

How to mount Linux volume and keep mount point consistency

by limillan | on 04 FEB 2022 | in [Amazon EC2](#), [Best Practices](#), [Technical How-to](#) | [Permalink](#) | [Share](#)

This post is written by: Leonardo Azize Martins, Cloud Infrastructure Architect, Professional Services

Customers often use [Amazon Elastic Compute Cloud \(Amazon EC2\)](#) Linux based instances with many [Amazon Elastic Block Store \(Amazon EBS\)](#) volumes attached. In this case, [device name](#) can vary depending on some facts, such as virtualization type, instance type, or operating system. As the device name can change, the customer shouldn't rely on the device name to mount volumes from it. The customer wants to avoid the situation where a volume is mounted on a different mount point just because the device name changed, or the device name doesn't exist because the name pattern changed.

Customers who want to utilize the latest instance family usually change the instance type when a new one is available. The device name can be different between instance families, such as T2 and T3. T2 uses `/dev/sd[a-z]`, while T3 uses `/dev/nvme[0-26]n1`. If you mount a device on T2 called `/dev/sdc`, when you change the instance family to T3 the same device won't be called `/dev/sdc` anymore. In this case, it will fail to mount.

Amazon EBS volumes are exposed as NVMe block devices on instances built on the [AWS Nitro System](#). The block device driver can assign NVMe device names in a different order than what you specified for the volumes in the block device mapping. In this situation, a device that should be mounted on `/data` could end-up being mounted on `/logs`.

On Linux, you can use the `fstab` file to mount devices using kernel name descriptors (the traditional way), file system labels, or the file system UUID. Kernel name descriptors aren't persistent and can change each boot. Therefore, they shouldn't be used in configuration files.

UUID is a mechanism for giving each filesystem a unique identifier. These identifiers' attributes are generated by filesystem utilities (`mkfs.*`) when the device is formatted, and they're designed so that collisions are unlikely. All GNU/Linux filesystems (including swap and LUKS headers of raw encrypted devices) support UUID.

As UUID is a filesystem attribute, it can also be used with Logical Volume Manager (LVM) and Linux software [RAID](#) (`mdadm`).

Depending on the `fstab` file configuration, you may find that you can't access your instance, which requires you to follow a [rescue process](#) to fix issues. This is the case if you configure the `fstab` file with the device name and change the instance type.

This post shows how to mount Linux volumes and keep mount points preserved when the instance type is changed or the instance is rebooted.

Overview of solution

When you create an instance, you specify block devices mapping. It doesn't mean that the Linux device has the same name or can be discovered in the same order as specified on the instance mapping. This situation can be more evident when using applications that require more volumes.

Using UUID to mount volumes lets you mitigate future issues when you stop and start your instance or change the instance type.

