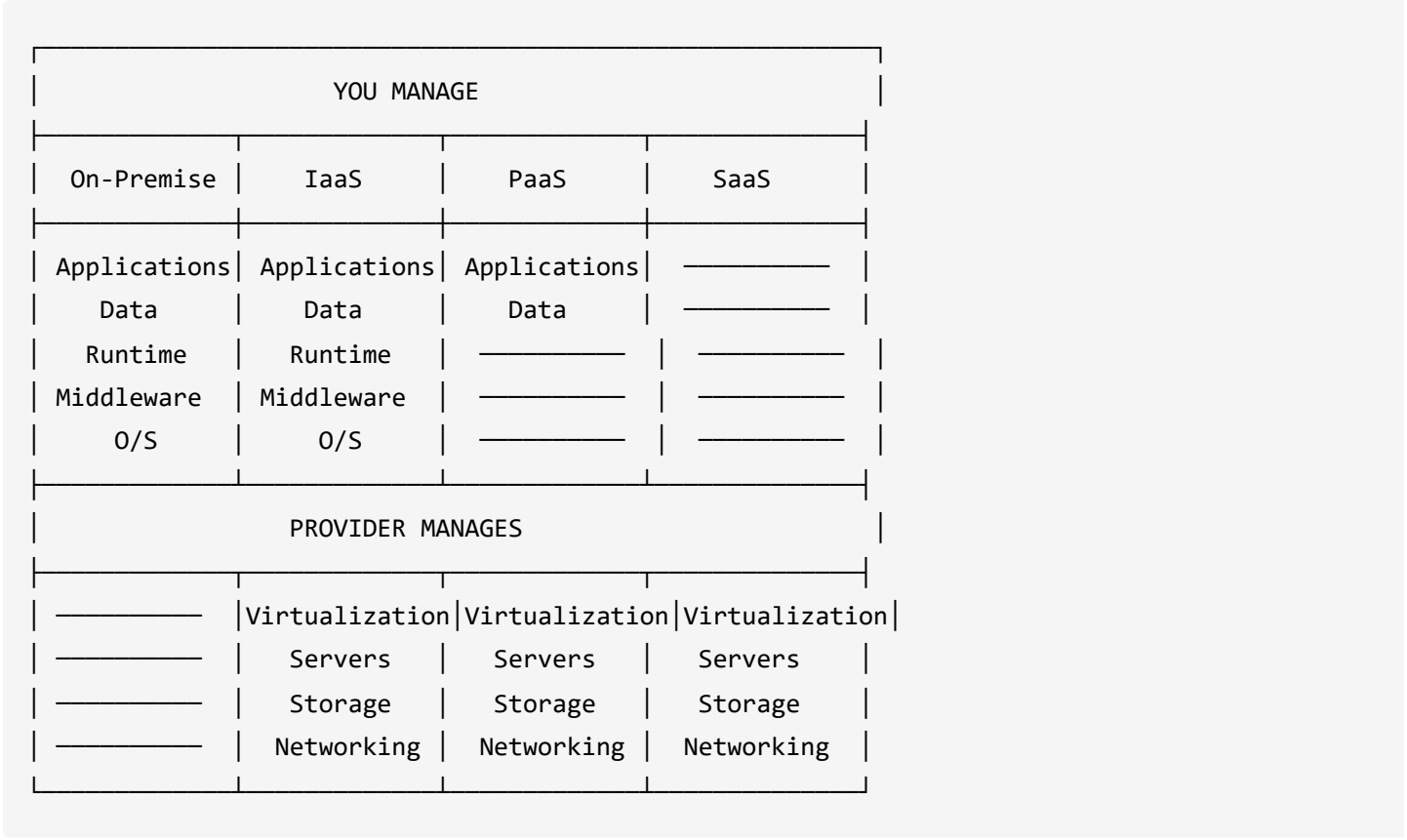
2. PaaS (Platform as a Service)

- **What:** Platform for building applications
- **Example:** AWS Elastic Beanstalk, Heroku
- **You Manage:** Applications, Data
- **Provider Manages:** OS, Runtime, Hardware

3. SaaS (Software as a Service)

- **What:** Complete software delivered via internet
- **Example:** Gmail, Salesforce, Office 365
- **You Manage:** Just use the software
- **Provider Manages:** Everything

Cloud Service Model Diagram:



Types of Scaling:

Vertical Scaling (Scale Up)

- Increase resources of existing server
- Add more RAM, CPU, Storage
- Has limitations (hardware max)
- **Example:** Upgrading from t2.micro to t2.large

Horizontal Scaling (Scale Out)

- Add more servers/instances
- Distribute load across multiple servers
- Preferred for high availability
- **Example:** Adding more EC2 instances behind a load balancer

CapEx vs OpEx Model:

Aspect	CapEx (Traditional)	OpEx (Cloud)
Payment	Large upfront investment	Pay monthly/hourly
Ownership	You own the hardware	Rent resources
Risk	High (capacity planning)	Low (adjust as needed)
Flexibility	Limited	High
Tax Treatment	Depreciation over years	Operating expense

AWS Overview

🔷 What is AWS?

Amazon Web Services (AWS) is the world's most comprehensive and widely adopted cloud platform, offering over 200 fully featured services from data centers globally.

