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

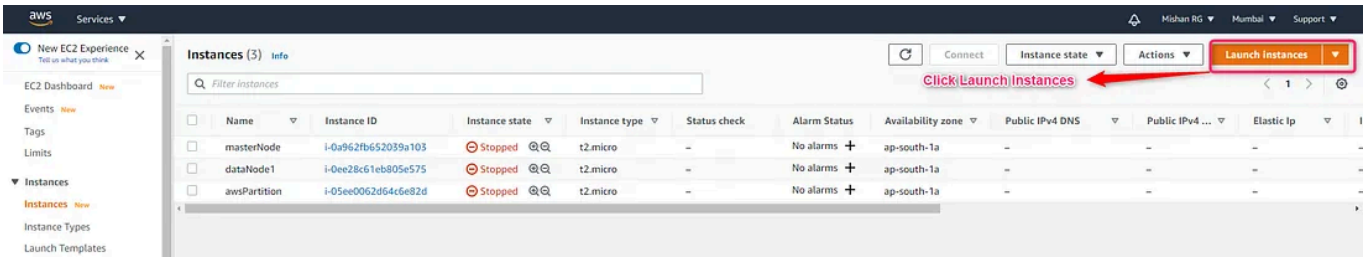
Hello Geeks, in this blog, I will talk about how we can launch an AWS instance from the AWS dashboard and attach extra storage, i.e., EBS Storage, to create a partition in that volume. I will be explaining all the processes step by step, supported by images. So let's get started...

Pre-requisites for the blog

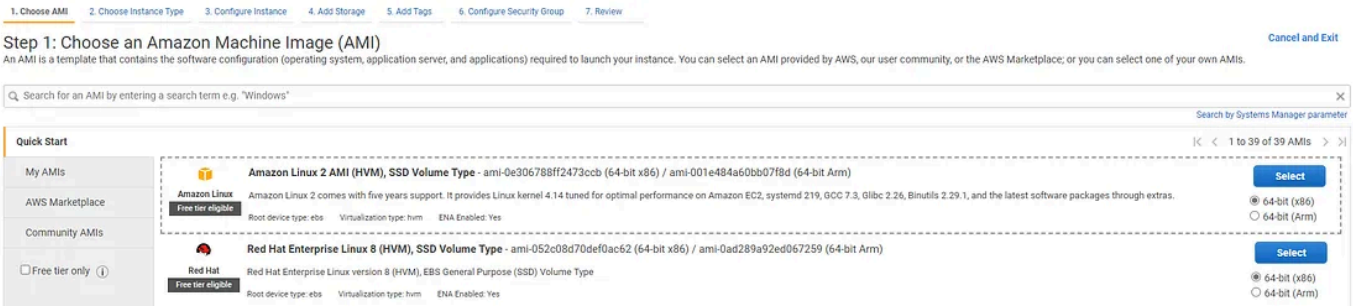
- Basic Knowledge of AWS, EBS
- Basic Linux Command Understanding
- Knowledge of Hard Disk Partition

Configuring An Instance

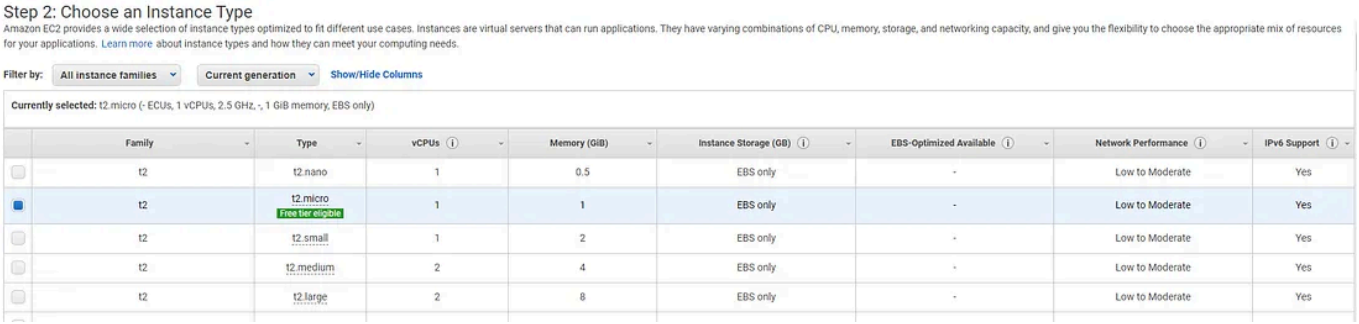
Firstly, let's log in first into our AWS dashboard and then launch an EC2 instance from the services.



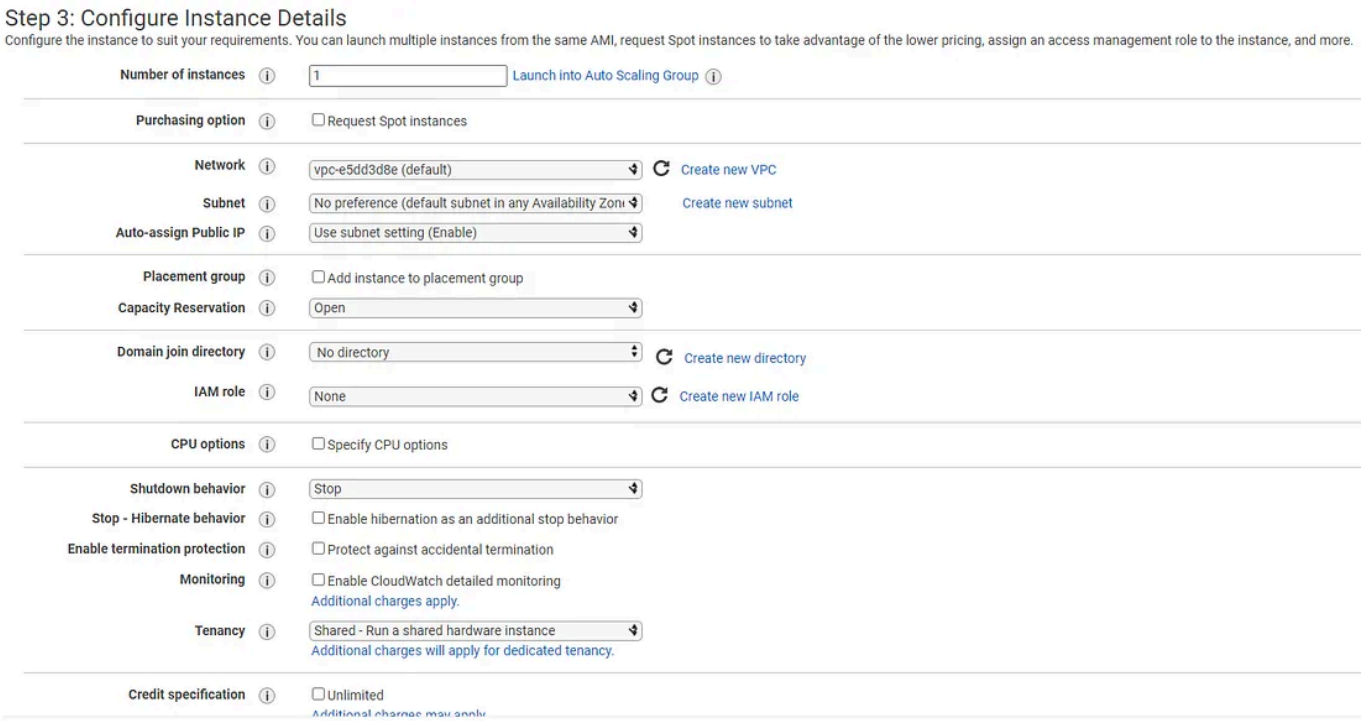
Then we select any AMI from the given list according to our need and flavor. Here I choose the Amazon Linux AMI.



