**AWS Block Storage Explained: Step-by-Step Guide with Practical Scenarios**

Introduction to AWS Storage Types

AWS offers three main types of storage solutions:

1. **Block Storage** (Today's focus)
2. **File Storage**
3. **Object Storage**

Block Storage Components:

* **Instance Storage** (Temporary)
* **EBS Volumes** (Persistent)

File Storage Options:

* **EFS** (Elastic File System) for Linux
* **FSx** for Windows

Object Storage Options:

* **S3** (Simple Storage Service) for unstructured data
* **Glacier** for archival data

Deep Dive into Block Storage

1. Instance Storage

**Characteristics:**

* Comes by default with EC2 instances (typically 8GB in /root)
* **Ephemeral nature**: Data is lost when you stop/terminate the instance
* Fixed size and type (depends on instance type like T2 micro, T2 nano, etc.)
* Cannot be detached and attached to other instances

**Use Case**: Temporary cache or buffers where data persistence isn't required

2. EBS Volumes (Elastic Block Store)

**Advantages:**

* **Persistent storage**: Data remains after restart/stop
* Scalable up to 16TB
* Available in two types:
  + **HDD** (Hard Disk Drive): SC1, ST1 standards
  + **SSD** (Solid State Drive):
    - General Purpose (gp2, gp3)
    - Provisioned IOPS (io1, io2) for high-performance databases
* Can be detached and attached to other instances (within the same AZ)

**Limitation**:

* Availability Zone dependent (can't attach to instances in different AZs)

Practical Demonstration

Step 1: Launch an EC2 Instance

1. Go to EC2 Dashboard → Launch Instance
2. Select instance type (e.g., t2.large)
3. Configure storage:
   * Default root volume (instance store)
   * Add additional volumes (4GB and 2GB)
4. Launch the instance

Step 2: Connect to the Instance

1. Copy public IP
2. Connect via SSH:

bash

Copy

Download

ssh -i your-key.pem ec2-user@your-instance-ip

Step 3: Prepare the Volumes

1. Check available block devices:

bash

lsblk

(Should show xvdb - 4GB and xvdc - 2GB)

1. Create mount points:

bash

mkdir -p /i\_volume /eb\_volume

Step 4: Format and Mount Volumes

1. Create partitions:

bash

fdisk /dev/xvdb

(Create primary partition, save changes)

Repeat for xvdc:

bash

fdisk /dev/xvdc

1. Create filesystems:

bash

mkfs.ext4 /dev/xvdb1

mkfs.ext4 /dev/xvdc1

1. Mount volumes:

bash

mount /dev/xvdb1 /i\_volume

mount /dev/xvdc1 /eb\_volume

Step 5: Configure Persistent Mount (fstab)

1. Edit /etc/fstab:

bash

nano /etc/fstab

1. Add only the 2GB volume for persistence:

/dev/xvdc1 /eb\_volume ext4 defaults 0 1

(Note: We intentionally omit the 4GB volume)

Step 6: Test with Files

1. Download test files to both volumes:

bash

cd /i\_volume

wget https://example.com/terraform.zip

cd /eb\_volume

wget https://example.com/terraform.zip

Step 7: Reboot and Verify

1. Reboot the instance:

bash

sudo reboot

1. After reboot, check volumes:

bash

lsblk

* + The 4GB volume (/i\_volume) won't be mounted (not in fstab)
  + The 2GB volume (/eb\_volume) remains mounted with files intact

Real-World Use Cases

Instance Storage Scenario:

**Temporary Processing Cache**: A video processing application uses instance storage as temporary space while transcoding videos. After processing completes, the final output is saved to S3. The temporary files can be safely lost when the instance stops.

EBS Volume Scenario:

**Database Storage**: A company runs a MySQL database on EC2 with an attached io1 EBS volume for high IOPS. Even if the instance crashes, the data persists on the EBS volume. They can:

1. Detach the volume from the failed instance
2. Attach it to a new instance in the same AZ
3. Continue operations with all data intact

Multi-Volume Scenario:

**Web Application**: A WordPress site uses:

* Root volume for the OS
* 50GB gp3 volume for application files (/var/www/html)
* 100GB io2 volume for the MySQL database
* All volumes are regularly backed up via EBS snapshots

Key Takeaways

1. **Instance Storage** is temporary but provides high-speed local storage
2. **EBS Volumes** offer persistence and flexibility but are AZ-bound
3. **fstab configuration** is crucial for ensuring volumes mount automatically
4. **Volume types** should match workload requirements (gp3 for general use, io1/io2 for databases)
5. **Always back up** important data, even on EBS volumes, using snapshots

This comprehensive understanding of AWS block storage helps in making informed decisions about data persistence, performance, and cost optimization in cloud architectures.