AWS History:

- **2002:** Amazon began offering web services
- **2004:** SQS (Simple Queue Service) launched
- **2006:** AWS officially launched with S3 and EC2
- **2024+:** Market leader with 32%+ market share

Why Companies Choose AWS:

1. **Market Leader** - Largest cloud provider with most services
2. **Global Infrastructure** - 33+ regions, 100+ availability zones
3. **200+ Services** - Compute, storage, ML, IoT, analytics
4. **Security** - Meets compliance standards (HIPAA, PCI, SOC)
5. **Cost Effective** - Pay-as-you-go with various discount options
6. **Innovation** - Constantly adding new services
7. **Community** - Large ecosystem, documentation, support

AWS Global Infrastructure:

```
┌─────────────────────────────────────────────────────┐
│                    AWS GLOBAL                       │
├─────────────────────────────────────────────────────┤
│  REGIONS (25+)                                      │
│  ├── Availability Zones (AZ)                        │
│  │   ├── Data Center 1                              │
│  │   ├── Data Center 2                              │
│  │   └── Data Center 3                              │
│  └── Edge Locations (CDN/CloudFront)                │
└─────────────────────────────────────────────────────┘
```

Key Concepts:

- **Region:** Geographic area with multiple AZs
- **Availability Zone (AZ):** One or more data centers
- **Edge Location:** CDN endpoints for faster content delivery

AWS Free Tier:

- **12 Months Free:** EC2, S3, RDS (limited)
- **Always Free:** Lambda, DynamoDB (limited)
- **Trials:** Short-term free trials of services

IAM (Identity & Access Management)

🔷 What is IAM?

IAM is AWS's identity and access management service that helps you securely control access to AWS resources. It answers two fundamental questions:

1. **Authentication:** WHO are you? (Identity)
2. **Authorization:** WHAT can you do? (Permissions)

In-Depth Explanation:

Think of IAM as the security system of an office building:

- **Root Account** = Building owner (has all keys, shouldn't be used daily)
- **IAM Users** = Employees with ID badges

- **IAM Groups** = Departments (Engineering, HR, Finance)
- **IAM Policies** = Access rules (who can enter which rooms)
- **IAM Roles** = Temporary visitor passes

Why IAM is Critical:

- **Security:** Prevent unauthorized access to resources
- **Compliance:** Audit trail of who did what
- **Principle of Least Privilege:** Give minimum permissions needed
- **Cost Control:** Prevent accidental resource creation

IAM Components:

1. Root Account

- Created when you first set up AWS account
- Has **complete access** to all AWS services
- ⚠️ **NEVER use for daily tasks**
- Enable MFA immediately
- Create IAM users instead

2. IAM Users

- Individual identities for people/applications
- Can have:
  - Console access (password)
  - Programmatic access (Access Keys)
- Principle of Least Privilege

3. IAM Groups

- Collection of users
- Assign permissions to group
- Users inherit group permissions
- Makes permission management easier

4. IAM Policies

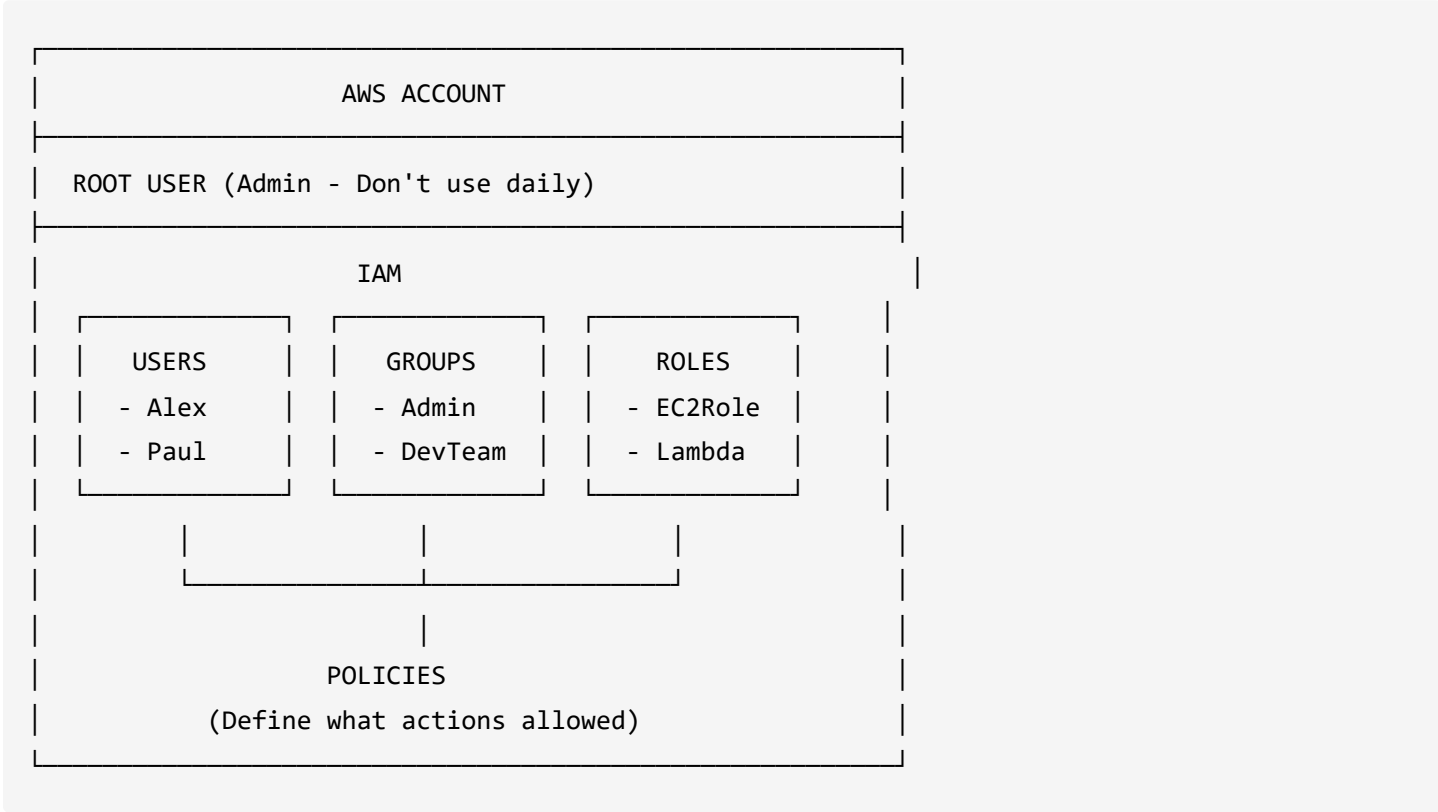
- JSON documents defining permissions
- Attached to Users, Groups, or Roles
- Types:
  - AWS Managed Policies
  - Customer Managed Policies

