



AWS Instance Launching and Disk Partition

How to launch an AWS instance, attach an EBS volume to it, and create a partition in it



MishanRG · [Follow](#)

Published in Towards AWS · 9 min read · Oct 30, 2020



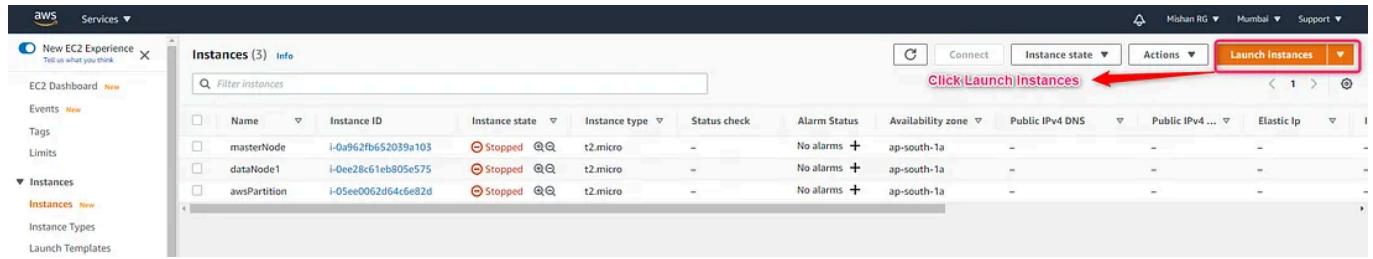
Hello Geeks, in this blog, I will talk about how we can launch an AWS instance from the AWS dashboard and attach extra storage, i.e., EBS Storage, to create a partition in that volume. I will be explaining all the processes step by step, supported by images. So let's get started...

Pre-requisites for the blog

- Basic Knowledge of AWS, EBS
- Basic Linux Command Understanding
- Knowledge of Hard Disk Partition

Configuring An Instance

Firstly, let's log in first into our AWS dashboard and then launch an EC2 instance from the services.



Then we select any AMI from the given list according to our need and flavor. Here I choose the Amazon Linux AMI.

Step 1: Choose an Amazon Machine Image (AMI)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. You can select an AMI provided by AWS, our user community, or the AWS Marketplace; or you can select one of your own AMIs.

Then we select the instance type what we want. The instance type lets us choose any instance depending on vCPU, memory(RAM), and storage.

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by:	All instance families	Current generation	Show/Hide Columns
Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, ~1 GiB memory, EBS only)			
	Family	Type	vCPUs (1)
<input type="checkbox"/>	t2	t2.nano	1
<input checked="" type="checkbox"/>	t2	t2.micro Free tier eligible	1
<input type="checkbox"/>	t2	t2.small	1
<input type="checkbox"/>	t2	t2.medium	2
<input type="checkbox"/>	t2	t2.large	2

Now we configure our instance detail. It includes the number of CPU, network type, Availability zone to launch, and more configuration of what we want in our instance. Here I left everything on default as it is good for what we are going for.

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take advantage of the lower pricing, assign an access management role to the instance, and more.

We need storage in our instance. The storage type is pre-selected when we choose our instance type. The storage type is the root storage where our OS is launched. And we added a new volume with our instance from the Add New Volume button, and we can add the required volume size.

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-027b63b5a2b94b7c3	8	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted
EBS	/dev/sdb	Search (case-insensit)	5	General Purpose SSD (gp2)	100 / 3000	N/A	<input checked="" type="checkbox"/>	Not Encrypted

Add New Volume Click add|new|Volume

Free tier eligible customers can get up to 30 GB of EBS General Purpose (SSD) or Magnetic storage. Learn more about free usage tier eligibility and usage restrictions.

Now we have to configure a security group to our instance. The security group is the firewall rules for controlling the incoming and outgoing network traffic for our instance. Here I have enabled all the traffic, which is

not the best way, but you can configure your own security rule as per your need.