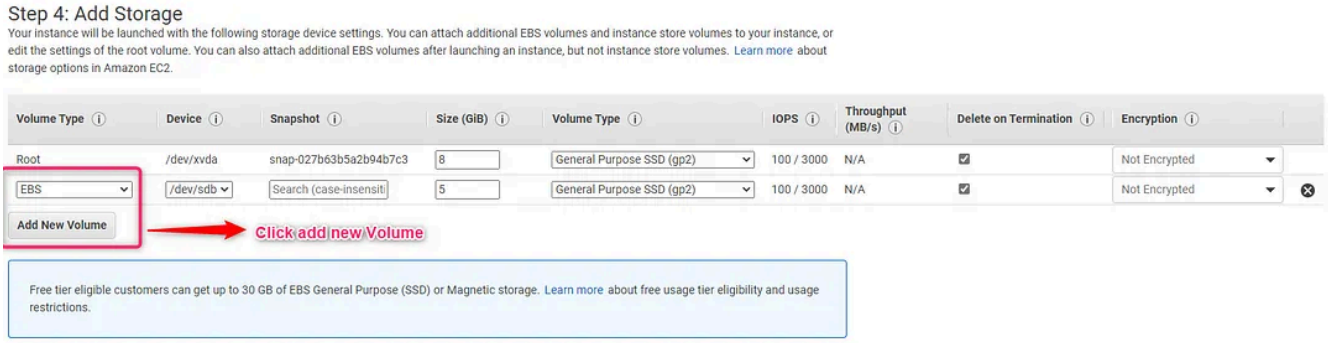
Then we select the instance type what we want. The instance type lets us choose any instance depending on vCPU, memory(RAM), and storage.



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.

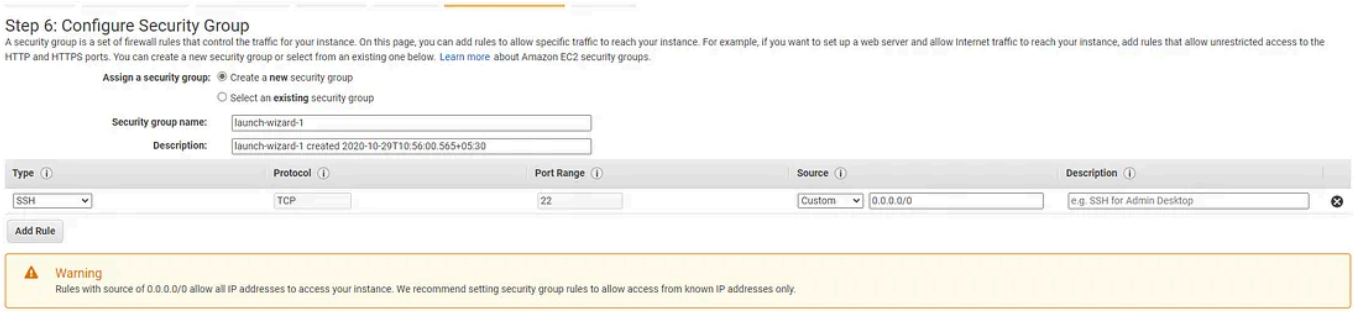


We need storage in our instance. The storage type is pre-selected when we choose our instance type. The storage type is the root storage where our OS is launched. And we added a new volume with our instance from the Add New Volume button, and we can add the required volume size.

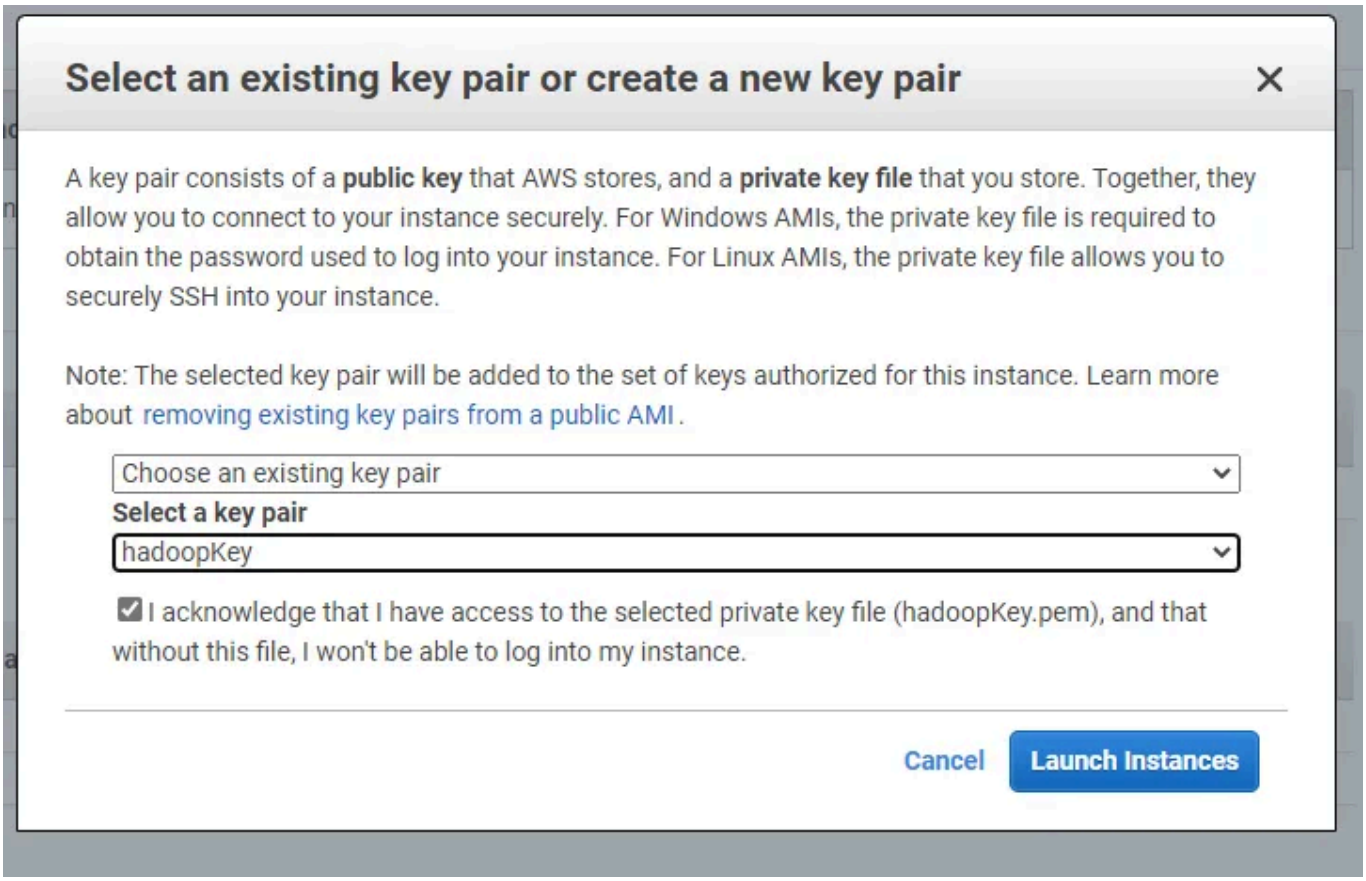


Now we have to configure a security group to our instance. The security group is the firewall rules for controlling the incoming and outgoing network traffic for our instance. Here I have enabled all the traffic, which is

