**EC2 STORAGE SECTION**

**EBS VOLUME [ELASTIC BLOCK STORE]**

Amazon Elastic Block Store (Amazon EBS) provides block level storage volumes for use with EC2 instances. EBS volumes behave like raw, unformatted block devices. You can mount these volumes as devices on your instances. EBS volumes that are attached to an instance are exposed as storage volumes that persist independently from the life of the instance. You can create a file system on top of these volumes, or use them in any way you would use a block device (such as a hard drive). You can dynamically change the configuration of a volume attached to an instance.

We recommend Amazon EBS for data that must be quickly accessible and requires long-term persistence.

EBS volumes are particularly well-suited for use as the primary storage for file systems, databases, or for any applications that require fine granular updates and access to raw, unformatted, block-level storage.

**What’s an EBS Volume?**

* An EC2 machine loses its root volume (main drive) when it is manually terminated.
* Unexpected terminations might happen from time to time (AWS would email you)
* Sometimes, you need a way to store your instance data somewhere
* An EBS (Elastic Block Store) Volume is a network drive you can attach to your instances while they run
* It allows your instances to persist data

**EBS Volume**

Pon5428962

9100922182

* It’s a network drive (i.e. not a physical drive)
  + It uses the network to communicate the instance, which means there might be a bit of latency
  + It can be detached from an EC2 instance and attached to another one quickly
* It’s locked to an Availability Zone (AZ)
* An EBS Volume in us-east-1a cannot be attached to us-east-1b
* To move a volume across, you first need to snapshot it
* Have a provisioned capacity (size in GBs, and IOPS)
* You get billed for all the provisioned capacity
* You can increase the capacity of the drive over time

**EBS Volume Types**

**EBS Volumes come in 4 types**

* GP2 (SSD): General purpose SSD volume that balances price and performance for a wide variety of workloads
* IO1 (SSD): Highest-performance SSD volume for mission-critical low-latency or high throughput workloads
* ST1 (HDD): Low cost HDD volume designed for frequently accessed, throughput intensive workloads
* SC1 (HDD): Lowest cost HDD volume designed for less frequently accessed workloads.

**EBS Volumes are characterized in Size | Throughput | IOPS (I/O Ops per Sec)**

* When in doubt always consult the AWS documentation – it’s good!
* Only GP2 and IO1 can be used as boot volumes

**EBS Volume Types Use cases**

**[GP2 (from AWS doc) ]**

1. Recommended for most workloads
2. System boot volumes
3. Virtual desktops
4. Low-latency interactive apps
5. Development and test environments
6. 1 GiB - 16 TiB
7. Small gp2 volumes can burst IOPS to 3000
8. Max IOPS is 16,000… min 100
9. 3 IOPS per GB, means at 5,334GB we are at the max IOPS

**[IO1 (from AWS doc)]**

1. Critical business applications that require sustained IOPS performance, or
   1. more than 16,000 IOPS per volume (gp2 limit)
2. Large database workloads, such as:
   1. MongoDB, Cassandra, Microsoft SQL Server, MySQL, PostgreSQL, Oracle
3. 4 GiB - 16 TiB
4. IOPS is provisioned (PIOPS) – MIN 100 - MAX 64,000 (Nitro instances) else MAX 32,000 (other instances)
5. The maximum ratio of provisioned IOPS to requested volume size (in GiB) is 50:1

**[ST1 (from AWS doc)]**

1. Streaming workloads requiring consistent, fast throughput at a low price.
2. Big data, Data warehouses, Log processing
3. Apache Kafka
4. Cannot be a boot volume
5. 500 GiB - 16 TiB
6. Max IOPS is 500
7. Max throughput of 500 MiB/s – can burst

**[SC1 (from AWS doc)] [Cold HDD]**

1. Throughput-oriented storage for large volumes of data that is
2. infrequently accessed
3. Scenarios where the lowest storage cost is important
4. Cannot be a boot volume
5. 500 GiB - 16 TiB
6. Max IOPS is 250
7. Max throughput of 250 MiB/s – can burst

**EBS – Volume Types Summary**

1. gp2: General Purpose Volumes (cheap)
2. 3 IOPS / GiB, minimum 100 IOPS, burst to 3000 IOPS, max 16000 IOPS
3. 1 GiB – 16 TiB , +1 TB = +3000 IOPS
4. io1: Provisioned IOPS (expensive)
5. Min 100 IOPS, Max 64000 IOPS (Nitro) or 32000 (other)
6. 4 GiB - 16 TiB. Size of volume and IOPS are independent
7. st1: Throughput Optimized HDD
8. 500 GiB – 16 TiB , 500 MiB /s throughput
9. sc1: Cold HDD, Infrequently accessed data
10. 500 GiB – 16 TiB , 250 MiB /s throughput

**Making an Amazon EBS volume available for use on Linux**

* After you attach an Amazon EBS volume to your instance, it is exposed as a block device.
* You can format the volume with any file system and then mount it.
* After you make the EBS volume available for use, you can access it in the same ways that you access any other volume. Any data written to this file system is written to the EBS volume and is transparent to applications using the device.

