**🧱 What is EBS?**

Think of **EBS (Elastic Block Store)** like a **hard drive** (or SSD) for your **EC2 virtual machine** in AWS.

* Just like your laptop needs a hard disk to store files, your EC2 instance needs storage too.
* EBS gives **block-level storage** (which means it stores data in blocks, like a regular disk).
* It is **persistent**, meaning your data **doesn’t disappear** if you stop or restart the EC2 instance.
* **How it works:**

1. You create an EC2 instance (a virtual server).
2. You attach an EBS volume to it (like plugging in a hard drive).
3. The EC2 instance can now **read/write data** to that volume.
4. You can **detach** the EBS volume and **attach it to another EC2** if needed.

**📦 Key Features:**

| **Feature** | **Explanation** |
| --- | --- |
| **Persistent** | Data remains even after the EC2 is stopped. |
| **Scalable** | You can increase the size anytime. |
| **Types** | EBS has different types like gp3 (general purpose), io2 (high performance), etc. |
| **Snapshots** | You can take backups (called **snapshots**) and restore them later. |
| **Encrypted** | You can encrypt your data for security. |

**Simple Analogy:**

EC2 = Computer  
EBS = Hard Drive  
Snapshot = Backup copy of the hard drive

**Common Use Cases:**

* Store application data (like a database).
* Host the root volume of an EC2 (i.e., the OS).
* Keep logs, uploads, or other persistent files.

**📸 What is a Snapshot in AWS EBS?**

Think of a **snapshot** like taking a **photo (backup)** of your hard drive (EBS volume) at a moment in time.

* It’s stored in **Amazon S3** (behind the scenes, you don’t see this directly).
* It is **incremental**, meaning:
  + The first snapshot = full backup.
  + Later snapshots = only changes since the last one (saves space and time).
* You can use a snapshot to **restore** a new EBS volume.

**✂️ What is Copy Snapshot?**

**Copy Snapshot** means making a **copy of an existing snapshot**.

**Why would you copy a snapshot?**

1. **Move it to another AWS Region** (for disaster recovery or performance).
2. **Share it** with another AWS account.
3. **Encrypt it** (you can copy and enable encryption during the copy).
4. **Create backups** in different places for safety.

**🧠 Quick Analogy:**

| **Term** | **Analogy** |
| --- | --- |
| EBS Volume | Your laptop’s hard drive |
| Snapshot | A backup photo of that hard drive |
| Copy Snapshot | Making a duplicate of that photo and keeping it in a new place or format |

**📦 What does Placement mean in AWS?**

**Placement in AWS means where and how your EC2 instances are physically placed in AWS data centers.**

**Imagine AWS is a big hotel with many rooms (servers). Placement is about which room your EC2 instance goes into and whether it's next to or far away from other instances.**

When you launch EC2 instances, you don’t always get to choose **exactly** where they run. But sometimes, you might **want control over the physical location** — for **performance, availability, or licensing reasons**.

So AWS gives you **placement options**.

**📍 1. Availability Zone (AZ) Placement**

* **Default placement** is across **Availability Zones**.
* An **AZ** is a **data center or group of them** in a region.
* Example: If you launch 3 EC2s in the same **Region** but different AZs → You get **high availability**.

**🧱 2. Placement Group**

This is where you **take control** of how instances are placed **inside** a region/AZ.

There are 3 types:

| **Type** | **What It Does** | **Use Case** |
| --- | --- | --- |
| **Cluster** | All instances are placed **very close** together in the same rack | **High performance, low latency**, e.g., HPC apps |
| **Spread** | Instances are placed **far apart** across hardware | For **high availability**, reduce risk of hardware failure |
| **Partition** | Instances are split across **separate partitions** (hardware groups) | For large **distributed systems**, like HDFS, Cassandra |

**📦 3. Dedicated Host / Dedicated Instance Placement**

* You can place your EC2 on **dedicated physical hardware** (not shared).
* Useful for:
  + Software licensing (e.g., Windows Server BYOL).
  + Compliance or isolation needs.