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: Create a new security group Select an existing security group

Security group name:	launch-wizard-1			
Description:	launch-wizard-1 created 2020-10-29T10:56:00.565+05:30			
Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom	0.0.0.0/0
e.g. SSH for Admin Desktop				

Add Rule

Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

We have to review all the configurations we have done till now and then add a key pair to our instance. Key pair is basically an authentication to access our instance. We create the key pair, and then AWS provides us a key, and we need to download it and save it safely. Here I had already created a key pair, so I used that one.

Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about [removing existing key pairs from a public AMI](#).

Choose an existing key pair

Select a key pair

hadoopKey

I acknowledge that I have access to the selected private key file (hadoopKey.pem), and that without this file, I won't be able to log into my instance.

[Cancel](#) [Launch Instances](#)

After reviewing and launching the instance, we can see a configuration message on our screen and view our instance from there.

Launch Status

Your instances are now launching
The following instance launches have been initiated: i-05ee0062d64c6e82d View launch log

Get notified of estimated charges
Create billing alerts to get an email notification when estimated charges on your AWS bill exceed an amount you define (for example, if you exceed the free usage tier).

How to connect to your instances
Your instances are launching, and it may take a few minutes until they are in the **running** state, when they will be ready for you to use. Usage hours on your new instances will start immediately and continue to accrue until you stop or terminate your instances.
Click [View Instances](#) to monitor your instances' status. Once your instances are in the **running** state, you can [connect](#) to them from the Instances screen. [Find out](#) how to connect to your instances.

Here are some helpful resources to get you started

- How to Connect to your Linux instance
- Amazon EC2: User Guide
- Learn about AWS Free Usage Tier
- Amazon EC2: Discussion Forum

While your instances are launching you can also

- Create status check alarms to be notified when these instances fail status checks. (Additional charges may apply)
- Create and attach additional EBS volumes. (Additional charges may apply)
- Manage security groups

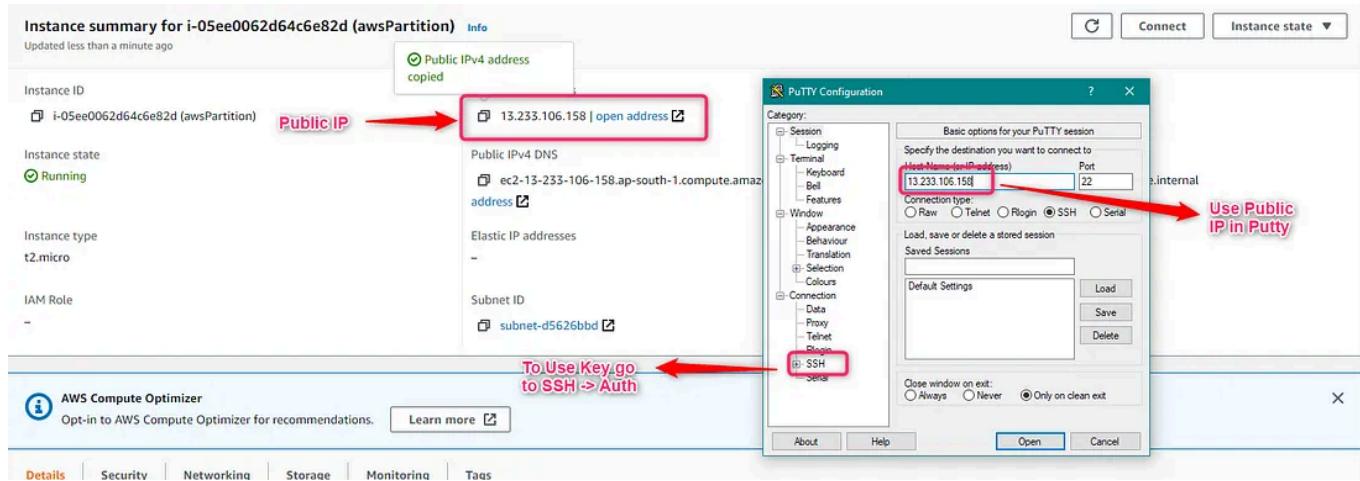
[View Instances](#)

As we can see in the below image, our instance has been launched and running.

