## Darshan Nikam Date: 27/02/2024

### **Elastic Block Storage**

Amazon EBS, it provides block-level storage volumes that can be attached to EC2 instances. This means that EBS volumes offer direct access to individual blocks of data, allowing EC2 instances to read and write data to the volume at a low level, similar to accessing a physical hard drive. This direct access enables a wide range of storage use cases and allows for efficient management of data within the AWS cloud environment.

|  |
| --- |
| Features of EBS   * Elasticity: Easily resize volumes or change types to meet changing storage needs. * Data Persistence: Data stored on EBS volumes persists even if the associated EC2 instance is terminated. * Snapshots: Create point-in-time backups of volumes for data protection and disaster recovery. * High Availability: Designed for high availability and durability with replication across multiple Availability Zones. * Encryption: Supports encryption at rest to secure data stored on EBS volumes. |

Amazon EBS provides the following volume types:

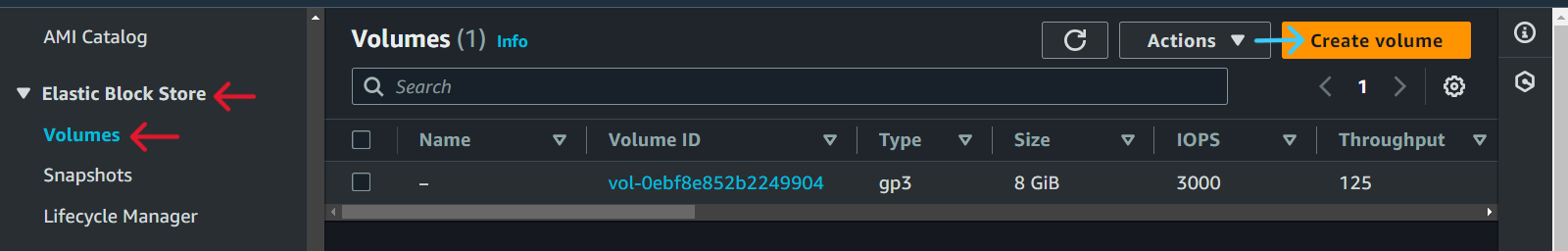
* **General Purpose SSD (gp2 and gp3)**
* **Provisioned IOPS SSD (io1 and io2) [Single EBS attach to up to 16 nitro-based Ec2 Instance]**
* **Throughput Optimized HDD (st1), Cold HDD (sc1)**
* **Magnetic (standard).**

They differ in performance characteristics and price, allowing you to tailor your storage performance and cost to the needs of your applications.

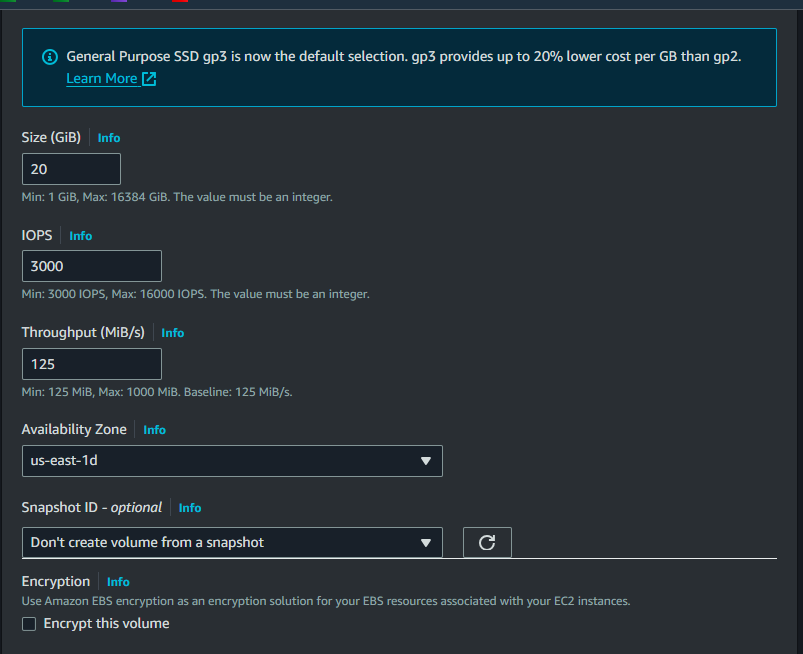
You can attach multiple EBS volumes to a single instance. The volume and instance must be in the same Availability Zone. Depending on the volume and instance types

**Mount EBS Volume to EC2 Instance**

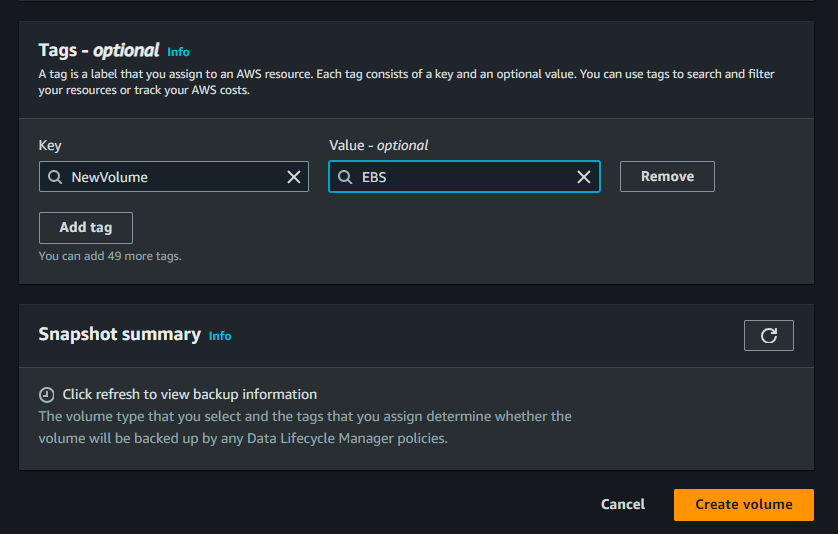
Step 1) Go EC2 dashboard and Click on the Volume option in the Elastic Block Store Menu. Then Click on the Create Volume Button.