**🔁 Summary Table:**

| **Term** | **Simple Meaning** |
| --- | --- |
| **Availability Zone** | General placement across AWS data centers |
| **Placement Group** | Control over instance proximity |
| **Cluster** | Put instances **close together** |
| **Spread** | **Spread them apart** |
| **Partition** | Divide them into **hardware groups** |
| **Dedicated Host** | Your own **physical server** in AWS |

**📂 What is Amazon EFS?**

**Amazon EFS (Elastic File System)** is a **serverless**, **fully managed**, **scalable** **file storage** service that can be used with **Linux-based EC2 instances and other services**.

🧠 Think of it like a **shared folder** in your office that multiple computers can access at the same time — **no setup, no capacity planning, just use and grow**.

**🧱 Type of Storage: File Storage**

AWS offers 3 main types of storage:

| **Service** | **Type** | **Example Use** |
| --- | --- | --- |
| **EBS** | Block storage | One EC2 instance only (like a hard disk) |
| **S3** | Object storage | Store files, backups, images |
| **EFS** | File storage | Shared file system, like a network drive |

**🧩 Key Features of EFS**

| **Feature** | **Description** |
| --- | --- |
| 🔄 **Shared Access** | Multiple EC2 instances (even across AZs) can **mount the same EFS** |
| 📈 **Scalable** | Grows and shrinks **automatically** as you add/remove files |
| ⚙️ **Fully Managed** | AWS handles servers, maintenance, availability |
| 🧩 **POSIX-Compatible** | Works like a normal Linux file system (ls, mkdir, chmod, etc.) |
| 🔒 **Secure** | Supports encryption (at rest & in transit), IAM policies, and NFS access control |
| 🗺️ **Regional** | Accessible across all AZs in a region |
| 💸 **Pay-As-You-Use** | You only pay for the storage you use (in GB) |

**🏗️ How It Works (Architecture)**

1. You create an **EFS file system**.
2. You **mount it** on one or more **Linux EC2 instances** using the **NFS protocol**.
3. EC2s can now **read/write to EFS like a local folder** (/mnt/efs).
4. As files are added/removed, EFS scales **automatically**.

**🔄 EFS Performance Modes**

| **Mode** | **Description** | **Use Case** |
| --- | --- | --- |
| **General Purpose** (default) | Low latency, good for most apps | Web servers, content management |
| **Max I/O** | Higher throughput, slightly more latency | Big data, analytics, large-scale apps |

**💰 EFS Storage Classes (to Save Cost)**

| **Class** | **Description** | **Cost** |
| --- | --- | --- |
| **Standard** | Frequent access | Higher |
| **Infrequent Access (IA)** | For files not often accessed | Lower (good for backups/archive) |

You can **enable Lifecycle Management** to move old files automatically to IA.

**✅ Use Cases for EFS**

* Shared file storage for **web applications**
* Centralized **configuration files**
* Shared **logs or backups**
* **Machine learning** or **data analytics**
* **Content management systems** (CMS)

**🛠️ Steps to Use EFS (Basic Workflow)**

1. Go to AWS Console > EFS > Create File System.
2. Choose VPC, security group, mount targets (for each AZ).
3. Launch EC2 instances (Linux).
4. Use the EFS mount helper to mount it:

bash

CopyEdit

**sudo mount -t efs fs-xxxx:/ /mnt/efs**

1. Done! Now EC2 can read/write files to the shared EFS.

**🔐 Security in EFS**

* IAM policies (who can create or access EFS)
* **Security groups** (controls network access)
* **Encryption at rest** using KMS
* **Encryption in transit** using TLS (via mount helper)

**🪣 What is Amazon S3?**

**Amazon S3** is a **cloud-based object storage service** where you can **store and retrieve any amount of data** — like files, images, videos, backups, documents, and more.

🧠 Think of S3 like an **infinite USB drive in the cloud**, accessible from anywhere.

**🗂️ How is S3 organized?**

S3 uses a simple structure:

* **Bucket** = Like a **folder** that stores your data.
* **Object** = The **actual file/data** you upload (e.g., photo, PDF, ZIP file).
* **Key** = The **name** of the object (like the file path).