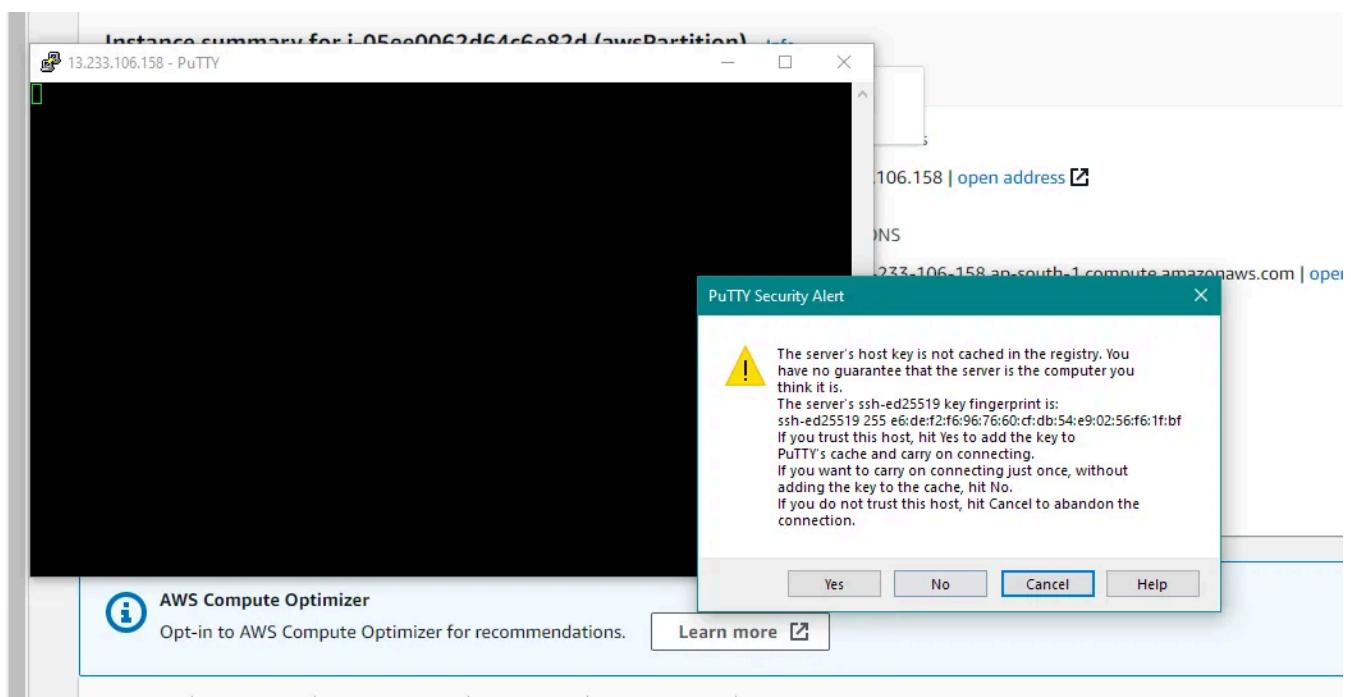
Name	Instance ID	Instance State	Instance Type	Status Check	Alarm Status	Availability Zone	Public IPv4 DNS	Public IPv4 IP	Elastic IP
masterNode	i-0a962fb652059a103	Stopped	t2.micro	-	No alarms	ap-south-1a	-	-	-
dataNode1	i-0ee28c61eb805e575	Stopped	t2.micro	-	No alarms	ap-south-1a	-	-	-
awsPartition	i-05ee0062d64c6e82d	Running	t2.micro	Initializing	No alarms	ap-south-1a	ec2-13-233-106-158.a...	13.233.106.158	-

Launching An Instance

Now we will launch this instance and use it. As I have launched an AWS Linux AMI, I can use the instance in two ways. One way is using it on the browser, and another way is using SSH to connect it from our system, and for SSH, we will be using the [PuTTY](#) app.



We open our Putty app and then copied the instance's public IP, and then convert the .ppk key pair file given by AWS into a .pem file using the application [PuTTYgen](#). Then we add that file by going in SSH and Auth and then uploading it there, and we open SSH connection.