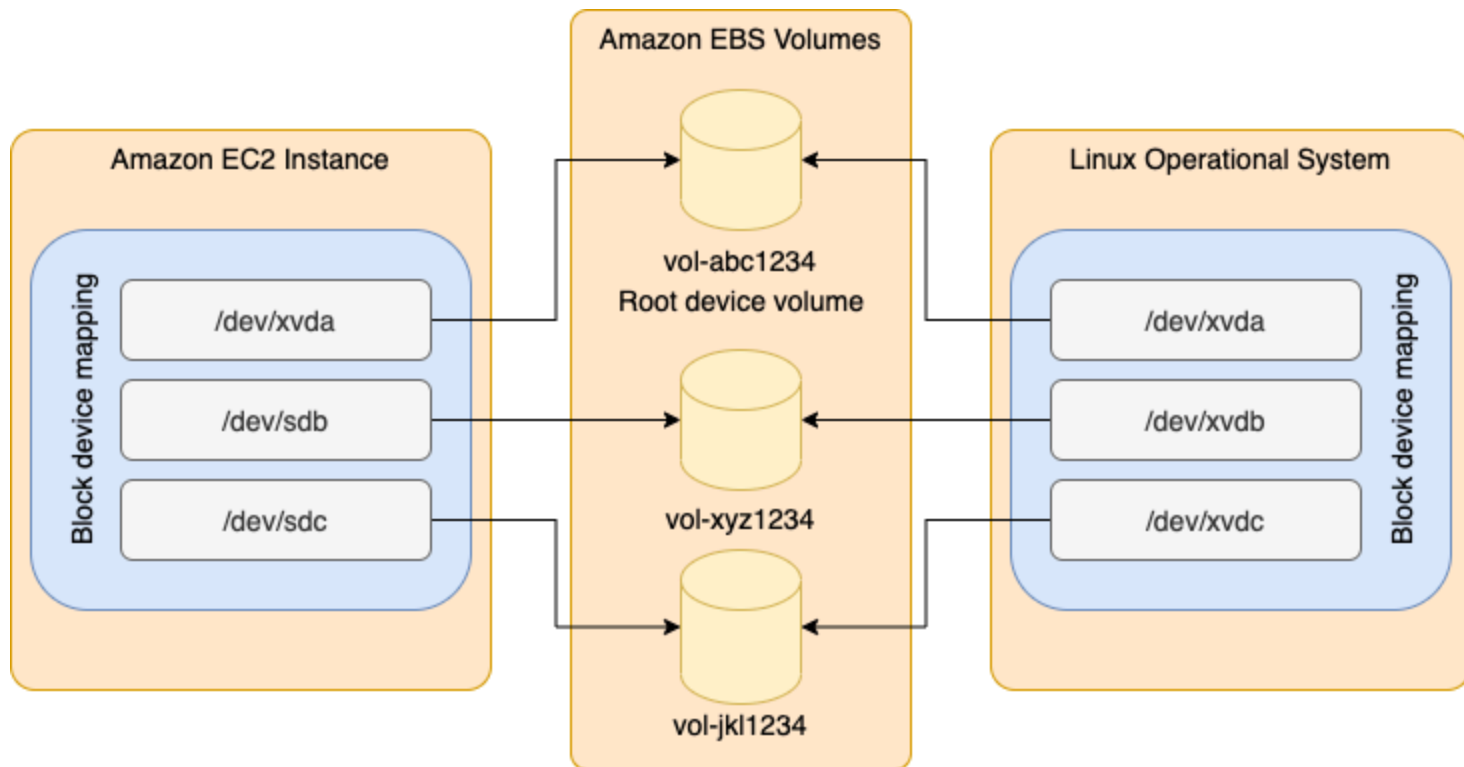


Figure 1: EC2 instance block device mapping

Walkthrough

You will create one instance with three volumes: one root volume and two data volumes. We use [Amazon Linux 2](#) in this post. In this instance type, volumes have a specific name format. Later, you will change the instance type. The new instance type volumes will have another name format.

Follow these steps:

- Create an instance with three volumes
- Create the filesystem on the volumes and mount them
- Change the instance type

Prerequisites

For this walkthrough, you should have the following prerequisites:

- An [AWS account](#)

- Linux knowledge

Create instance

Create one instance with three volumes: a root volume and two data volumes. Use [Launch Instance Wizard](#) with the following details.

Launch Amazon Linux 2 instance

1. On Step 1, choose **Amazon Linux 2 AMI (HVM), SSD Volume Type**.
2. On Step 2, choose **micro**.
3. On Step 3, choose **Next**.
4. On Step 4, add two new volumes, **device** `/dev/sdb` 10 GiB and **device** `/dev/sdc` 12 GiB.

Step 4: Add Storage

Your instance will be launched with the following storage device settings. You can attach additional EBS volumes and instance store volumes to your instance, or edit the settings of the root volume. You can also attach additional EBS volumes after launching an instance, but not instance store volumes. [Learn more](#) about storage options in Amazon EC2.

Volume Type ⓘ	Device ⓘ	Snapshot ⓘ	Size (GiB) ⓘ	Volume Type ⓘ	IOPS ⓘ	Throughput (MB/s) ⓘ	Delete on Termination ⓘ	Encryption ⓘ
Root	/dev/xvda	snap-0fee1f2b4c19cfcac	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypte ▾
EBS ▾	/dev/sdb ▾	Search (case-insensit	10	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypte ▾ ×
EBS ▾	/dev/sdc ▾	Search (case-insensit	12	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypte ▾ ×

Figure 2: Launch instances, add storage

Create filesystem and mount

Connect to your instance using [EC2 Instance Connect](#) or any other method that feels comfortable for you. Mount the device using UUID instead of the device name. Run the following instructions as the user root.