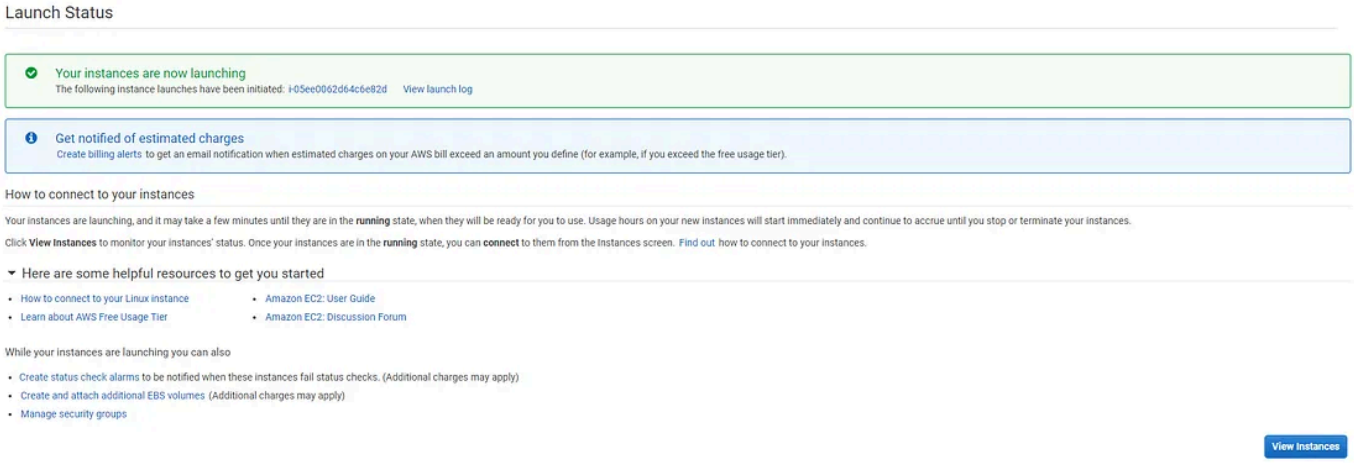
not the best way, but you can configure your own security rule as per your need.



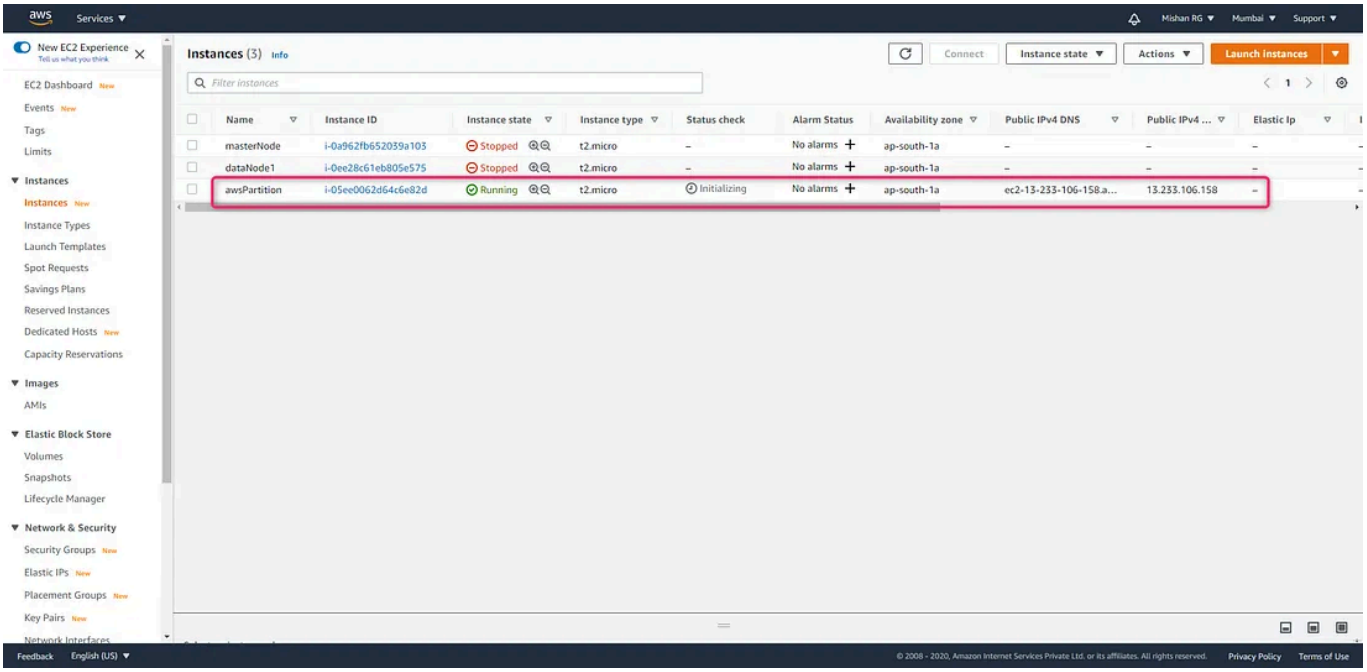
We have to review all the configurations we have did till now and then add a key pair to our instance. Key pair is basically an authentication to access our instance. We create the key pair, and then AWS provides us a key, and we need to download it and save it safely. Here I had already created a key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



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

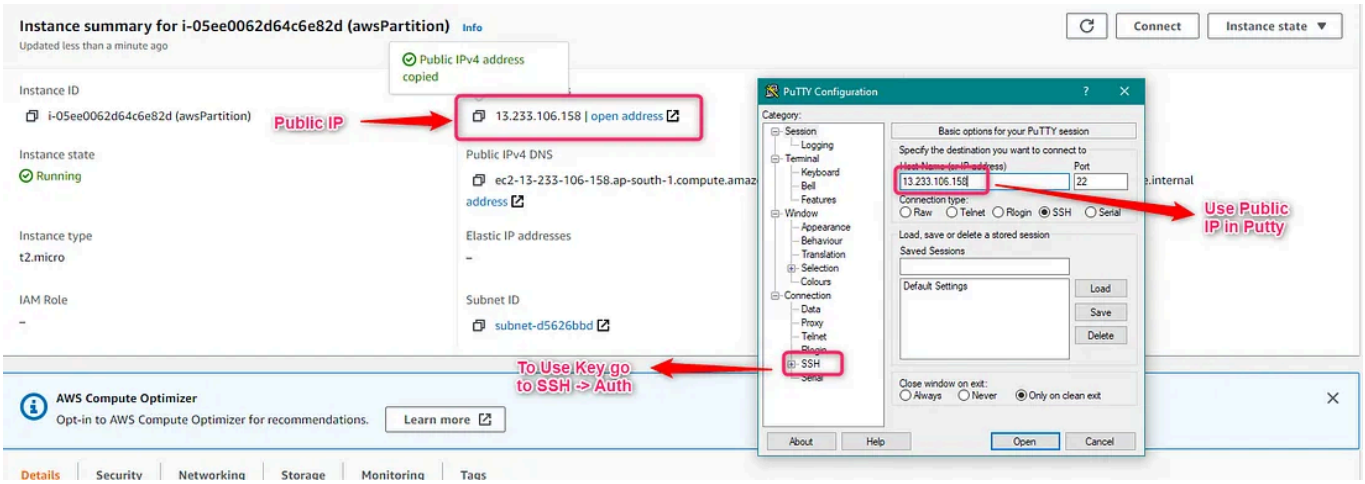


As we can see in the below image, our instance has been launched and running.