Then we get connected to our instance, and then we have to enter the username, which will be “ec2-user” by default, and then get into the root account as it provides more function and power. The command is:

```
$sudo -su root
```

The screenshot shows a terminal window on an Amazon Linux 2 AMI. The user has logged in as 'ec2-user'. The terminal displays a welcome message for Amazon Linux 2, a link to the Amazon Linux 2 documentation, and a note about security updates. The user then runs the command `sudo -su root`, which changes the user to 'root'. The prompt changes to show the root user's name.

```
root@ip-172-31-45-85:/home/ec2-user
[ec2-user@ip-172-31-45-85 ~]$ sudo -su root
root@ip-172-31-45-85 ec2-user#
```

We have launched our instance; we will check if the EBS volume is attached. To check it, we use the command:

```
$ fdisk -l
```

We can see that the EBS volume we attached to 5GB is visible but not assigned to a device as our root volume. So in the next step, we will attach it to our instance.

```
root@ip-172-31-45-85:/home/ec2-user
└─ login as: ec2-user
└─ Authenticating with public key "imported-openssh-key"

      _|_(_|_)_
      _\|_|_|_|_ Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
25 package(s) needed for security, out of 39 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-45-85 ~]$ sudo -su root
[root@ip-172-31-45-85 ec2-user]# fdisk -l
Disk /dev/xvda: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 66B3909F-969E-4FD1-901C-CEE3A9974A83

Device      Start    End  Sectors Size Type
/dev/xvda1     4096 16777182 16773087   8G Linux filesystem
/dev/xvda128   2048     4095     2048   1M BIOS boot

Partition table entries are not in disk order.

Disk /dev/xvdb: 5 GiB, 5368709120 bytes, 10485760 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[root@ip-172-31-45-85 ec2-user]#
```

Mounting EBS volume on our Instance

We can see our instance is also launched, and EBS storage is also attached to it now; we will mount it to the instance. As the topic is about partition, we will attach a part of, i.e., 3GB of the storage from the EBS volume, and then the same way, we can attach the remaining storage.

Mounting can be completed in 3 easy steps:

Step 1 Disk Partition

First, we need to divide the storage from the main storage and get how much storage we need. Hard disk space is divided using sectors, and 1 sector is equal to 512 bytes. So we need to divide the storage using sectors. We get inside that disk using the command:

```
$ fdisk <diskname>
```

```
https://aws.amazon.com/amazon-linux-2/
25 package(s) needed for security, out of 39 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-45-85 ~]$ sudo -su root
[root@ip-172-31-45-85 ec2-user]# fdisk -l
bash: fdisk: command not found
[root@ip-172-31-45-85 ec2-user]# fdisk -l
Disk /dev/xvda: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 66B3909F-969E-4FD1-901C-CEE3A9974A83

Device      Start    End  Sectors Size Type
/dev/xvda1     4096 16777182 16773087   8G Linux filesystem
/dev/xvda128   2048      4095      2048   1M BIOS boot

Partition table entries are not in disk order.

Disk /dev/xvdb: 5 GiB, 5368709120 bytes, 10485760 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[root@ip-172-31-45-85 ec2-user]# fdisk /dev/xvdb
```

During partition, there are two types of partition we can do, i.e., primary and another is extended. We can create 4 primary partitions in one hard drive independent of any brand or OS. And if we create 4 partitions, the remaining space will be unallocated space. So to save our data, we don't create a 4th

partition. We treat that space as a new hard drive and create extended storage, and we can't use it, so we start creating other partitions there, and they are called a logical partition.

```
Disk /dev/xvdb: 5 GiB, 5368709120 bytes, 10485760 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
[root@ip-172-31-45-85 ec2-user]# fdisk /dev/xvdb

Welcome to fdisk (util-linux 2.30.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xc07cc958.

Command (m for help): █
```

As we are inside the disk, we can use the various command to word with the disk:

- n to create a new partition
- p to show the detail of the hard drive partition
- p for selecting primary storage type and e for extended storage type
- d to delete the partition
- q to quit from the command menu
- w to save the configuration we did

We check the status using command p and start creating a new partition using command n.

```
Welcome to fdisk (util-linux 2.30.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xc07cc958.

Command (m for help): p
Disk /dev/xvdb: 5 GiB, 5368709120 bytes, 10485760 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xc07cc958

Command (m for help): █
```

In the below image, we can see we have used the “p” command to check the hard drive status, and we used the “n” command to create new storage inside how much we need. We selected the p option for primary storage, and then as some part of the hard drive is used, which is the 2047 sector. We start with 2048, and then we set our endpoint to +3G, which is up-top 3GiB. We can also write sector value by calculating how much we need. When we recheck the status using the “p” command, we can see a new device has been made named /dev/xvdb1 with a size of 3GiB.

```
Welcome to fdisk (util-linux 2.30.2).
Changes will remain in memory only, until you decide to write them.
Be careful before using the write command.

Device does not contain a recognized partition table.
Created a new DOS disklabel with disk identifier 0xc07cc958.

Command (m for help): p
Disk /dev/xvdb: 5 GiB, 5368709120 bytes, 10485760 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xc07cc958

Command (m for help): n
Partition type
  p  primary (0 primary, 0 extended, 4 free)
  e  extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1):
First sector (2048-10485759, default 2048):
Last sector, +sectors or +size{K,M,G,T,P} (2048-10485759, default 10485759): 3G
Value out of range.
Last sector, +sectors or +size{K,M,G,T,P} (2048-10485759, default 10485759): +3G

Created a new partition 1 of type 'Linux' and of size 3 GiB.

Command (m for help): p
Disk /dev/xvdb: 5 GiB, 5368709120 bytes, 10485760 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xc07cc958

Device      Boot Start      End Sectors Size Id Type
/dev/xvdb1        2048 6293503 6291456   3G 83 Linux

Command (m for help):
```

To check the status

Ask for partition type.. To create new

We selected primary storage and starting point is by default set and end point we gave +3G

After completing the partition, we have to save what we have configured, and we save it using the “w” command.

```

Command (m for help): p
Disk /dev/xvdb: 5 GiB, 5368709120 bytes, 10485760 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xc07cc958

Device      Boot Start      End Sectors Size Id Type
/dev/xvdb1        2048 6293503 6291456   3G 83 Linux

Command (m for help): w
The partition table has been altered.
Calling ioctl() to re-read partition table.
Syncing disks.

[root@ip-172-31-45-85 ec2-user1]#

```

You all might be wondering why can't we use sector 0 to 2047. It is because the storage is allocated for the Inode table created during formatting. This table stores the information of all the partitions in the hard drive. It stores the metadata information of all the files located in the hard drive so that it will be easy to search for data inside with metadata's help. Without an inode table, it can take 3-4 days to search and image files in the hard drive sectors. We will discuss more in the formatting part.

As we have seen that the new device has been added to our partition, we need to allocate the device to a driver as no program can run without a driver in an OS. We can allocate the driver to that device using the command.

```

$ udevadm settle // in RHE Linux 8
$ partprobe <deviceName> // till RHE Linux 7

```

Step 2 Formatting

Now we have to format the device we created. The formatting creates a fresh Inode table, and we cannot use any hard drive space without creating an Inode table. Whenever we open any file in OS, it goes to the Inode table to check the metadata and check the sector in which the file is located in the hard drive and give us the output.

We can create only 4 primary partitions because the Inode table's size is 64byte. The metadata size given by every primary partition is 16byte, so we can only store 4 partition information in the Inode table.