Step 2) Now Configure your volume details like volume size, IPOS, Throughput, and availability zone, and you have the option to create a volume from a snapshot or not (Using a snapshot to create a volume has many use cases like data recovery, data replication, data migration between two deferent Instance or regions, Scaling storage, and testing & development).

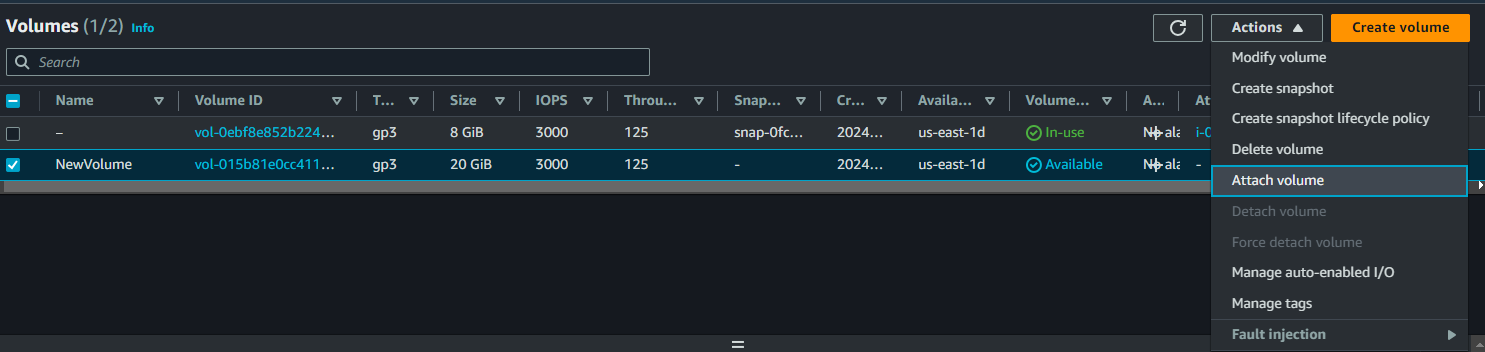


Step 3) Then simply add the tag (is optional), and click on the create volume button.

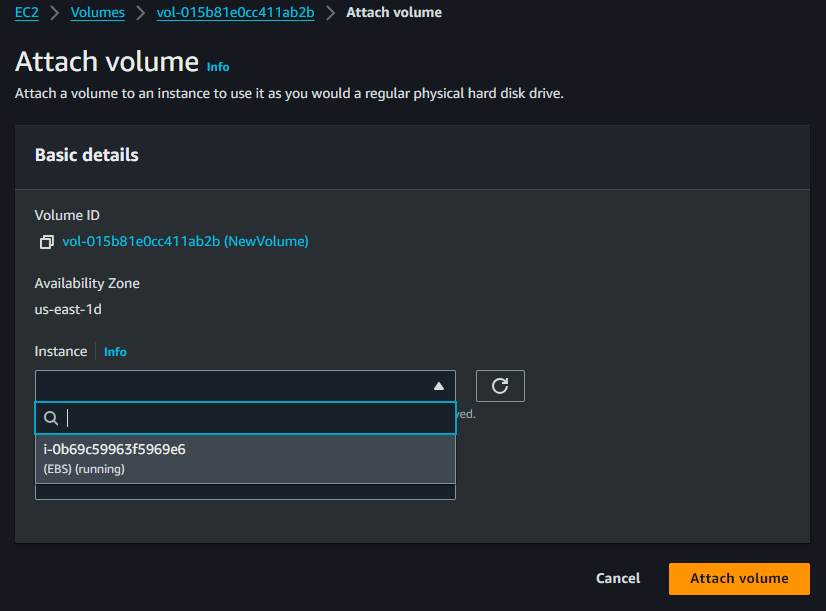


Now your volume is created. So we attach this volume to the Instance.

Step 4) Click on the Action Button and select the Attach volume option.



Step 5) Select your Instance ID where you want to attach this volume and click on the Attach volume button.