Example:

| **Component** | **Example** |
| --- | --- |
| Bucket | my-photo-storage |
| Object | vacation/beach.jpg |
| Key | vacation/beach.jpg |

**🧩 Key Features**

| **Feature** | **Description** |
| --- | --- |
| ☁️ **Unlimited Storage** | Store as much data as you want |
| 🌎 **Accessible Globally** | From anywhere via internet |
| 📂 **Object Storage** | Stores files, not block devices |
| 🛡️ **Secure** | Supports encryption, IAM policies, bucket policies |
| 🔄 **Versioning** | Keeps **multiple versions** of the same object |
| 🗑️ **Lifecycle Rules** | Automatically move or delete files after some time |
| 📁 **Folders (Pseudo)** | Uses object key names to simulate folders |
| 🔗 **Static Website Hosting** | Host websites directly from S3 |
| 🧾 **Logging** | Tracks access and actions for security/auditing |
| 🔐 **Access Control** | IAM, bucket policies, ACLs, signed URLs |

**🛠 Common Use Cases**

* 📸 Store media files (images, videos, audio)
* 🗄️ Backup and archive data
* 🌐 Host static websites
* 📤 Upload/download user files (e.g., resumes, invoices)
* 📦 Store logs or analytics data
* 🤖 Input/output for data processing (Lambda, ML jobs, etc.)

**🧪 Storage Classes (Cost Saving Options)**

S3 offers **different storage classes** based on how often you access the data:

| **Class** | **Use Case** | **Cost** |
| --- | --- | --- |
| **Standard** | Frequent access | 💲💲 |
| **Intelligent-Tiering** | Auto move between classes | 💲+ |
| **Standard-IA** | Infrequent access | 💲 |
| **One Zone-IA** | Infrequent, one AZ only | 💲 |
| **Glacier** | Archive, minutes to hours to access | 💰 (cheap) |
| **Glacier Deep Archive** | Long-term archive, hours to access | 💰💰 (very cheap) |

**🔐 S3 Security**

* **IAM** (who can do what)
* **Bucket Policies** (rules at bucket level)
* **ACLs** (access control lists per object)
* **Encryption**:
  + **SSE-S3** (managed by AWS)
  + **SSE-KMS** (you manage keys)
* **Public Access Settings** (block public access by default)

**🔄 Versioning**

* Keeps multiple versions of the same object.
* Helps you **recover from accidental deletion or overwrite**.

**📉 Lifecycle Rules**

* Move files to cheaper storage (e.g., Glacier) after 30 days.
* Delete old versions after 90 days.
* **Automatic cost savings**.

**📊 S3 vs EBS vs EFS**

| **Feature** | **S3** | **EBS** | **EFS** |
| --- | --- | --- | --- |
| Storage Type | Object | Block | File |
| Use Case | Files, backups, websites | One EC2's disk | Shared file system |
| Access | API (HTTPS) | OS-level | Mount |
| Multi-instance Access | Yes | No | Yes |
| Scalability | Auto | Manual | Auto |

**🧠 Analogy**

Imagine S3 as **Google Drive** for your applications.  
You can **store, share, secure**, and **access files** from anywhere — with **infinite space** and **fine-grained control**.

**📥 Example: Uploading to S3 (Console)**

1. Go to AWS Console → S3
2. Create a **bucket** (e.g., my-app-files)
3. Upload files (drag-and-drop or browse)
4. Set permissions (public/private)
5. Done! You can access the file via a URL if allowed.

**🪣 What Are S3 Storage Classes?**

S3 has **multiple storage classes** to help you **store data at the right cost** based on **how often** and **how quickly** you need to access it.

Think of it like choosing between a **fast locker**, a **cheap archive**, or a **smart automatic storage** — all in the cloud.

**📊 S3 Storage Classes Comparison**

| **Storage Class** | **Best For** | **Access Speed** | **Durability** | **Availability** | **Min Storage Time** | **Use Case** |
| --- | --- | --- | --- | --- | --- | --- |
| **S3 Standard** | Frequently accessed data | Milliseconds | 99.999999999% (11 9s) | 99.99% | None | Websites, apps, hot data |
| **S3 Intelligent-Tiering** | Data with unknown access patterns | Milliseconds | 11 9s | 99.9%–99.99% | 30/90 days (some tiers) | Logs, unpredictable data |
| **S3 Standard-IA** (Infrequent Access) | Infrequently accessed, but needed fast | Milliseconds | 11 9s | 99.9% | 30 days | Backups, older docs |
| **S3 One Zone-IA** | Infrequent access, low-cost | Milliseconds | 11 9s | 99.5% | 30 days | Re-creatable data (e.g. cache) |
| **S3 Glacier** | Archival storage (cheap) | Minutes to hours | 11 9s | Varies | 90 days | Compliance, long-term backup |
| **S3 Glacier Deep Archive** | Long-term cold archive | Hours | 11 9s | Varies | 180 days | Tape replacement, archive only |