- Inline Policies

5. IAM Roles

- Temporary permissions
- For AWS services or external identities
- No permanent credentials
- Assumed when needed

IAM Architecture:



IAM Policy Example (JSON):

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "*"
        }
    ]
}
```

IAM Best Practices:

1. ✅ Enable MFA on root account
2. ✅ Create individual IAM users
3. ✅ Use groups to assign permissions
4. ✅ Grant least privilege
5. ✅ Use roles for applications
6. ✅ Rotate credentials regularly
7. ✅ Use strong password policy
8. ✅ Never share credentials

Creating IAM User - Steps:

1. Go to IAM Dashboard
2. Click "Users" → "Create User"
3. Enter username
4. Set permissions (Add to group or attach policy)
5. Set password policy
6. Enable console access if needed
7. Download credentials (CSV)

User vs Group Permissions:

Aspect	User Direct	Through Group
Management	Complex with many users	Easy - one change affects all
Best Practice	Avoid for teams	Recommended
Flexibility	Individual control	Standardized access

MFA (Multi-Factor Authentication)

🔷 What is MFA?

Multi-Factor Authentication adds an extra layer of security beyond username and password.

MFA Factors:

1. **Something you know** - Password
2. **Something you have** - Phone/Token device
3. **Something you are** - Biometrics

MFA Options in AWS:

- **Virtual MFA** - Google Authenticator, Authy
- **Hardware MFA** - Physical token device
- **U2F Security Key** - YubiKey

Why Enable MFA?

- Protects against password compromise
- **Critical for root account**
- Recommended for all IAM users
- Industry best practice

Enabling MFA Steps:

1. Go to IAM → Users → Select user
2. Security credentials tab
3. Assign MFA device
4. Follow setup wizard

5. Enter two consecutive codes to verify

Important Q&A (Real Interview Questions) - Section 1

Q1: What is the difference between Root User and IAM User?

Answer:

Root User	IAM User
Created with AWS account	Created by admin/root
Has complete, unrestricted access	Has limited, defined access
Cannot be restricted by policies	Can be restricted by policies
Only for billing and emergency	For daily administrative tasks
One per AWS account	Multiple allowed (up to 5000)
Should have MFA enabled	Should have MFA enabled

Interview Tip: Always mention that root user should NEVER be used for daily tasks.

Q2: What is the Principle of Least Privilege?

Answer: The principle of least privilege means giving users/applications only the minimum permissions they need to perform their job functions. This reduces the attack surface and limits damage from compromised credentials.

Example: A developer who only needs to read S3 buckets should NOT have `s3:*` (all S3 permissions) but only `s3:GetObject` and `s3:ListBucket`.

Q3: Explain the difference between IAM Users, Groups, Roles, and Policies.

Answer:

- **IAM User:** An identity representing a person or application with permanent credentials
- **IAM Group:** A collection of users; makes permission management easier
- **IAM Role:** A set of permissions that can be assumed temporarily by users, applications, or AWS services

- **IAM Policy:** A JSON document that defines permissions (what actions are allowed/denied on which resources)

Key Point: Users and Roles are identities. Policies are permissions. Groups are for organizing users.

Q4: What is the difference between IAM Role and IAM User?

Answer:

IAM User	IAM Role
Has permanent long-term credentials	Has temporary credentials
For people or applications	For AWS services or cross-account access
Has password and/or access keys	No permanent credentials
Directly assigned permissions	Permissions assumed when needed
Cannot be assumed	Can be assumed by users, services, or accounts

Use Case Example:

- EC2 instance needs to access S3 → Use IAM Role (not access keys)
- Developer needs AWS CLI access → Use IAM User

Q5: What happens when you attach multiple policies to a user?

Answer: AWS uses a **union** of all permissions. If any policy allows an action, it's allowed (unless explicitly denied). The evaluation order is:

1. Explicit Deny (always wins)
2. Explicit Allow
3. Implicit Deny (default)

Q6: What is IAM Policy Structure?

Answer: An IAM policy has these elements:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UniqueIdentifier",
      "Effect": "Allow/Deny",
      "Action": ["service:action"],
      "Resource": "arn:aws:service:region:account:resource",
      "Condition": {"operator": {"key": "value"}}
    }
  ]
}
```

- **Version:** Always use "2012-10-17"
- **Effect:** Allow or Deny
- **Action:** What API calls are permitted
- **Resource:** Which AWS resources are affected
- **Condition:** When the policy applies (optional)

Q7: What is AWS STS and how does it relate to IAM Roles?