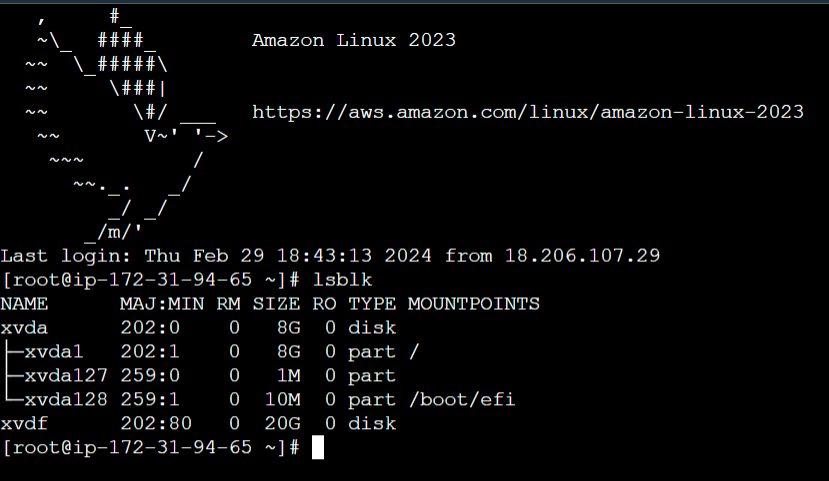
Now your volume is attached to your instance but this volume is not accessible, we need to initialize the volume which means formatting the volume with the file system (same as when we upgrade our PC's SSD) then we create a partition of volume (like our PC’s C drive, D drive, and E drive) after that we need to mount the volume to a directory within the instance file system.

Why we need this: - This makes the contents of the volume accessible to the operating system and any applications running on the instance.

So we need to access or connect to our Ec2 instance using SSH and run the following commands to mount the volume to the Ec2 instance.

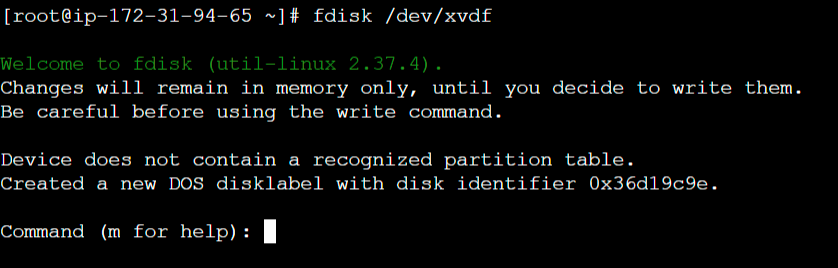
**Commands:-**

* **lsbkl -> to list all block information and partitions**

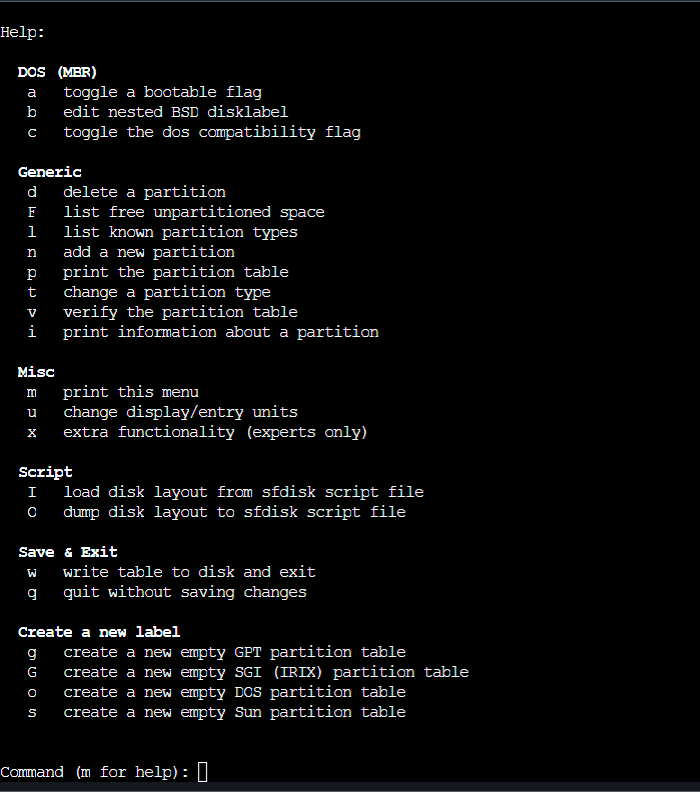


Now we see our new volume **xvdf** attached but it does not have a partition like **xvda** or no files system mounted.

* **fdisk /dev/xvdf ->** to make a partition of volume



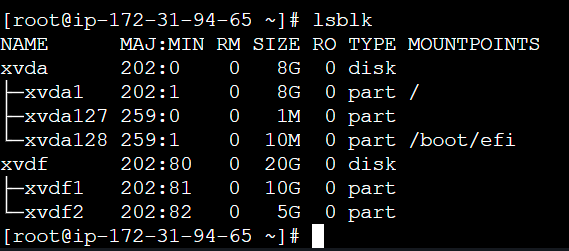
* **type m for help ->**  all options list will open with description.



* **type n for the new partition**
* **Then select partition type, by default on the primary just press Enter**

We have up to **four primary partitions**, or Up to **three primary partitions and one extended partition**, **then choose the partition number**

* **The option for partition starts from the first sector, simply press Enter, and proceed with the last sector.**
* **Type +10G and enter -> means create a 10 GB partition of disk and by default start from the last sector.**
* **Type w and press Enter for save and exit**
* Run the **lsbkl** command to check the partition.