**💡 Quick Summary Table**

| **Class** | **Access** | **Cost** | **Use** |
| --- | --- | --- | --- |
| 🔥 **Standard** | Frequent | $$$ | Main apps, daily use |
| 🤖 **Intelligent-Tiering** | Auto-manages | $$ | When you don’t know access pattern |
| ❄️ **Standard-IA** | Infrequent | $ | Quick access to old data |
| 🧊 **One Zone-IA** | Infrequent, 1 AZ | $ (cheaper) | Recoverable data |
| 🏔️ **Glacier** | Archive | $$ (low) | Long-term storage |
| 🕳️ **Glacier Deep Archive** | Very cold | $ (very low) | Compliance, rarely accessed |

**🔄 Intelligent-Tiering: How it Works**

* Starts in **frequent access tier**
* Moves data automatically to:
  + **Infrequent Access Tier**
  + **Archive/Deep Archive Tier** (optional)
* No performance loss
* Small monthly monitoring fee (~$0.0025 per object)

**🌐 What is a VPC?**

**Amazon VPC (Virtual Private Cloud)** is like your own **private network inside AWS** — where you can **launch and control** resources (like EC2, RDS) securely.

🧠 Think of it like building your own data center inside AWS — with full control over networking.

**🧩 Key VPC Components**

Here are the **main building blocks** of a VPC:

**1. 🧱 VPC**

* A **virtual network** in AWS.
* You define its **IP range** using a **CIDR block** (e.g., 10.0.0.0/16).
* You launch AWS resources (like EC2s) inside it.

**2. 📦 Subnets**

* Smaller networks **inside a VPC**.
* Two types:
  + **Public Subnet** – connected to the internet.
  + **Private Subnet** – isolated from the internet.
* Subnets are **AZ-specific**.

🧠 Think of subnets as rooms in a building (your VPC).

**3. 🌍 Internet Gateway (IGW)**

* Lets instances in **public subnets** access the **internet**.
* Must be **attached to your VPC**.
* Public EC2s need:
  + Public IP
  + Route to IGW
  + IGW attached

**4. 🚪 NAT Gateway / NAT Instance**

* Lets **private instances** access the **internet** (e.g., for updates) **without exposing them** to inbound internet traffic.
* Placed in **public subnet**.
* NAT = **Network Address Translation**.

**5. 📜 Route Tables**

* Control **where traffic goes** inside the VPC.
* Subnets are associated with route tables.
* Used to send traffic:
  + Within VPC
  + To IGW (internet)
  + To NAT
  + To VPN/peering

**6. 🔐 Security Groups**

* Like a **firewall** for EC2 instances.
* Controls **inbound and outbound traffic**.
* Instance-level (not subnet-level).
* **Stateful** – response traffic is allowed automatically.

**7. 🧱 Network ACLs (NACLs)**

* Optional **subnet-level firewall**.
* Controls **traffic in and out of subnets**.
* **Stateless** – must allow both inbound and outbound explicitly.
* Good for extra security.

**8. 🧭 DHCP Options Set**

* Controls **DNS and domain name settings** inside the VPC.
* By default, AWS provides its own DNS.

**9. 🛣️ VPC Peering**

* Connects two VPCs (same or different AWS accounts) **privately**.
* Allows **resources in each VPC to talk** to each other.

**10. 🔐 VPN Gateway & Customer Gateway**

* Used to create a **secure VPN connection** between AWS VPC and your **on-premises network**.
* **VPN Gateway** = on AWS side
* **Customer Gateway** = your company’s router/firewall

**11. 🛰️ Transit Gateway**

* Connects **multiple VPCs and on-prem networks** through a **central hub**.
* Scalable alternative to many VPC peering connections.

**✅ What is AWS Trusted Advisor?**

**AWS Trusted Advisor** is like a **smart assistant** for your AWS account.  
It helps you **optimize performance, security, reliability, and cost** by **analyzing your AWS environment** and giving **recommendations**.