Answer: AWS Security Token Service (STS) is the service that issues temporary credentials when a role is assumed. When you call `AssumeRole`, STS returns:

- Temporary Access Key ID
- Temporary Secret Access Key
- Session Token
- Expiration time (default 1 hour, max 12 hours)

Q8: How would you grant an EC2 instance access to S3?

Answer:

1. **Best Practice (Use IAM Role):**
   - Create an IAM role with S3 permissions
   - Attach the role to the EC2 instance
   - The instance automatically gets temporary credentials
2. **NOT Recommended (Access Keys):**
   - Never store access keys on EC2 instances
   - If instance is compromised, keys are exposed

Q9: What is the difference between AWS Managed Policy and Customer Managed Policy?

Answer:

AWS Managed Policy	Customer Managed Policy
Created and managed by AWS	Created and managed by you
Cannot be modified	Fully customizable
Automatically updated by AWS	You must update manually
Good for common use cases	For specific requirements
Example: AmazonS3ReadOnlyAccess	Custom policy for your application

Q10: What is Cross-Account Access and how do you implement it?

Answer: Cross-account access allows resources in one AWS account to access resources in another account.

Implementation:

1. Account B creates an IAM role with trust policy allowing Account A
2. Attach permissions policy to the role
3. Account A users assume the role using `sts:AssumeRole`

Q11: What is MFA and why is it important?

Answer: Multi-Factor Authentication adds an extra layer of security requiring:

1. Something you know (password)
2. Something you have (MFA device/phone)

Why Important:

- Protects against password compromise
- Required for compliance (PCI-DSS, HIPAA)
- Especially critical for root and admin accounts
- Can be required for specific actions (e.g., deleting resources)

Q12: What is Identity Federation in AWS?

Answer: Identity Federation allows users from external identity providers to access AWS resources without creating IAM users. Types:

- **SAML 2.0:** Enterprise identity providers (Active Directory, Okta)
- **Web Identity:** Social logins (Google, Facebook) via Cognito
- **Custom Identity Broker:** For non-SAML enterprise systems

Q13: What are IAM Access Keys and when should you use them?

Answer: Access keys consist of Access Key ID + Secret Access Key, used for programmatic access (CLI, SDK, API).

Best Practices:

- Never embed in code
- Rotate regularly (90 days recommended)
- Use IAM roles instead when possible
- Delete unused access keys
- Never share or commit to version control

Q14: Explain the IAM Policy Evaluation Logic.

Answer: AWS evaluates policies in this order:

1. **Explicit Deny:** If any policy explicitly denies → DENIED
2. **Organization SCP:** If SCP denies → DENIED
3. **Resource-based Policy:** Check if resource policy allows
4. **Identity-based Policy:** Check user/role policies
5. **IAM Permission Boundary:** If defined, must also allow
6. **Session Policy:** For role sessions, must also allow
7. **Default:** If nothing allows → DENIED (implicit deny)

Q15: What is the difference between Resource-based Policy and Identity-based Policy?

Answer:

Identity-based Policy	Resource-based Policy
Attached to IAM users, groups, roles	Attached to AWS resources

Identity-based Policy	Resource-based Policy
Specifies what actions identity can perform	Specifies who can access the resource
Example: IAM user policy	Example: S3 bucket policy
Must use ARN for resources	Must specify Principal

📝 Key Takeaways - Section 1

1. **Virtualization** enables running multiple OS on single hardware
2. **Cloud Computing** provides on-demand IT resources over internet
3. **AWS** is the leading cloud provider with 200+ services
4. **IAM** controls who can access what in AWS
5. **Root Account** should be protected with MFA and rarely used
6. **Groups** simplify permission management for multiple users
7. **Policies** define permissions in JSON format
8. **MFA** adds critical security layer

SECTION 2: MFA Device Setup, AWS CLI, EC2 Instances & EBS Storage

1. MFA Device Setup

What is MFA?

Multi-Factor Authentication (MFA) adds an extra layer of security to your AWS account by requiring:

1. Something you know (password)
2. Something you have (MFA device)

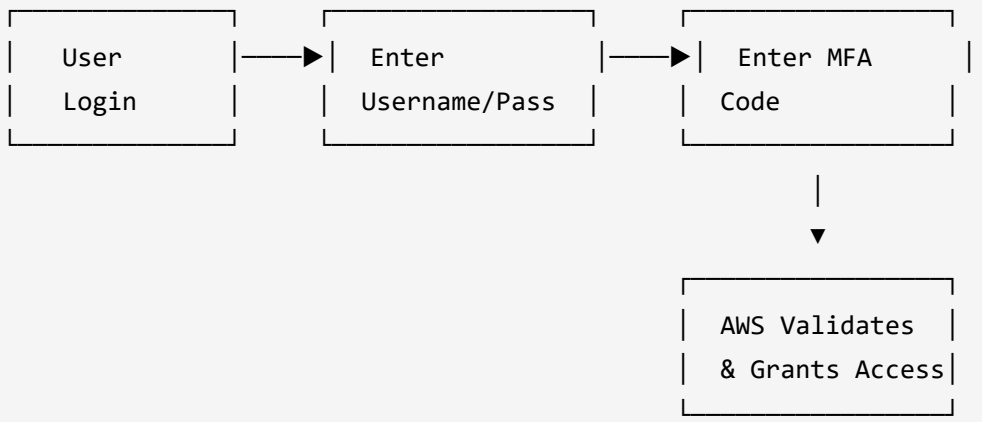
MFA Device Options

Option	Description	Use Case
Authenticator App	Google Authenticator, Authy	Most common, mobile-based
Hardware TOTP Token	Physical token device	High security environments
Security Keys	FIDO U2F keys	Enterprise security
SMS (Deprecated)	Text message codes	Not recommended