**Now you will see new partitions xvdf1 and xvdf2 that are successfully created.**

**Now we need to create the filesystem on that partition to store the files. Without creating a filesystem we can’t use those partitions.**

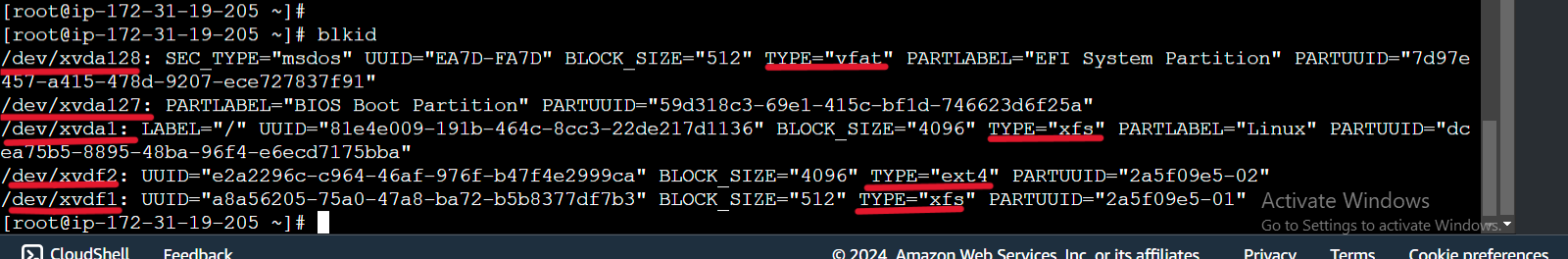
|  |
| --- |
| **Linux File system types**  Linux supports various file system types, each designed for different purposes. Some of the common file system types used in Linux are   1. **ext4:** This is the default file system for many Linux distributions. It is a journaling file system and is backward compatible with its   predecessors ext3 and ext2. ext4 offers improvements in performance, scalability, and reliability compared to ext3.   1. **XFS:** Known for its high performance and scalability, XFS is particularly well-suited for large-scale deployments and environments with   heavy I/O workloads. It supports features like journaling, extended attributes, and large file systems.   1. **Btrfs (B-tree file system):** Btrfs is a modern copy-on-write (COW) file system that offers features such as snapshots, subvolumes, and   built-in RAID support. It aims to provide better data integrity, scalability, and easier administration. While it's considered stable, it's still  under active development.   1. **ZFS:** Although originally developed for Solaris, ZFS is available on Linux through projects like OpenZFS. ZFS is feature-rich, offering data   integrity through checksums, built-in RAID, snapshots, and easy volume management. However, due to licensing issues,  it's not I included in many Linux distributions by default.   1. **F2FS (Flash-Friendly File System):** Optimized for NAND flash storage devices like SSDs and eMMC, F2FS is designed to improve   performance and extend the lifespan of flash-based storage. It employs techniques such as wear leveling and TRIM support.   1. **NTFS:** While primarily associated with Windows, Linux has NTFS support through the NTFS-3G driver. This allows Linux systems to read   and write to NTFS-formatted drives, enabling interoperability with Windows systems.   1. **VFAT:** Another file system commonly associated with Windows, VFAT (Virtual File Allocation Table) is used for compatibility with older   Windows systems and removable storage devices like USB drives and SD cards. Linux supports VFAT for read and write operations.   1. **ISO 9660:** This is the standard file system used for CD-ROMs and DVD-ROMs. It provides a read-only file system suitable for distributing   software and data on optical media. |

**To create a filesystem run the following commands:**

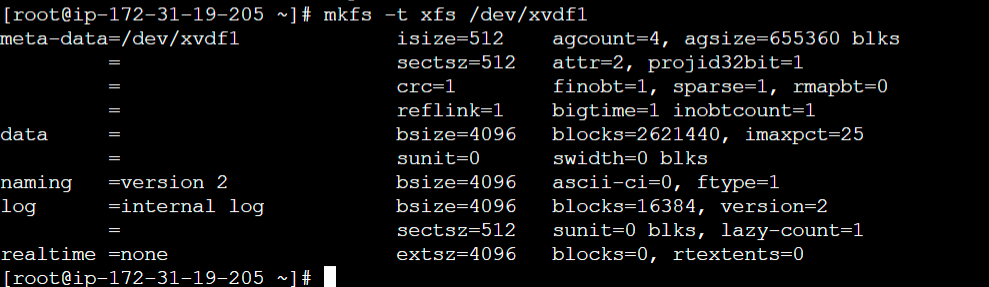
* **mkfs –t xfs /dev/xvdf1 -> mkfs –t xfs - used to create a xfs filesystem**
* **mkfs –t ext4 /dev/xvdf2 -> mkfs –t ext4 - used to create a ext4 filesystem**

**(As using same command mkfs –t [file type] /dev/sda you create other types of the filesystem )**

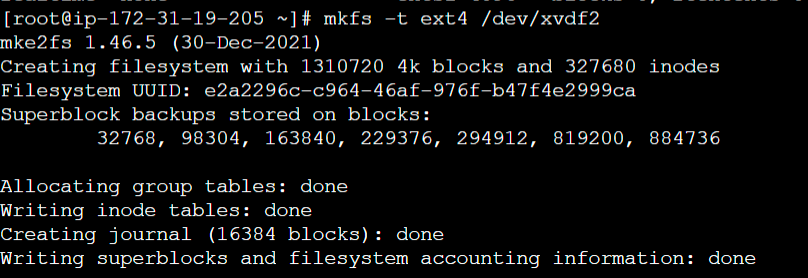
* **blkid -> to check all blocks filesystem type and other information**