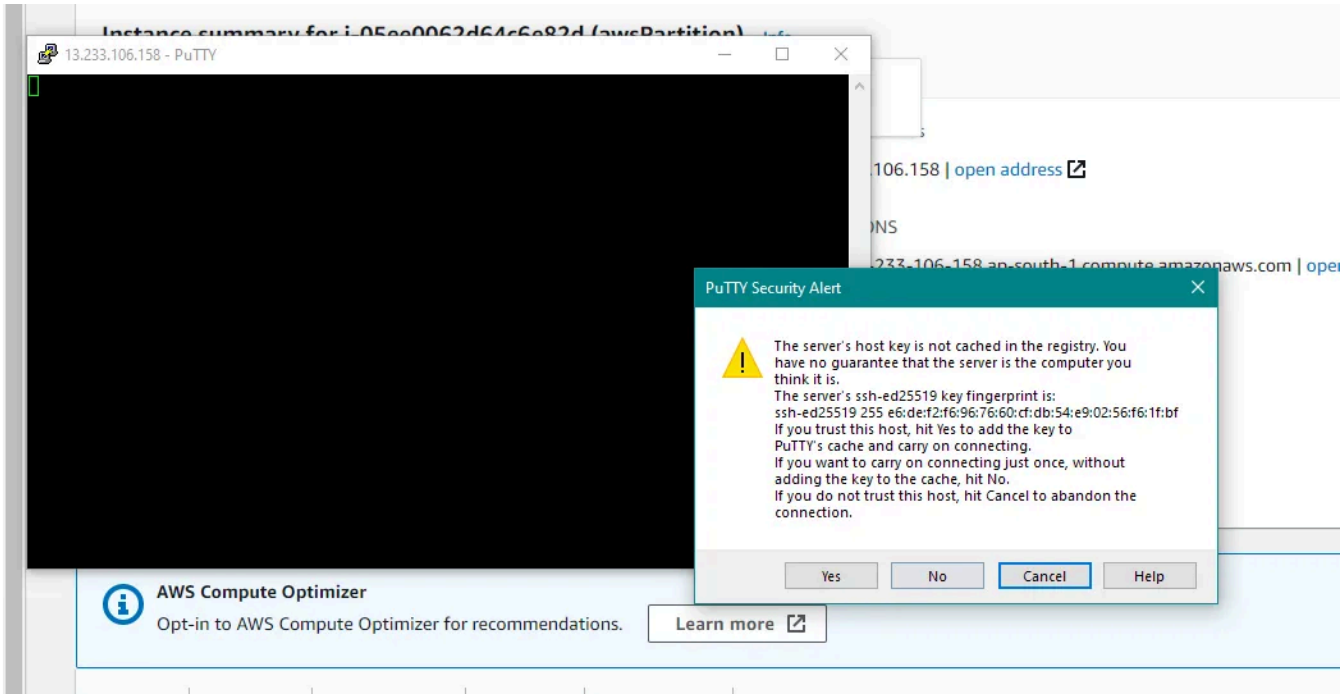


Launching An Instance

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.

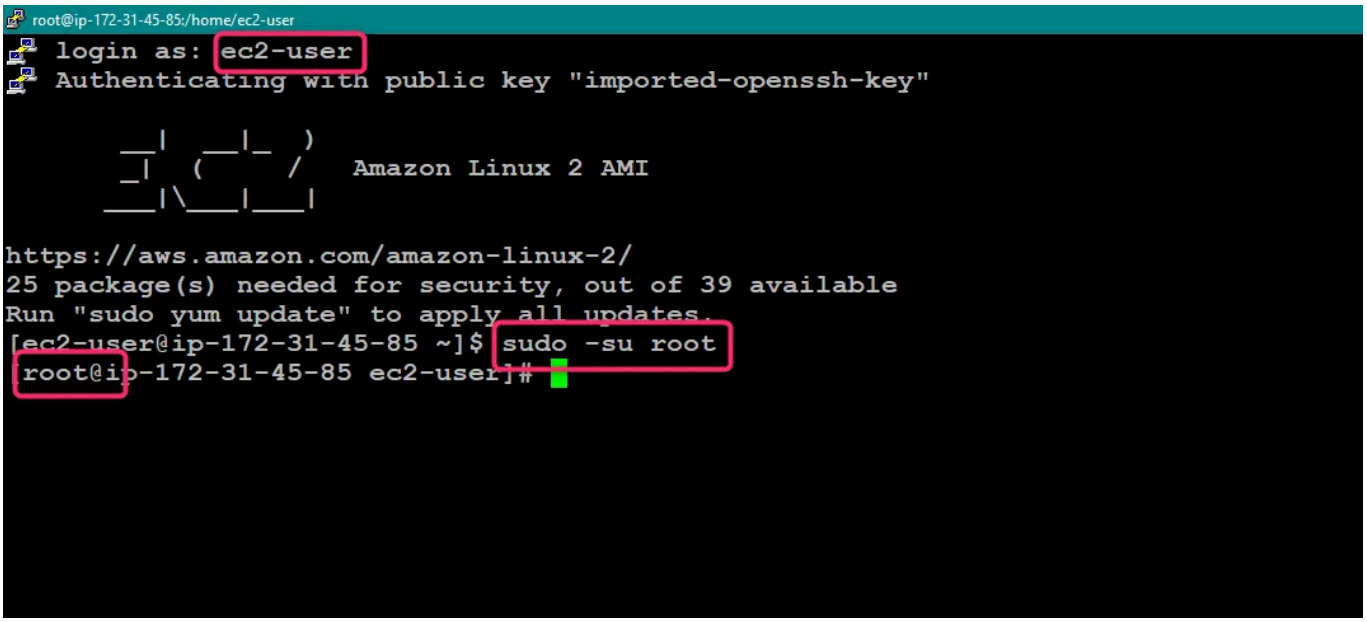


We open our Putty app and then copied the instance's public IP, and then convert the .ppk key pair file given by AWS into a .pem file using the application PuTTYgen. Then we add that file by going in SSH and Auth and then uploading it there, and we open SSH connection.