Setting Up MFA (Step-by-Step)

```
1. Go to IAM → Users → Select User → Security Credentials
2. Under MFA, click "Assign MFA device"
3. Give the device a name
4. Choose device type (Authenticator App recommended)
5. Install Google Authenticator on your mobile
6. Scan the QR code displayed
7. Enter two consecutive MFA codes (30 seconds apart)
8. Click "Add MFA"
```

Diagram: MFA Authentication Flow



---

2. AWS Access Methods

AWS provides **three ways** to access the platform:

Method 1: Management Console (Browser)

- GUI-based interface
- Good for beginners
- Manual operations
- Visual dashboard

Method 2: AWS CLI (Command Line Interface)

- Text-based commands
- Automation capable
- Scripting support

- Used by DevOps engineers

Method 3: SDKs & APIs

- Programmatic access
- Integration with applications
- Supports multiple languages (Python, Java, Node.js, etc.)
- Used for application development

Comparison Table

Feature	Console	CLI	SDK/API
Ease of Use	★★★★★	★★★☆☆	★★☆☆☆
Automation	✗	✓	✓
Scripting	✗	✓	✓
Bulk Operations	✗	✓	✓
Learning Curve	Low	Medium	High

---

3. AWS CLI Setup & Configuration

Installation Steps

Windows:

```
# Download MSI installer from AWS website
# Or use winget
winget install Amazon.AWSCLI

# Verify installation
aws --version
```

Linux/Mac:

```
# Using curl
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
unzip awscliv2.zip
sudo ./aws/install

# Verify
aws --version
```

CLI Configuration

```
# Configure AWS CLI
aws configure

# You'll be prompted for:
# 1. AWS Access Key ID
# 2. AWS Secret Access Key
# 3. Default region (e.g., ap-south-1)
# 4. Default output format (json/yaml/table/text)
```

Important AWS CLI Commands

```
# IAM Commands
aws iam list-users                    # List all IAM users
aws iam create-user --user-name alex  # Create new user
aws iam delete-user --user-name alex  # Delete user

# EC2 Commands
aws ec2 describe-instances            # List all instances
aws ec2 start-instances --instance-ids i-xxx
aws ec2 stop-instances --instance-ids i-xxx

# S3 Commands
aws s3 ls                             # List all buckets
aws s3 cp file.txt s3://bucket-name/  # Upload file
```

Access Key Best Practices

⚠️ SECURITY WARNINGS:

- Never share access keys
- Never commit keys to Git
- Rotate keys regularly
- Use IAM roles when possible
- Delete unused access keys

---

4. EC2 Instances - Deep Dive

What is EC2?

Elastic Compute Cloud (EC2) is AWS's core compute service that provides resizable virtual servers in the cloud.

In-Depth Explanation:

Think of EC2 as renting a computer:

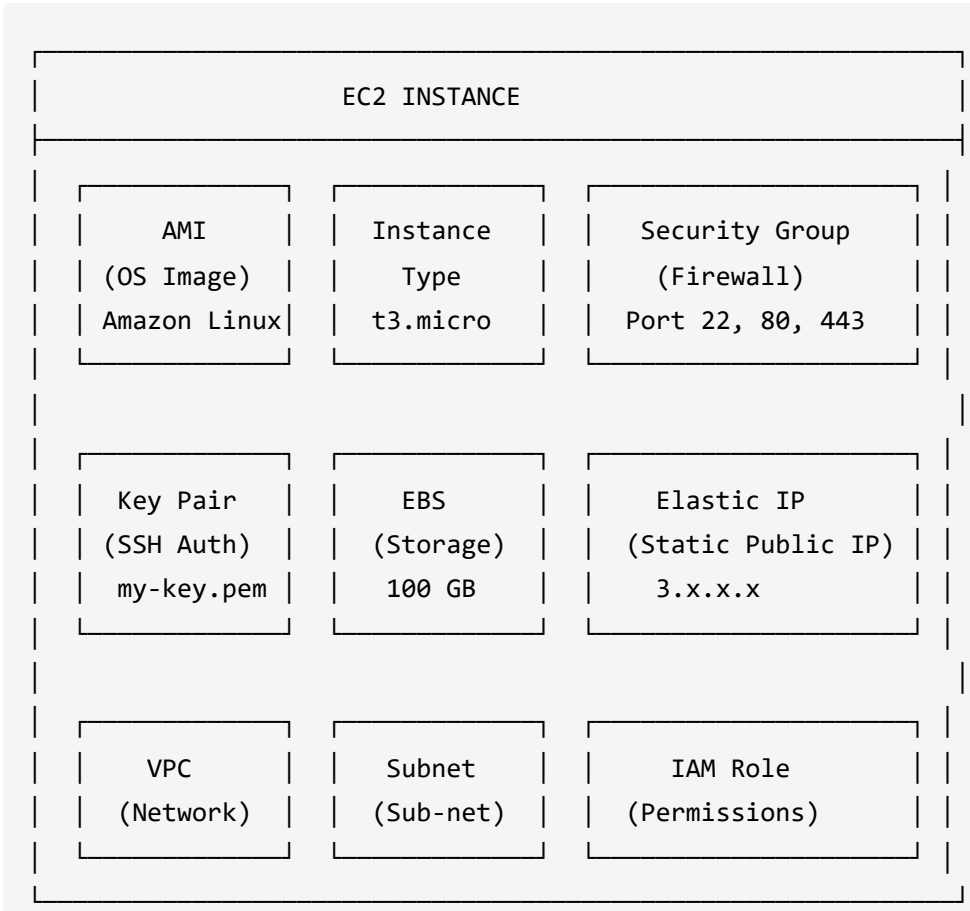
- **Before EC2:** Buy servers ($10,000+), wait weeks for delivery, manage hardware, pay even when idle
- **With EC2:** Launch a server in minutes, pay by the hour/second, scale up/down instantly, terminate when not needed

Why "Elastic"?