* **xfs file system created**

****

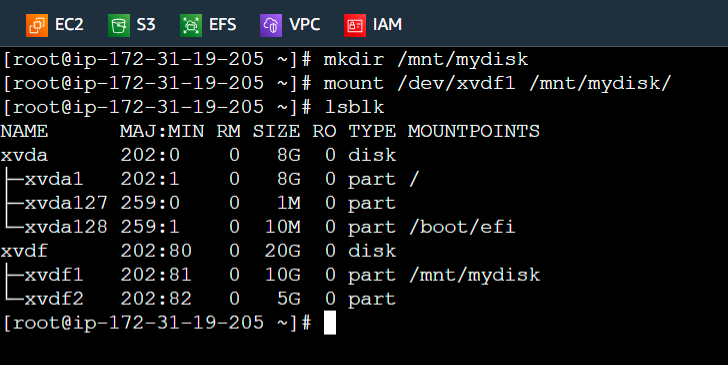
* **ext4 file system created**

****

**After the file system is created, we want to mount it to the directory in our file system so we can access it. So first we need to create a mount point in the [ /mnt ] directory this is the temporary mount point. (We can mount the filesystem in any other directory)**

* **mkdir /mnt/mydisk1**
* **mount /dev/xvdf1 /mnt/mydisk1**

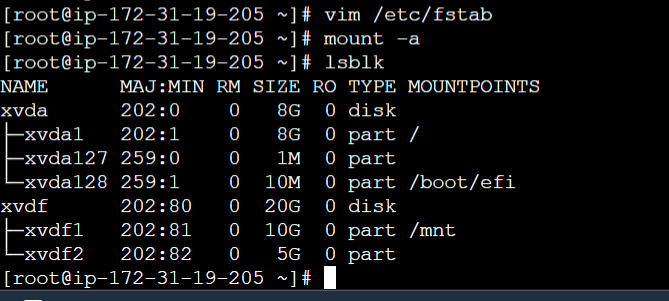
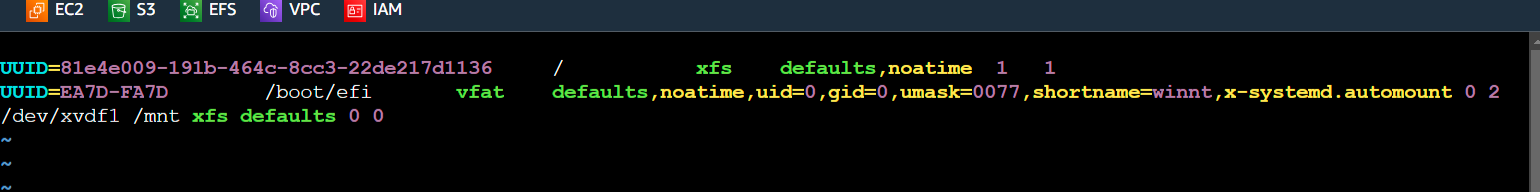
**Now, you will see your filesystem is temporarily mounted and available to use.**



* **umount /dev/mnt/mydisk -> to unmount the volume**

**To permanently mount the volume to the directory we need to add an entry in “/etc/fstab” file.**

* **vim /etc/fstab**
* **add entry [/dev/xvdf1 /mnt xfs default 0 0] in the file and save the file.**
* **mount –a -> to mount all file systems we listed.**
* **lsbkl -> to check file is mounted.**



**Here we can see Volume is permanently mounted to /mnt**

**\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**Single EBS Volume attaches to multiple instances.**

**Amazon EBS Multi-Attach that allows you to share a single EBS volume with up to 16 Nitro-based EC2 instances in the same Availability Zone. This can be useful for specific use cases like sharing configuration files or specific datasets across a cluster of instances. But it's important to note that Multi-Attach only works with Provisioned SSD volume io1 or io2 EBS volume types, designed for high-performance workloads.**

**Provisioned IOPS SSD (io1)**

**Clustering or Shared Storage: Use clustered file systems like GFS2 or OCFS2 to provide concurrent access to shared storage.**

**Attach the io1 volume to one instance and share it across others using the clustered file system.**

**Mounting Considerations: Mount the volume as read-only on all but one instance to avoid concurrency issues.**

**If simultaneous writes are needed, implement a distributed locking mechanism or use a clustered file system.**

**Replica: Replicate data across multiple io1 volumes attached to different instances.**

**Use AWS DataSync or third-party tools for synchronization.**

**Performance and Latency: Sharing may impact performance predictability due to increased latency and contention for I/O resources. Monitor performance metrics closely and adjust volume configuration as needed.**

**Provisioned IOPS SSD (io2):**

**Improved Performance: io2 volumes offer higher performance and better I/O consistency compared to io1.**

**Shared access may have less impact on performance due to improved throughput and IOPS capabilities.**

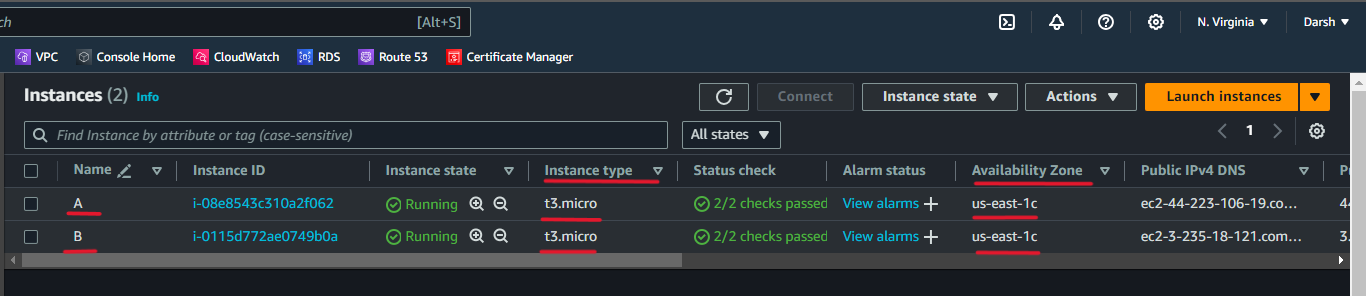
**Optimized for Critical Workloads: Suitable for mission-critical applications and high-performance databases requiring low-latency I/O.**