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:

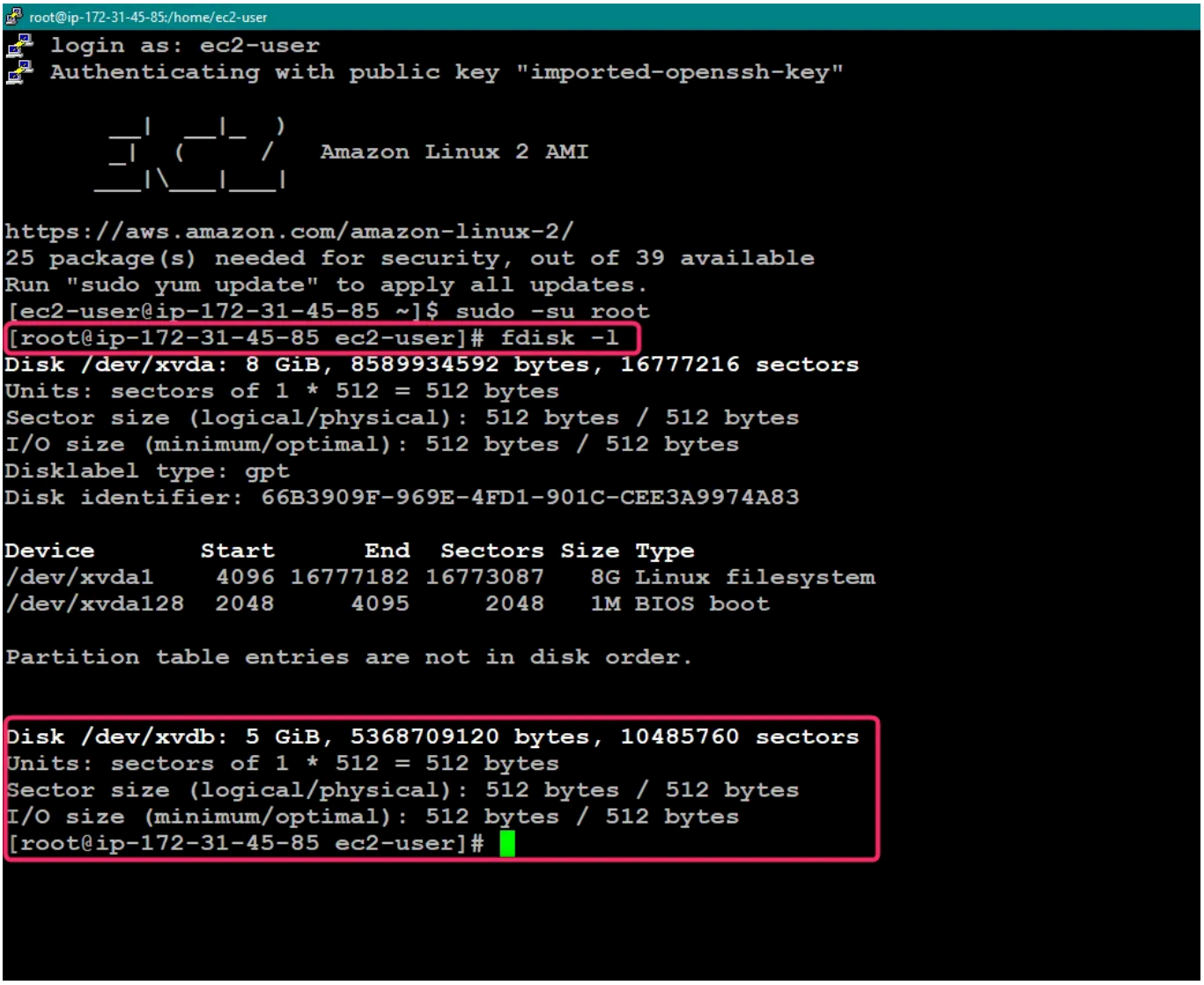
```
$sudo -su root
```



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.



Mounting EBS volume on our Instance

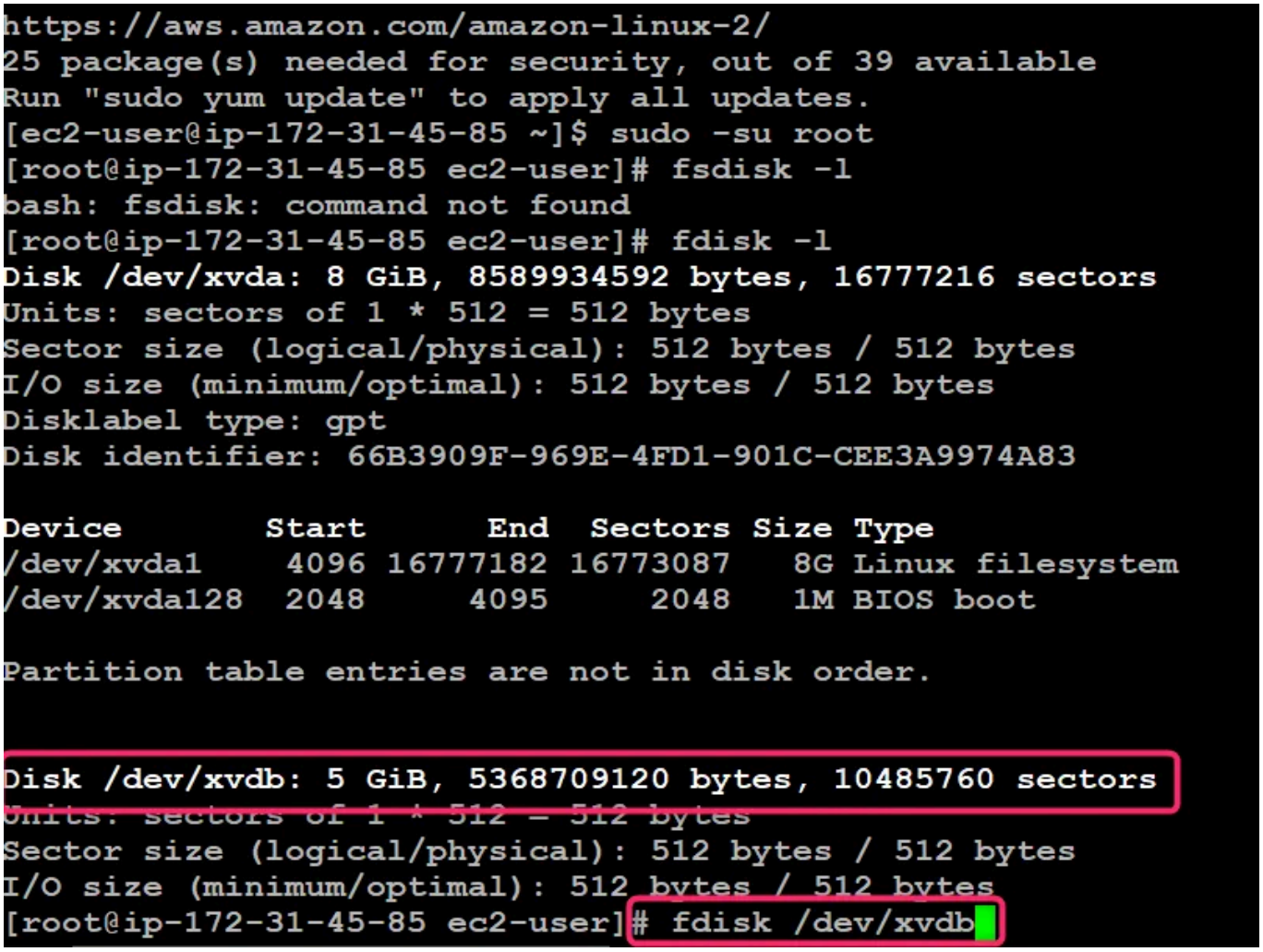
We can see our instance is also launched, and EBS storage is also attached to it now; we will mount it to the instance. As the topic is about partition, we will attach a part of, i.e., 3GB of the storage from the EBS volume, and then the same way, we can attach the remaining storage.