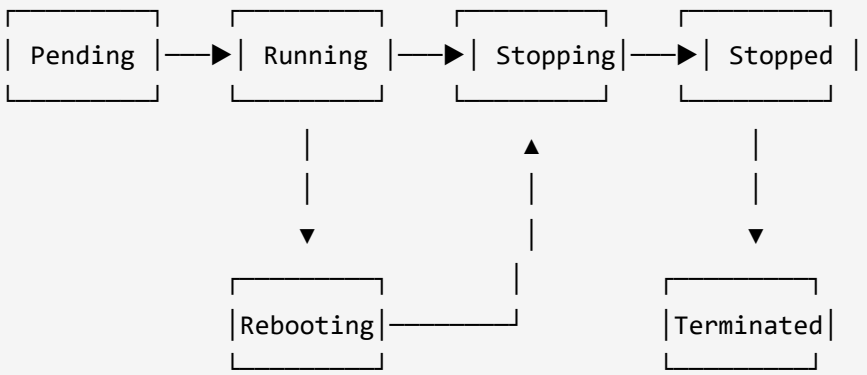
The term "Elastic" means you can:

1. **Scale vertically:** Change instance type (t3.micro → t3.large)
2. **Scale horizontally:** Add/remove instances based on demand
3. **Pay elastically:** Only pay for what you use

EC2 Architecture:



EC2 Instance Lifecycle



Key Components

1. **AMI (Amazon Machine Image)** - OS template
2. **Instance Type** - Hardware configuration
3. **Security Groups** - Virtual firewall
4. **Key Pairs** - SSH authentication
5. **EBS Volumes** - Storage

5. Instance Types & Categories

Instance Naming Convention

```
Example: t3.large

t  = Instance Family (General Purpose)
3  = Generation (3rd generation)
.  = Separator
large = Size
```

Instance Families

Family	Purpose	Use Cases
T (General Purpose)	Balanced compute, memory, networking	Web servers, small databases
M (General Purpose)	Balanced resources	Application servers
C (Compute Optimized)	High CPU performance	Batch processing, gaming
R (Memory Optimized)	Large memory	Databases, caching
G/P (Accelerated Computing)	GPU instances	Machine learning, graphics
I (Storage Optimized)	High I/O	Data warehousing

T-Series Sizes Comparison

```
┌──────────────┬──────────┬──────────┬─────────────┐
│ Instance     │ vCPUs    │ Memory   │ Use Case    │
├──────────────┼──────────┼──────────┼─────────────┤
│ t3.nano      │ 2        │ 0.5 GB   │ Micro       │
│ t3.micro     │ 2        │ 1 GB     │ Small       │
│ t3.small     │ 2        │ 2 GB     │ Dev/Test    │
│ t3.medium    │ 2        │ 4 GB     │ Small App   │
│ t3.large     │ 2        │ 8 GB     │ Production  │
│ t3.xlarge    │ 4        │ 16 GB    │ Large App   │
│ t3.2xlarge   │ 8        │ 32 GB    │ High Load   │
└──────────────┴──────────┴──────────┴─────────────┘
```

Use Case Examples

1. Simple Website/Blog

```
Instance Type: t3.micro or t3.small
- Low traffic
- Static content
- Simple CMS
```

2. E-commerce Application

```
Instance Type: m5.large or m5.xlarge
- Higher traffic
- Database operations
- More memory needed
```

3. Video Rendering/Streaming

```
Instance Type: g5 series (GPU)
- Real-time processing
- High compute power
- GPU acceleration needed
```

4. In-Memory Database (Real-time Analytics)

```
Instance Type: r6g series (Memory Optimized)
- Large RAM requirements
- Fast data access
- Analytics workloads
```

---

6. Purchasing Options

Pricing Models Comparison

Option	Commitment	Discount	Best For
On-Demand	None	0%	Variable workloads
Reserved	1-3 years	Up to 75%	Steady-state workloads

Option	Commitment	Discount	Best For
Spot	None	Up to 90%	Flexible, interruptible
Savings Plans	1-3 years	Up to 72%	Flexible commitment
Dedicated Hosts	Varies	Varies	Compliance requirements

On-Demand Instances

```
✓ Pay per second (Linux) or per hour (Windows)
✓ No upfront commitment
✓ Most flexible
✓ Best for: Testing, unpredictable workloads

✗ Most expensive option
```

Reserved Instances

```
✓ Up to 75% discount
✓ Predictable costs
✓ Best for: Steady, predictable workloads

✗ Requires 1-3 year commitment
✗ Less flexible
```

Spot Instances

```
✓ Up to 90% discount
✓ Access to spare capacity
✓ Best for: Batch jobs, CI/CD, fault-tolerant apps

✗ Can be interrupted with 2-minute warning
✗ Not suitable for critical applications
```

Dedicated Hosts

```
✓ Physical server dedicated to you
✓ Hardware control
✓ Best for: Compliance, licensing requirements

✗ Most expensive
✗ Limited flexibility
```

---

7. EBS - Elastic Block Store

What is EBS?