**Cost Considerations: io2 volumes are priced higher than io1 due to improved performance.**

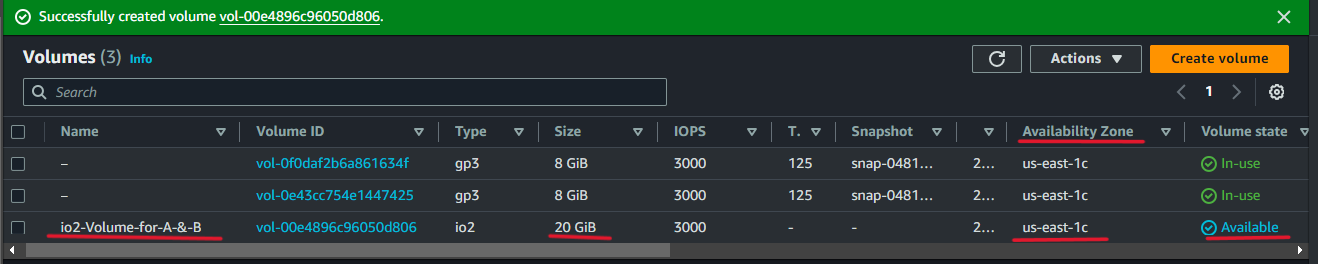
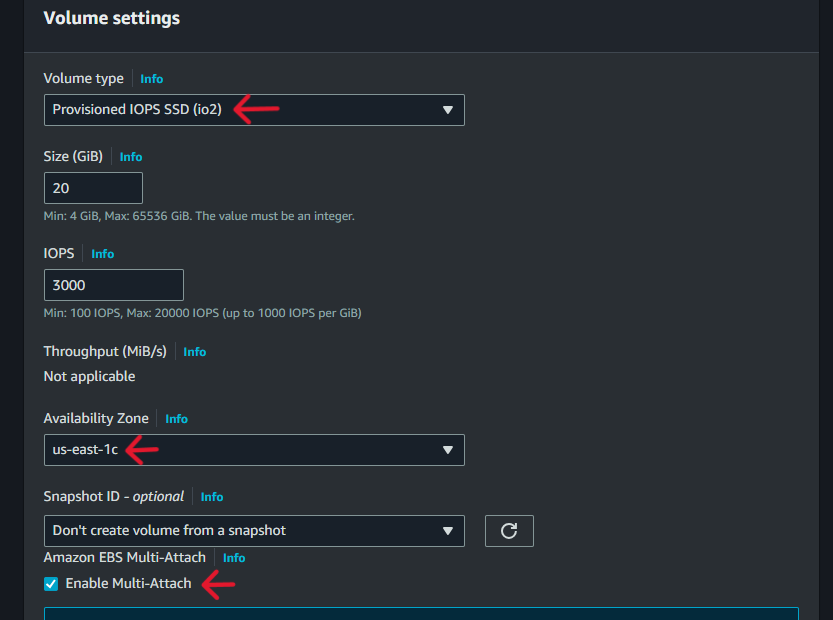
**Evaluate cost implications against performance benefits.**

**Practice: Configuring Amazon EBS Multi-Attach**

1. **Launch 2 or more (up to 16) Instances in the same Availability Zone. Ensure the instances are based on Nitro-based instance types. Here we are using “t3.micro” Instance type is a Nitro-based. (Allow SSH or required rules)**

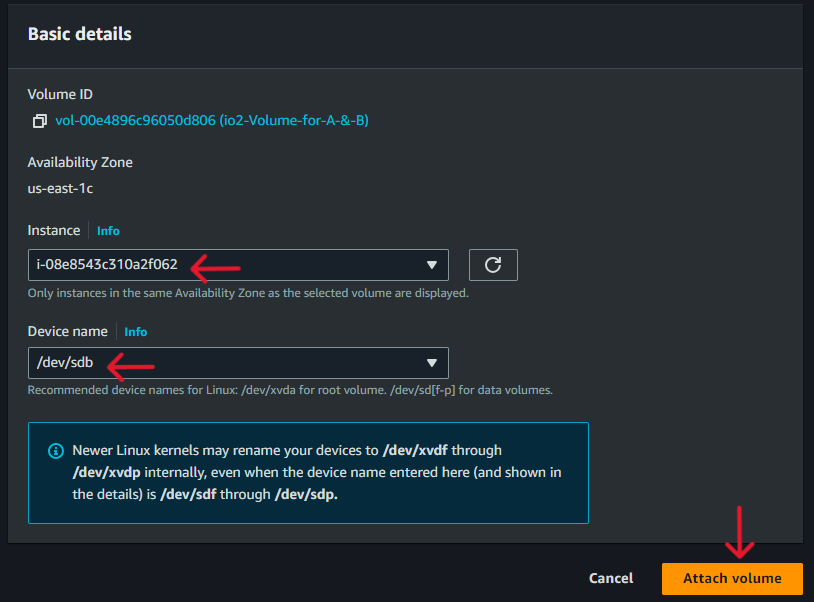


1. **Create a new io1 or io2 EBS Volume in the same availability zone where Instance launched. Enable Multi-Attach.**

