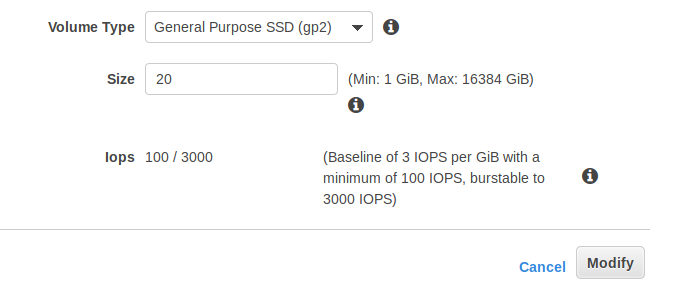
**AWS INTERVIEW QUESTIONS AND ANSWERS**

* **HOW TO EXTEND AWS EBS VOLUMES WITH NO DOWNTIME (WITH STOPPING EC2)**

**STEP 1: Extend the EBS volume.**

In order to extend the volume size, follow these simple steps:

1. Login to your AWS console
2. Choose “EC2” from the services list
3. Click on “Volumes” under ELASTIC BLOCK STORE menu (on the left)
4. Choose the volume that you want to resize, right-click on “Modify Volume”
5. You’ll see an option window like this one:



1. Set the new size for your EBS volume (in this case i extended an 8GB volume to 20GB)
2. Click on modify.

**STEP 1: Apply in ec2.**

SSH to the EC2 instance where the EBS we’ve just extended is attached to.

1. Type the following command to list our block devices:

**lsblk**

output:

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT  
xvda 202:0 0 20G 0 disk   
└─xvda1 202:1 0 8G 0 part /

As you can see size of the root volume reflects the new size as 20GB, the size of the partition reflects the original size as 8GB.

If there is no space at all then to run command like 0% - To avoid a **No space left on the block device** error, mount the temporary file system **tmpfs** to the **/tmp** mount point. This creates a 10 M **tmpfs** mounted to **/tmp:**

**sudo mount -o size=10M,rw,nodev,nosuid -t tmpfs tmpfs /tmp**

1. Extend the partition before you can extend the file system.

**sudo growpart /dev/xvda 1**

Now we can check that the partition reflects the increased volume size (we can check it with the lsblk command we already used):

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT  
xvda 202:0 0 20G 0 disk   
└─xvda1 202:1 0 20G 0 part /

1. Last but not least, we need to extend the filesystem itself.

* If your filesystem is an ext2, ext3, or ext4, type:

**sudo resize2fs /dev/xvda1**

* If your filesystem is an XFS, then type:

**sudo xfs\_growfs /dev/xvda1**

1. Finally, we can check our extended filesystem by typing: **df -h**

**EBS VOLUME TYPES**

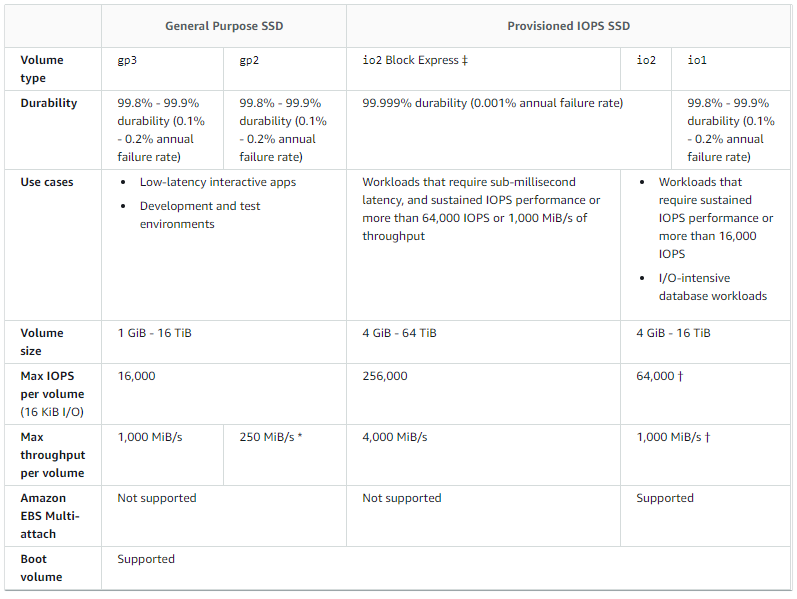
Aws EBS volumes differ in performance characteristics and price, so that we can choose our storage based on performance and cost that matches the needs of our applications.

1. **SOLID STATE DRIVES (SSD)**
2. **HARD DISK DRIVES (HDD)**
3. **PREVIOUS GENERATION VOLUME TYPES**

**SOLID STATE DRIVES (SSD)**

Optimized for transactional workloads involving frequent read/write operations with small I/O size

* **General Purpose SSD:** Provides a balance of price and performance. We recommend these volumes for most workloads.
* **Provisioned IOPS SSD:** Provides high performance for mission-critical, low-latency, or high-throughput workloads.