Elastic Block Store (EBS) provides persistent block storage volumes for EC2 instances - think of it as a virtual hard drive that exists independently of the EC2 instance.

In-Depth Explanation:

Analogy: EBS is like an external hard drive for your computer:

- It stores data persistently (data survives power off)
- You can detach from one computer and attach to another
- You can make backups (snapshots)
- It exists independently from the computer

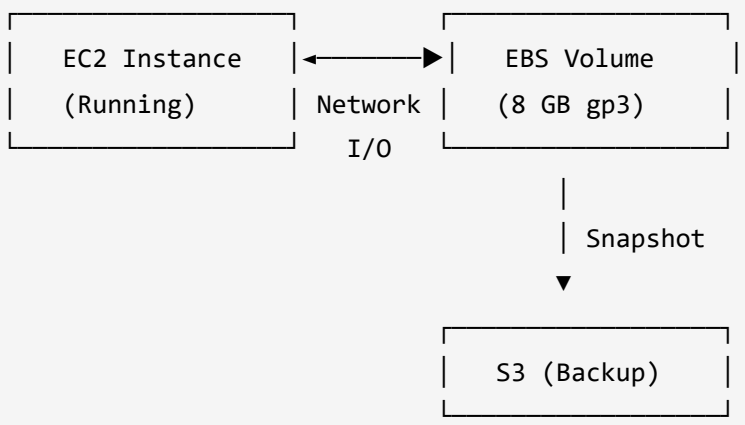
Why EBS is Important:

1. **Persistence:** Data survives instance stop/start
2. **Flexibility:** Resize, change type without data loss
3. **Backup:** Snapshots for disaster recovery
4. **Encryption:** Protect sensitive data at rest
5. **High Availability:** Automatically replicated within AZ

Key Characteristics

- Works like a virtual hard drive
- Data persists independently from instance
- Can be attached/detached from instances
- Supports snapshots for backup
- Available in different types for various workloads

How EBS Works



EBS Volume Types

Type	Name	Use Case	IOPS	Throughput
gp3	General Purpose SSD	General workloads	3,000-16,000	125-1000 MB/s
gp2	General Purpose SSD	Boot volumes	100-16,000	128-250 MB/s
io1/io2	Provisioned IOPS	Databases	Up to 64,000	1,000 MB/s
st1	Throughput Optimized HDD	Big data	500	500 MB/s
sc1	Cold HDD	Infrequent access	250	250 MB/s

When to Use Which Type

```
gp3 (Recommended Default):
├── Web servers
├── Development environments
├── Small to medium databases
└── Boot volumes

io1/io2 (High Performance):
├── Production databases (MySQL, PostgreSQL)
├── I/O intensive applications
├── Applications requiring consistent IOPS
└── Mission-critical workloads

st1 (Throughput Optimized):
├── Big data processing
├── Data warehouses
├── Log processing
└── Streaming workloads

sc1 (Cold Storage):
├── Infrequently accessed data
├── Archive storage
└── Lowest cost requirement
```

EBS Features

1. Snapshots

- Point-in-time backup of EBS volume
- Stored in S3 (managed by AWS)
- Can create new volumes from snapshots
- Cross-region copy supported
- Incremental backups (only changed blocks)

2. Encryption

- Data-at-rest encryption
- Data-in-transit encryption
- Uses AWS KMS keys
- No performance impact
- Enabled at volume creation

3. Resizing

- Can increase size without downtime
- Can change volume type
- No need to restart instance
- Changes take time to apply

EBS Best Practices

✓ Use gp3 as default (better performance, lower cost)
✓ Enable encryption for sensitive data
✓ Take regular snapshots
✓ Use io2 for critical databases
✓ Monitor performance metrics
✓ Delete unused volumes to save costs

EBS vs Instance Store

Feature	EBS	Instance Store
Persistence	✓	✗
Data survives stop	✓	✗
Snapshots	✓	✗
Network attached	✓	✗ (Local)
Performance	Good	Very High
Use case	General	Temporary cache

---

8. Important Q&A (Real Interview Questions) - Section 2

Q1: What is EC2 and why is it important?

Answer: EC2 (Elastic Compute Cloud) is AWS's core compute service that provides virtual servers in the cloud. It's important because:

- Eliminates need to invest in hardware upfront

- Scale capacity in minutes, not weeks
- Pay only for capacity you use
- Foundation for many AWS architectures

Q2: Explain EC2 instance states and their differences.

Answer:

State	Description	Billing
Pending	Instance is launching	No charge
Running	Instance is active	Charged
Stopping	Instance is stopping	Charged until stopped
Stopped	Instance is off	No compute charge, EBS charged
Terminated	Instance is deleted	No charge
Rebooting	Instance is restarting	Charged

Key Point: Stopped instances don't incur compute charges but EBS volumes are still charged.

Q3: What happens to EBS data when EC2 is terminated?

Answer: By default:

- **Root volume:** Deleted (DeleteOnTermination = true)
- **Additional volumes:** Retained (DeleteOnTermination = false)

You can modify this behavior during launch or via CLI:

```
aws ec2 modify-instance-attribute --instance-id i-xxx \
  --block-device-mappings "[{\"DeviceName\":\"/dev/xvda\",\"Ebs\":{\"DeleteOnTermination\":fals
```

Q4: Can we attach one EBS volume to multiple EC2 instances?

Answer:

- **Standard EBS:** No, one volume to one instance only
- **EBS Multi-Attach (io1/io2 only):** Yes, up to 16 Nitro instances in the same AZ

Use Case for Multi-Attach: Clustered applications like Oracle RAC, shared storage for high availability.