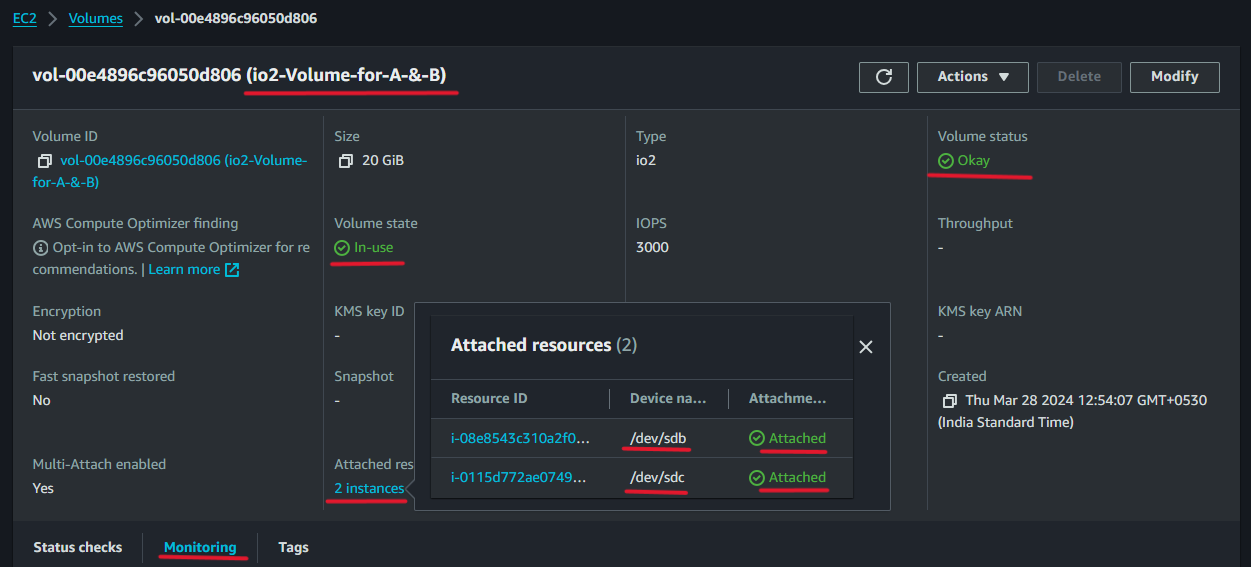


1. **Click on the Actions button and Select the Attach Volume option.**

* **Select the Instance and Device name.**
* **Click on the Attach volume button.**
* **Now follow the above step to attach the volume to the second Instance.**



1. **Click on Volume ID to check the Details or Status.**



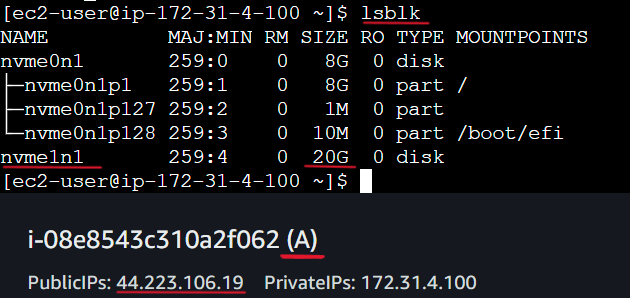
**Here we can see volume status is okay and successfully attached to both Instances.**

**Now we need to partition the volume and then mount it to a directory so we can utilize the volume.**

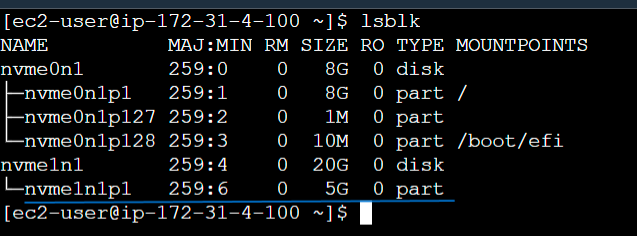
**(We also did this earlier)**

1. **SSH into each Instance and run the following commands**

* **lsbkl -> to list all block information and partitions**

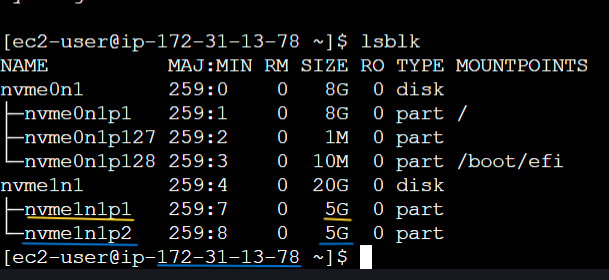


* **sudo fdisk /dev/nvme1n1 ->** to make a partition of volume
* **type m for help ->**  all options list will open with a description
* **type n for the new partition**
* **Then select partition type, by default on the primary just press Enter**
* We have up to **four primary partitions**, or Up to **three primary partitions and one extended partition**, **then choose the partition number**
* **The option for partition starts from the first sector, simply press Enter, and proceed with the last sector.**
* **Type +5G and enter -> create a 5 GB partition of disk and by default start from the last sector.**
* **Type w and press Enter for save and exit**
* Run the **lsbkl** command to check the partition.