Mounting can be completed in 3 easy steps:

Step 1 Disk Partition

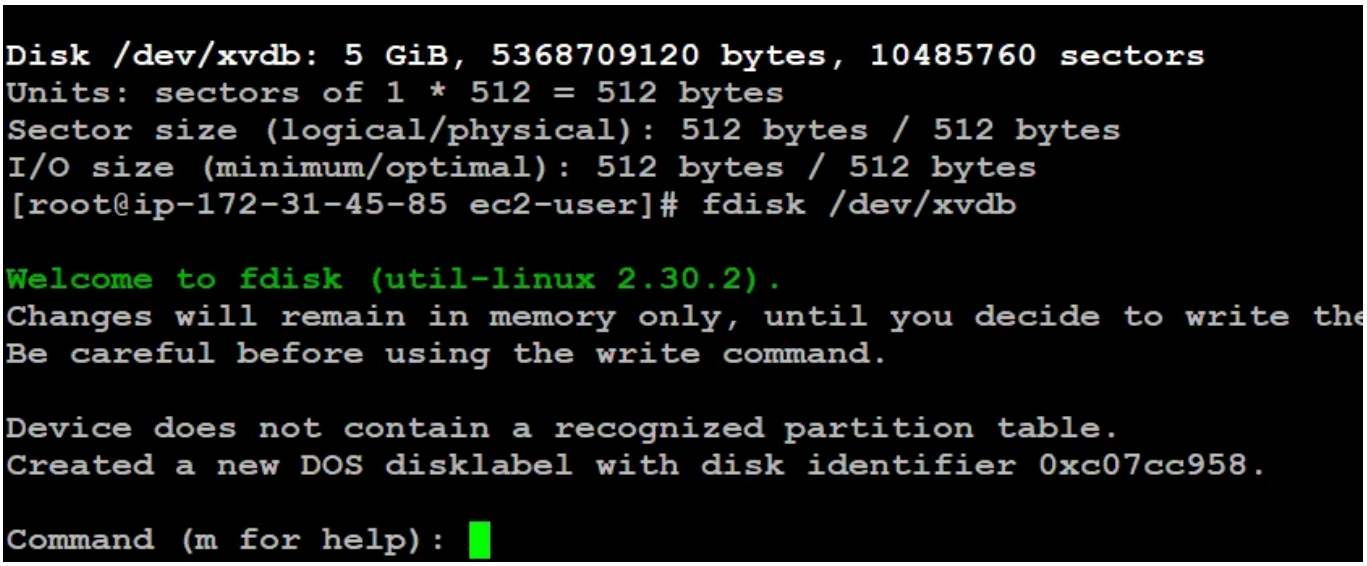
First, we need to divide the storage from the main storage and get how much storage we need. Hard disk space is divided using sectors, and 1 sector is equal to 512 bytes. So we need to divide the storage using sectors. We get inside that disk using the command:

```
$fdisk <diskname>
```



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.



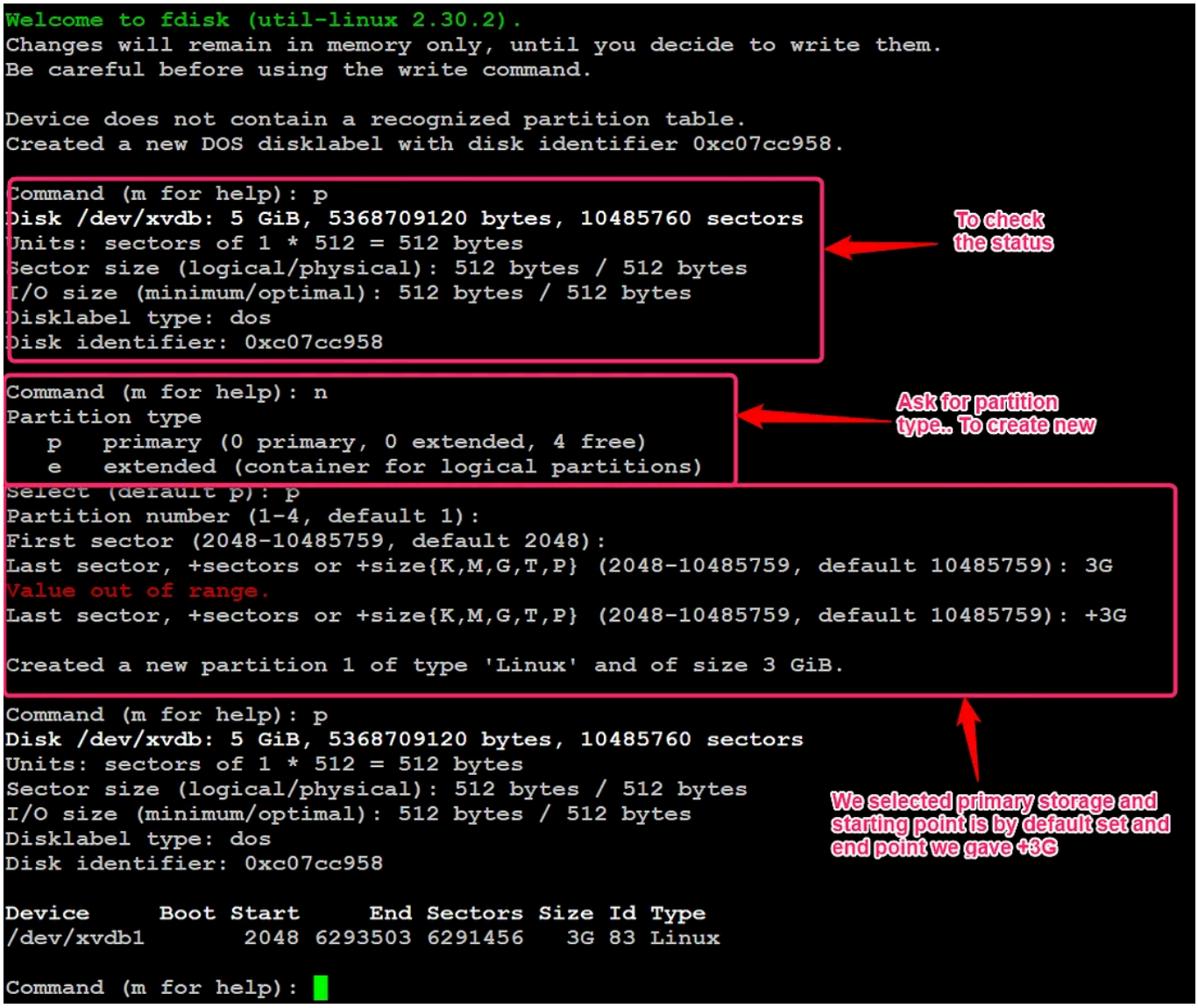
As we are inside the disk, we can use the various command to word with the disk:

- 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

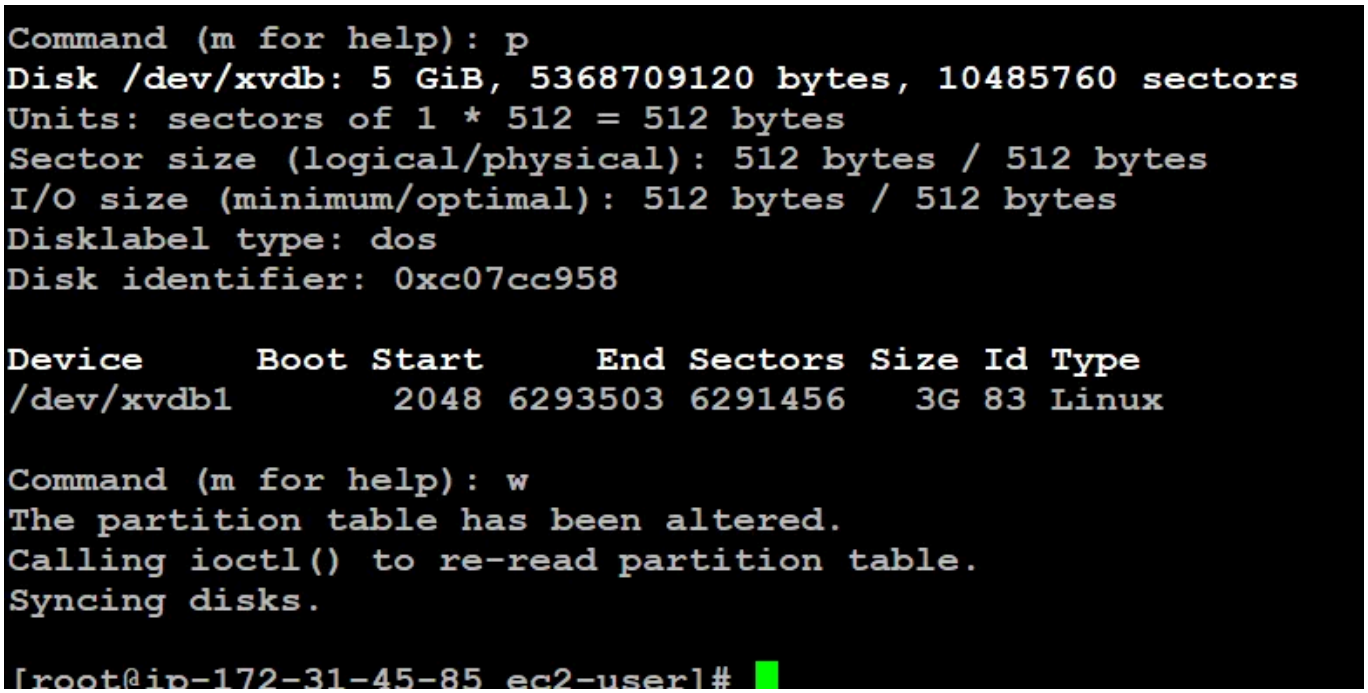
We check the status using command p and start creating a new partition using command n.

```
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.



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



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.

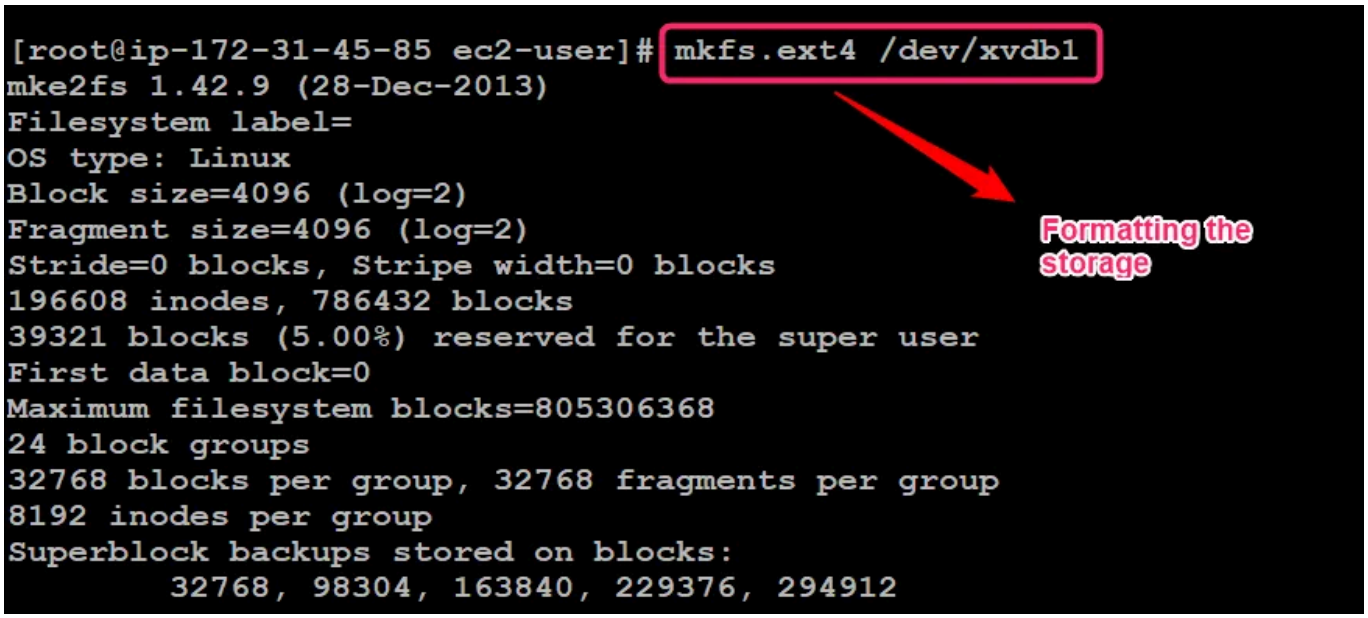
Now to format the device, we can use the command.

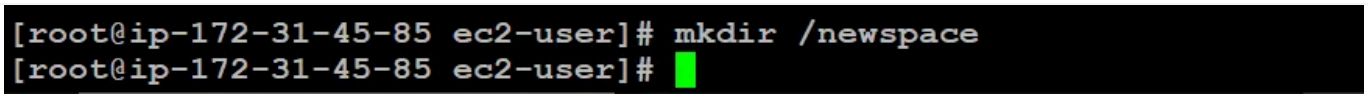
```
$ mkfs.ext4 <deviceName>