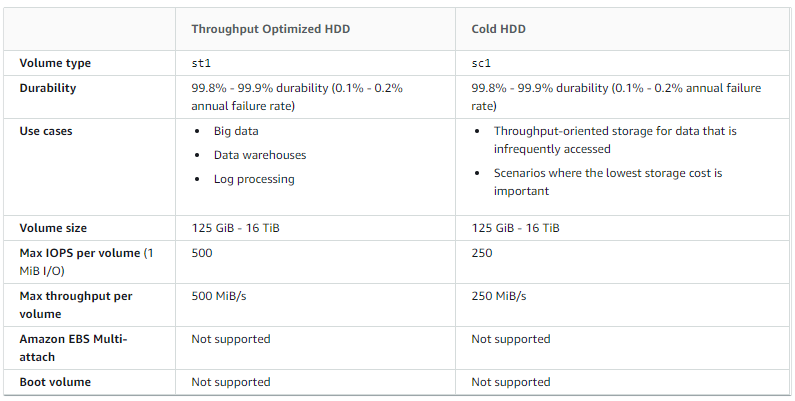
**PREVIOUS GENERATION VOLUME TYPES**

Hard disk drives that can be used for workloads with small datasets where data is accessed infrequently and performance is not of primary importance. AWS recommends that you consider a current generation volume type instead. I don’t know much about this type

**HARD DISK DRIVES (HDD)**

Optimized for large streaming workloads where the dominant performance attribute is throughput.

* **Throughput Optimized HDD:** A low-cost HDD designed for frequently accessed, throughput-intensive workloads.
* **Cold HDD:** The lowest-cost HDD design for less frequently accessed workloads.



* **How do I troubleshoot problems connecting to my Amazon EC2 Linux instance using SSH?**

**STEP 1:** To determine the issue first try to ssh the machine with verbose

**ssh -v -i my\_key.pem** [**ec2-user@11.22.33.44**](mailto:ec2-user@11.22.33.44)

**STEP 2:** Use the output messages from the SSH client to determine the type of issue/error.

1. **Error: "Connection timed out" or "Connection refused":**

The request fails to reach the instance and times out. This might happen if SSH isn't running on the instance or if a firewall is blocking access.

**Resolution:**

Verify the following:

* There isn't a firewall blocking the connection.
* The SSH service is running on the instance.
* The SSH tcp port 22 is in the listening state.

**There isn't a firewall blocking the connection.**

Check the NACL rules and Instance security group rules.

**The SSH service is running on the instance and The SSH tcp port 22 is in the listening state.**

METHOD 1: Connect to aws ec2 with EC2 Serial Console to restart ssh service

METHOD 2: Use AWS Systems Session Manager

1. Open the AWS Systems Manager console.

2. Start a session.

3. To disable firewalls and restart the SSH service, run the following commands.

sudo iptables -F

sudo service sshd restart

4. Verify that the SSH tcp port (22) is in a listening state.

sudo netstat -tnlp | grep :22

METHOD 3: Use a user data script to restart ssh service, Drawback – This procedure requires a stop and start of your EC2 instance.

1. **Error: "Permission denied" or "Authentication failed":**

**Resolution:**

Verify the following:

* You're trying to connect using the wrong user name for your AMI.
* The permissions are incorrect on the instance.
* The incorrect SSH public key (.pub) file is in the authorized\_keys file.

1. **Error: "Server refused our key":**

**Error: "imported-openssh-key" or "Putty Fatal Error":**

**Resolution:**

Verify the following:

* You're using the incorrect user name for your AMI when connecting to your EC2 instance. The usual user names are ec2-user, ubuntu, centos, root, or admin.
* The user trying to access the instance was deleted from the server or the account was locked.
* There are permissions issues on the instance or you're missing a directory.
* You're using the wrong private key when you negotiate an SSH session with an EC2 instance.
* **TRANSIT GATEWAY VS VPC PEERING?**

Transit Gateway solves the complexity involved with creating and managing multiple VPC peering connections at scale. While this makes TGW a good default for most network architectures, VPC peering is still a valid choice due to the following advantages it has over TGW:

* **LOWER COST**: With VPC peering you only pay for data transfer charges. Transit Gateway has an hourly charge per attachment in addition to the data transfer fees.
* **NO BANDWIDTH LIMITS:** With Transit Gateway, Maximum bandwidth (burst) per VPC connection is 50 Gbps. VPC peering has no aggregate bandwidth. Individual instance network performance limits and flow limits (10 Gbps within a placement group and 5 Gbps otherwise) apply to both options. Only VPC peering supports placement groups.
* **LATENCY:** Unlike VPC peering, Transit Gateway is an additional hop between VPCs.
* **SECURITY GROUPS COMPATIBILITY**: Security groups referencing works with intra-Region VPC peering. It does not currently work with Transit Gateway.

**VPC PEERING**

Using the VPC peering we can create an internal and more secure communication between two VPCs on the same or different regions and same or different AWS Account, we can close a connection between your VPC and the Partner or Customer VPC for example.

* Connect two different VPCs in the same regions
* Connect two VPCs on different regions (Inter-Region)
* Connect two VPCs on different accounts (Cross Accounts)

**Steps create a VPC Peering**

* We need a requester VPC
* We need an accepter VPC
* We need to configure the Route Table for the Accepter VPC
* We need to configure the Route Table for the Requester VPC
* We need to configure the Security Group for the service that will allow the communications from another VPC