**Now new partition nvme1n1p1 of 5 GB is successfully created.**

* **Following the above step create a partition for the B instance.**

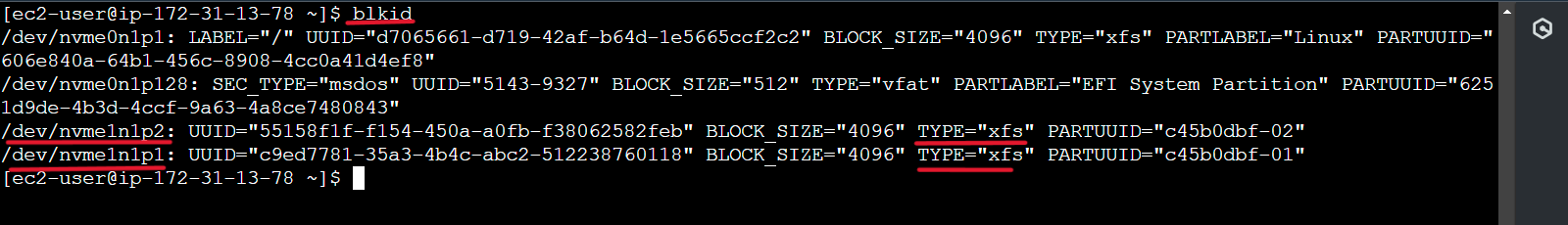


**Here we can see both partitions in the B Instance**

**Now we need to create the filesystem on that partition to store the files. Without creating a filesystem we can’t use those partitions.**

**To create a filesystem run the following commands:**

* **sudo mkfs –t xfs /dev/nvme1n1p1 -> to create a xfs filesystem in A Instance run it**
* **sudo mkfs –t xfs /dev/nvme1n1p2 -> to create a xfs filesystem in B Instance run it**
* **blkid -> to check all blocks filesystem type and other information**



**Here xfs file system was created in both Instances and we can see it B Instance as well as A Instance ( in the same case we need to reboot the Instance )**

**After the file system is created, we want to mount it to the directory in our file system so we can access it. So first we need to create a mount point in the [ /mnt ] directory this is the temporary mount point. (We can mount the filesystem in any other directory)**

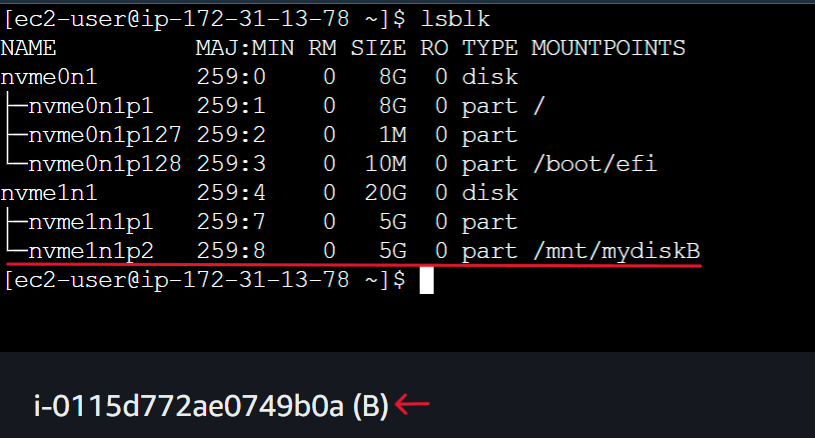
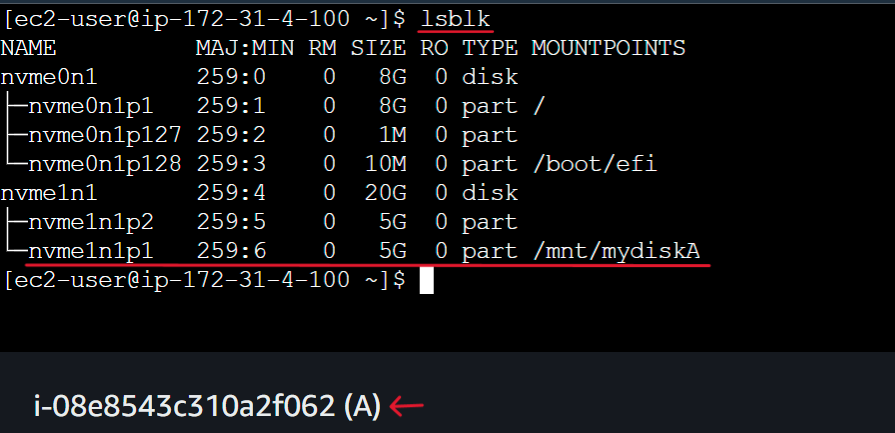
**For A Instance**

* **sudo mkdir /mnt/mydiskA**
* **sudo mount /dev/nvme1n1p1 /mnt/mydiskA**

**For B Instance**

* **sudo mkdir /mnt/mydiskB**
* **sudo mount /dev/nvme1n1p2 /mnt/mydiskB**

**Now, you will see your filesystem is temporarily mounted and available to use.**



**Here our single EBS Volume is now successfully deployed for use across multiple instances.**