Format and mount the device

1. Run the following command to confirm that you have three disks:

```
Bash
$ lsblk
```

2. Format disk as **xfs**, and run the following commands:

```
Bash
mkfs.xfs /dev/xvdb
mkfs.xfs /dev/xvdc
```

3. Create mount point, and run the following commands:

Bash

```
mkdir /mnt/disk1  
mkdir /mnt/disk2
```

4. Add mount instructions, and run the following commands:

Bash

```
echo "$(blkid -o export /dev/xvdb | grep ^UUID=) /mnt/disk1 xfs defaults,noatime" | tee -a  
echo "$(blkid -o export /dev/xvdc | grep ^UUID=) /mnt/disk2 xfs defaults,noatime" | tee -a
```

5. Mount volumes, create dummy file, and run the following commands:

Bash

```
mount -a  
touch /mnt/disk1/file1.txt  
touch /mnt/disk2/file2.txt
```

You will have an fstab file like the following:

Bash

```
cat /etc/fstab  
UUID=7b355c6b-f82b-4810-94b9-4f3af651f629 / xfs defaults,noatime 1 1  
UUID="2c160dd6-586c-4258-84bb-c79933b9ae02" /mnt/disk1 xfs defaults,noatime  
UUID="3e7a0c28-7cf1-40dc-82a1-2f5cfb53f9a4" /mnt/disk2 xfs defaults,noatime
```

Change instance type

Change the instance type from **t2.micro** to **t3.micro**.

Launch Amazon Linux 2 instance

1. Stop instance.
2. [Change the instance](#) type to **micro**.
3. Start instance.
4. Connect to your instance using [EC2 Instance Connect](#).
5. Check the device name, and run the following command:

Bash

```
lsblk
```

6. List files, and run the following command:

Bash

```
ls -l /mnt/*
```

Note that the device names are changed from `xvd*` to `nvme*`. All of the devices are mounted without any issue and with the correct mount points.

Cleaning up

To avoid incurring future charges, delete the instance and all of the volumes that you created in this post.

The other side

UUID is an attribute of the filesystem that was generated when you formatted your device. Therefore, it will follow your device even if you create an AMI or snapshot. So you don't need to worry about a restore process, and it will smoothly proceed to an instance restore. You must be careful if you restore a snapshot from one volume and attach it to the same instance, as you will end-up with two volumes that are using the same UUID. If you try to mount the restored volume on the same instance, then it will fail and you will find this message on `/var/log/messages` file.

```
kernel: XFS (xvdf1): Filesystem has duplicate UUID f6646b81-f2a6-46ca-9f3d-c746cf015379 - can't mount
```

It is even more important to be careful if you attach a volume created from the snapshot of the root volume and restart your instance. Since both volumes have the same UUID, you may find that a volume other than the one attached to `/dev/xvda` or `/dev/sda` has become the root volume of your instance. See the following example for details. Note that both volumes have the same UUID, but the one mounted on `/` is `/dev/xvdf1`, not `/dev/xvda1`, which is the real root volume for this instance.

Bash

```
$ blkid
```

```
/dev/xvda1: LABEL="/" UUID="f6646b81-f2a6-46ca-9f3d-c746cf015379" TYPE="xfs" PARTLABEL="Linux"  
/dev/xvdf1: LABEL="/" UUID="f6646b81-f2a6-46ca-9f3d-c746cf015379" TYPE="xfs" PARTLABEL="Linux"
```

Bash

```
$ lsblk
```

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

```
xvdf      202:80    0    8G    0 disk
└─xvdf1   202:81    0    8G    0 part /
```

and Linux software RAID (mdadm), UUID, as an attribute of the filesystem, will be the same even after a backup and restore process, snapshot, or clone. To learn more, check out our [block device mappings](#) and [device names on Linux instances](#) documentation.

TAGS: [Amazon EC2](#), [Best Practices](#), [Compute](#), [Technical How-To](#)