Q5: What is the difference between gp2 and gp3?

Answer:

Feature	gp2	gp3
Baseline IOPS	3 IOPS/GB (scales with size)	3,000 IOPS (independent)
Max IOPS	16,000	16,000
Throughput	128-250 MB/s	125-1000 MB/s
Cost	Higher	20% cheaper
Configuration	Size-based	Independent IOPS/throughput

Interview Tip: Always recommend gp3 for new deployments unless specific gp2 requirements exist.

Q6: What is the difference between Instance Store and EBS?

Answer:

Aspect	EBS	Instance Store
Persistence	Data persists	Data lost on stop/terminate
Attachment	Network-attached	Physically attached
Reboot behavior	Data preserved	Data preserved
Stop/Terminate	Data preserved (EBS)	Data lost
Use Case	Databases, boot volumes	Temp data, cache, buffers
Snapshots	Supported	Not supported

Q7: How do you choose the right EC2 instance type?

Answer: Consider these factors:

1. **Workload type:** Compute, memory, storage, or GPU intensive?
2. **Resource requirements:** How much CPU, RAM needed?
3. **Budget:** Cost constraints?
4. **Compliance:** Any specific hardware requirements?

Quick Guide:

- General purpose (T, M) → Web servers, small databases

- Compute optimized (C) → Batch processing, high-performance computing
- Memory optimized (R, X) → Large databases, in-memory caching
- Storage optimized (I, D) → Data warehousing, distributed file systems
- GPU (P, G) → Machine learning, graphics rendering

Q8: Explain the difference between On-Demand, Reserved, and Spot instances.

Answer:

Type	Commitment	Discount	Interruption	Best For
On-Demand	None	0%	Never	Variable workloads
Reserved	1-3 years	Up to 75%	Never	Steady-state
Spot	None	Up to 90%	Yes (2 min notice)	Flexible, fault-tolerant

Interview Tip: Mention that production databases should NEVER use Spot instances.

Q9: What is a Security Group? How is it different from NACL?

Answer:

Aspect	Security Group	NACL
Level	Instance level	Subnet level
State	Stateful	Stateless
Rules	Allow only	Allow and Deny
Default	Deny all inbound	Allow all
Evaluation	All rules evaluated	Rules evaluated in order

Q10: What happens when you stop vs terminate an EC2 instance?

Answer:

Action	Stop	Terminate
Instance state	Stopped	Terminated
EBS root volume	Preserved	Deleted (by default)

Action	Stop	Terminate
Instance charges	No	No
EBS charges	Yes	No (if deleted)
Public IP	Released	Released
Elastic IP	Retained	Retained (but charged)
Can restart?	Yes	No

Q11: How do you connect to an EC2 instance?

Answer:

- **Linux:** SSH using key pair

```
ssh -i mykey.pem ec2-user@<public-ip>
```

- **Windows:** RDP using password (retrieved with key pair)
- **Session Manager:** Browser-based, no open ports required (requires IAM role)
- **EC2 Instance Connect:** Browser-based SSH

Q12: What is User Data in EC2?

Answer: User Data is a script that runs automatically when an instance first launches. Used for:

- Installing software
- Applying patches
- Configuring applications

```
#!/bin/bash
yum update -y
yum install -y httpd
systemctl start httpd
echo "Hello World" > /var/www/html/index.html
```

Q13: What is EC2 Placement Group?

Answer: Placement groups control how instances are placed on underlying hardware:

Type	Description	Use Case
Cluster	Instances close together in single AZ	Low latency, HPC

Type	Description	Use Case
Spread	Instances on different hardware	High availability
Partition	Instances in logical partitions	Hadoop, Cassandra

Q14: How does EBS pricing work?

Answer: EBS pricing components:

1. **Volume storage:** $/GB-month
2. **Provisioned IOPS:** $/IOPS-month (io1/io2)
3. **Provisioned throughput:** $/MB/s-month (gp3)
4. **Snapshots:** $/GB-month (S3 storage)
5. **Data transfer:** Cross-AZ transfer charged

Q15: What is the maximum size of an EBS volume?

Answer:

- **Maximum size:** 64 TiB (64,000 GB)
- **Minimum size:** 1 GiB (gp2/gp3), 4 GiB (io1/io2)

Note: You can only increase EBS size, not decrease.

Quick Reference Commands - Section 2

```
# EC2 Operations
aws ec2 describe-instances
aws ec2 run-instances --image-id ami-xxx --instance-type t3.micro
aws ec2 terminate-instances --instance-ids i-xxx

# EBS Operations
aws ec2 describe-volumes
aws ec2 create-volume --size 100 --volume-type gp3 --availability-zone ap-south-1a
aws ec2 attach-volume --volume-id vol-xxx --instance-id i-xxx --device /dev/sdf
aws ec2 create-snapshot --volume-id vol-xxx --description "Backup"

# Check AWS CLI configuration
aws configure list
aws sts get-caller-identity
```

Summary - Section 2

Topic	Key Points
MFA	Additional security layer, use Authenticator apps
AWS CLI	Command-line access, automation, scripting
EC2	Virtual servers, multiple types, flexible pricing
Instance Types	T (general), C (compute), R (memory), G (GPU)
Pricing	On-Demand, Reserved, Spot, Savings Plans
EBS	Persistent storage, gp3 default, snapshots for backup

SECTION 4: Elastic Load Balancing (ELB), Auto Scaling Groups (ASG) & Launch Templates

---