**To format and mount an EBS volume on Linux**

1. Connect to your instance using SSH
2. The device could be attached to the instance with a different device name than you specified in the block device mapping.
3. Use the lsblk command to view your available disk devices and their mount points (if applicable) to help you determine the correct device name to use.
4. Determine whether there is a file system on the volume. New volumes are raw block devices, and you must create a file system on them before you can mount and use them.

#sudo file -s /dev/xvdf

1. If you discovered that there is a file system on the device in the previous step, skip this step

#sudo mkfs -t xfs /dev/xvdf

Do not use this command if you're mounting a volume that already has data on it (for example, a volume that was created from a snapshot). Otherwise, you'll format the volume and delete the existing data.

1. If you get an error that mkfs.xfs is not found, use the following command to install the XFS tools and then repeat the previous command:

#sudo yum install xfsprogs

1. Use the mkdir command to create a mount point directory for the volume

#sudo mkdir /data

1. mount the volume at the directory you created

#sudo mount /dev/xvdf /data

1. Review the file permissions of your new volume mount to make sure that your users and applications can write to the volume

## **Automatically mount an attached volume after reboot**

1. Create a backup of your /etc/fstab file that you can use if you accidentally destroy or delete this file while editing it.

#sudo cp /etc/fstab /etc/fstab.orig

1. Use the blkid command to find the UUID of the device.

#sudo blkid

1. Open the /etc/fstab file using any text editor, such as nano or vim

#sudo vim /etc/fstab

1. Add the following entry to /etc/fstab to mount the device at the specified mount point.

UUID=aebf131c-6957-451e-8d34-ec978d9581ae /data xfs defaults,nofail 0 2

1. To verify that your entry works, run the following commands to unmount the device and then mount all file systems in /etc/fstab

#sudo umount /data

#sudo mount -a

**EBS vs Instance Store**

* Some instance do not come with Root EBS volumes
* Instead, they come with “Instance Store” (= ephemeral storage)
* Instance store is physically attached to the machine (EBS is a network drive)

**Pros:**

* Better I/O performance on no network (EBS gp2 has an max IOPS of 16000, io1 of 64000)
* Good for buffer / cache / scratch data / temporary content
* Data survives reboots

**Cons:**

* On stop or termination, the instance store is lost
* You can’t resize the instance store
* Backups must be operated by the user

**Local EC2 Instance Store [Ephimeral]**

* Physical disk attached to the physical server where your EC2 is.
* Very High IOPS (because physical)
* Disks up to 7.5 TiB (can change over time), stripped to reach 30 TiB (can change over time…)
* Block Storage (just like EBS)
* Cannot be increased in size
* Risk of data loss if hardware fails

**EBS Snapshots**

* Incremental : only backup changed blocks
* EBS Backups use IO and you shldn’t run them while your application is handling lot of traffic.
* Snapshots will be stored in s3 [but you cant directly see them] but u get billed for s3 usage for EBS backups or snapshots.
* Not necessary to detach volume to do snapshot but recommended.
* Max 100,000 snapshots per Account
* Can copy the snapshots across AZ or region
* Can create AMI from snapshot
* EBS volumes restored by snapshots need to be pre-warned [using fio or dd command to read the entire volume]
* Snapshots can be automated using Amazon Data Lifecycle Manager

**EBS Migration**

EBS volumes are locked to a specific AZ

To migrate it to a dif AZ pr region:

you need to create snapshot

[optional] copy the snapshot toa dif region

Create a volume from the snapshot in the AZ of your choice.

**EBS Encryption**

* When you create an encrypted EBS volume you get the following:
  + Data at rest is encrypted inside the volume
  + All the data in flight moving bet the instances and the volume is encrp
  + All snapshots are encrypted
  + All volumes created from the snapshot is encrypted
  + Encypt has minimal impact on latencty
  + Copying unencrypted snapshots allows encryption
  + Snapshots of encrypted volumes are encrypted

**Encrypt an unencrypted EBS volume**

* + Create an EBS snapshot
  + Encrypt the EBS snapshot using copy
  + Create new EBS volume from the snapshot
  + Now you can attach the encrp volume to the original instance

**EBS RAID Options**

* EBS already reduntant storage [replicated within an AZ]
* But if you Want to increase IOPS to say 100 000 IOPS
* What if you want ot mirror your EBS volumes?
* You would mount volumes in paraalel in RAID settings.
* RAID is possible as long as your OS supports it
* Some RAID options are:
  + RAID 0
  + RAID 1
  + RAID 5 [not recommended]
  + RAID 6 [not recommended]

**EBS vs EFS – Elastic File System**

EBS volumes:

* can be attached to only one instance at a time
* are locked at the Availability Zone (AZ) level
* gp2: IO increases if the disk size increases
* io1: can increase IO independently

To migrate an EBS volume across AZ

* Take a snapshot
* Restore the snapshot to another AZ
* EBS snapshots require EBS backups which uses IO and you shouldn’t run them while your application is actively using EBS volume else you get performance issue.

Root EBS Volumes of instances get terminated by default if the EC2 instance [You can disable that]

* Mounting 100s of instances across AZ
* EFS share website files (WordPress)
* Only for Linux Instances (POSIX)
* EFS has a higher price point than EBS
* Can leverage EFS-IA for cost savings