Now to format the device, we can use the command.

```
$ mkfs.ext4 <deviceName>
```

Here mkfs is make file system, and ext4 is one of the file systems of Linux. We can check the filesystem available in Linux using the command.

```
$ mkfs
```

Different filesystems have different ways of storing data in them. You can check the type of filesystem and their behavior [here](#).

```
[root@ip-172-31-45-85 ec2-user]# mkfs.ext4 /dev/xvdb1
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
196608 inodes, 786432 blocks
39321 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=805306368
24 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
      32768, 98304, 163840, 229376, 294912
```

Formatting the storage

Open in app ↗

Sign up

Sign in



Search



Write



```
[root@ip-172-31-45-85 ec2-user]# mkdir /newspace
[root@ip-172-31-45-85 ec2-user]#
```

Step 4 Mounting

Now, as we have created a partition and formatted the space, we need to mount it. In OS, we can access only files and directory. We need any file or directory to access our storage device, so we create a new directory in our system and mount that storage device in that directory. The command is

```
$mkdir /newspace //any name can be given to directory
$mount <deviceName> /newspace
```

```
[root@ip-172-31-45-85 ec2-user]# mkfs.ext4 /dev/xvdb1
mke2fs 1.42.9 (28-Dec-2013)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=0 blocks, Stripe width=0 blocks
196608 inodes, 786432 blocks
39321 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=805306368
24 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
    32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done
```

```
[root@ip-172-31-45-85 ec2-user]# mkdir /newspace
[root@ip-172-31-45-85 ec2-user]# mount /dev/xvdb1 /newspace
[root@ip-172-31-45-85 ec2-user]#
```

In Windows, the OS is stored in C:/ directory storage, and in Linux, the OS is stored in / “root” directory.

Now all three steps are completed. We can check the hard drive detail and check if the new space is available to use or not. In the below image, we can see that in the 5GiB storage device, we can see our 3GiB storage device.

```
[root@ip-172-31-45-85 ec2-user]# mkdir /newspace
[root@ip-172-31-45-85 ec2-user]# mount /dev/xvdbl /newspace
[root@ip-172-31-45-85 ec2-user]# fdisk -l
Disk /dev/xvda: 8 GiB, 8589934592 bytes, 16777216 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: gpt
Disk identifier: 66B3909F-969E-4FD1-901C-CEE3A9974A83

Device      Start     End   Sectors Size Type
/dev/xvda1    4096 16777182 16773087   8G Linux filesystem
/dev/xvda128  2048      4095      2048   1M BIOS boot

Partition table entries are not in disk order.

Disk /dev/xvdb: 5 GiB, 5368709120 bytes, 10485760 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xc07cc958

Device      Boot Start     End   Sectors Size Id Type
/dev/xvdbl        2048 6293503 6291456   3G 83 Linux
[root@ip-172-31-45-85 ec2-user]#
```

And using the “\$ df -h” command, we can see that the new space has been made and allocated to the “/new space ” directory.

```
Device      Boot Start   End Sectors Size Id Type
/dev/xvdb1        2048 6293503 6291456   3G 83 Linux
[root@ip-172-31-45-85 ec2-user]# dh -f
bash: dh: command not found
[root@ip-172-31-45-85 ec2-user]# df -h
Filesystem      Size  Used Avail Use% Mounted on
devtmpfs        474M    0  474M   0% /dev
tmpfs          492M    0  492M   0% /dev/shm
tmpfs          492M  464K 492M   1% /run
tmpfs          492M    0  492M   0% /sys/fs/cgroup
/dev/xvda1       8.0G  1.4G  6.7G  17% /
tmpfs          99M    0   99M   0% /run/user/0
tmpfs          99M    0   99M   0% /run/user/1000
/dev/xvdb1       2.9G  9.0M  2.8G   1% /newspace
[root@ip-172-31-45-85 ec2-user]#
```

Similarly, we can create a partition for the remaining 2GiB and allocate it to a new directory. And we can create 3 more partitions on that 2GiB of storage.

Conclusion

Finally, we can have completed our task of launching an AWS instance with an EBS volume and creating a partition in that storage.

Guess I have explained every detailed bit by bit, and if you have any doubts or suggestions, you can comment on this blog or contact me on my LinkedIn post.

Mishan Regmi - Christ University, Bangalore - Bengaluru, Karnataka, India | LinkedIn

View Mishan Regmi's profile on LinkedIn, the world's largest professional community. Mishan's education is listed on...

www.linkedin.com