```

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.

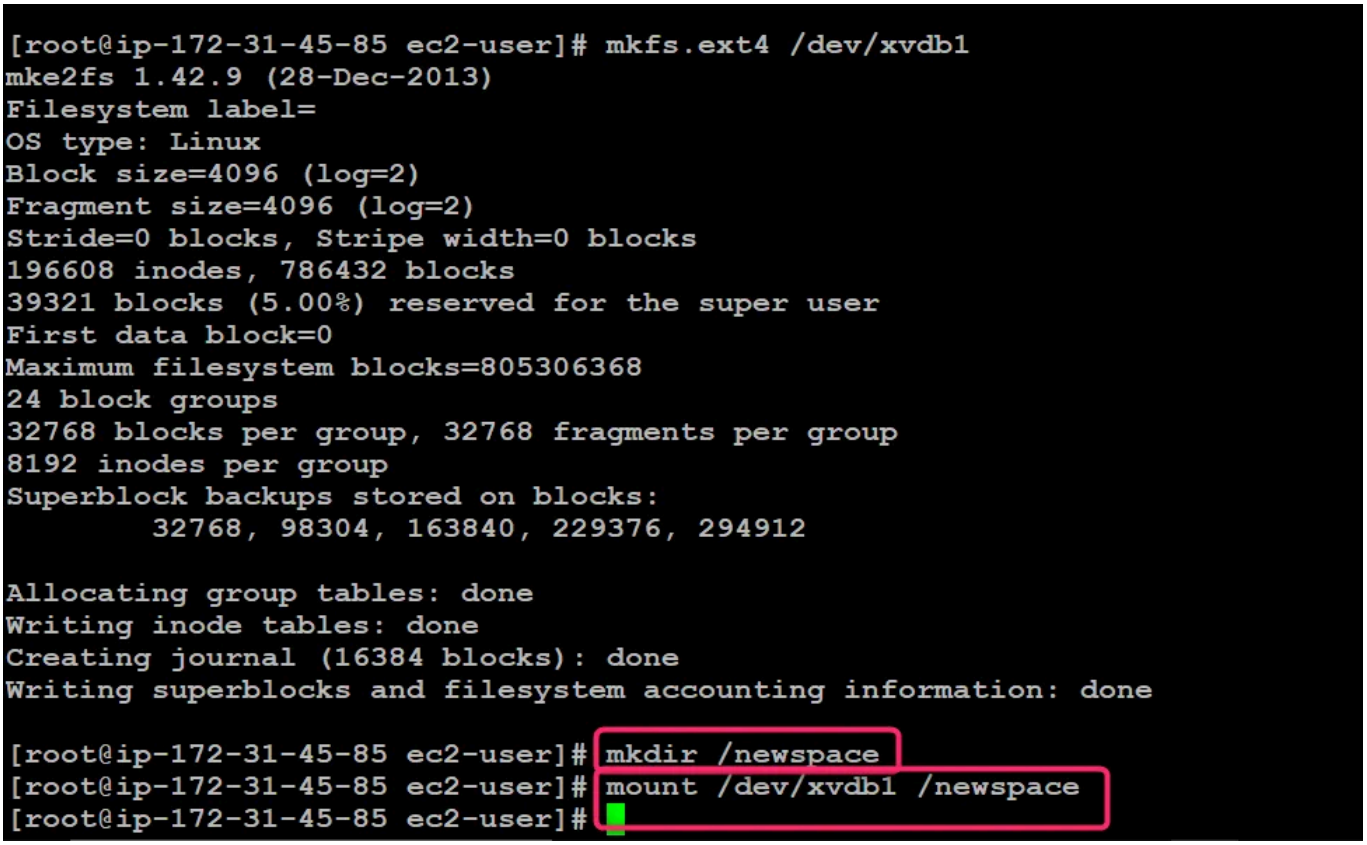




Step 4 Mounting

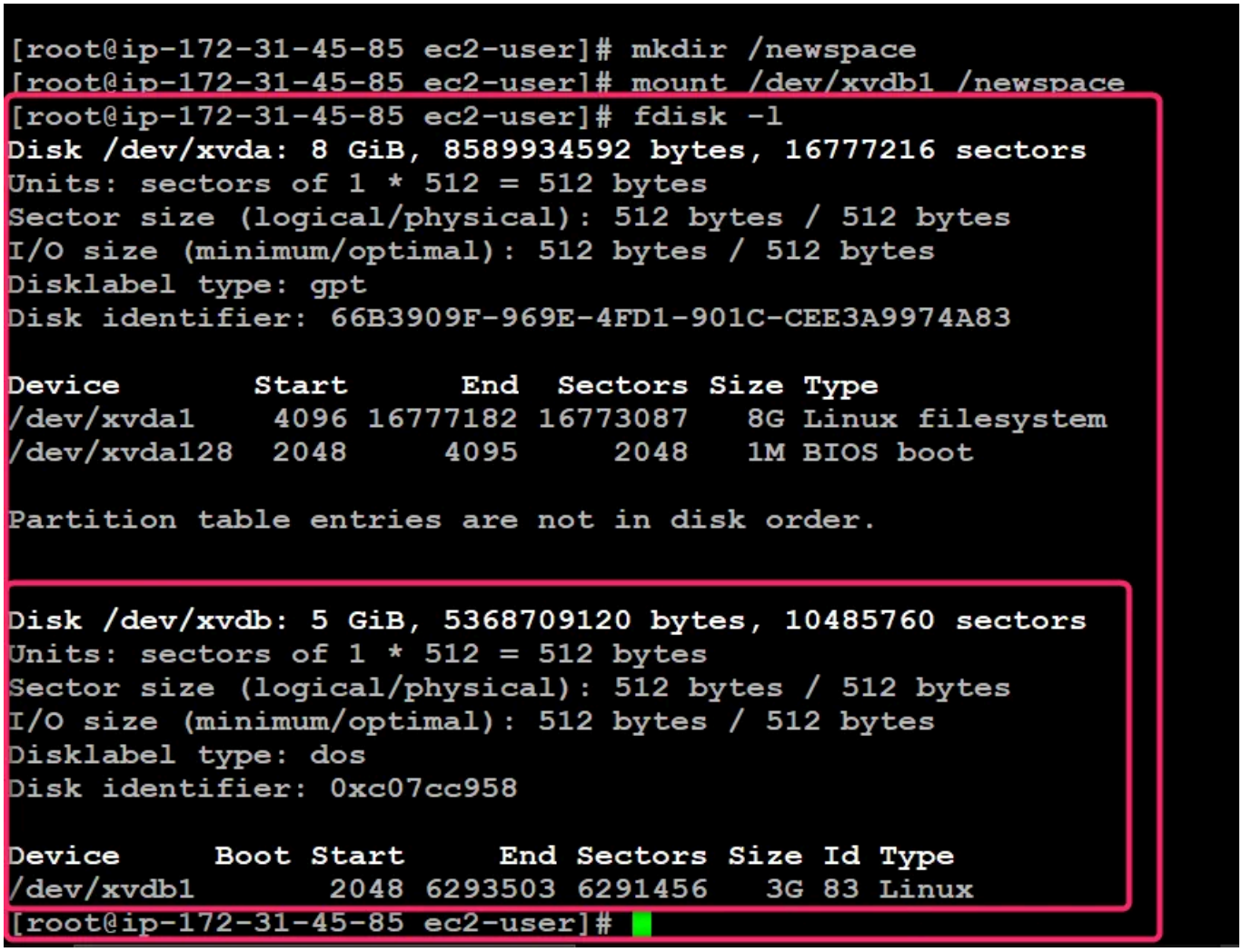
Now, as we have created a partition and formatted the space, we need to mount it. In OS, we can access only files and directory. We need any file or directory to access our storage device, so we create a new directory in our system and mount that storage device in that directory. The command is

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

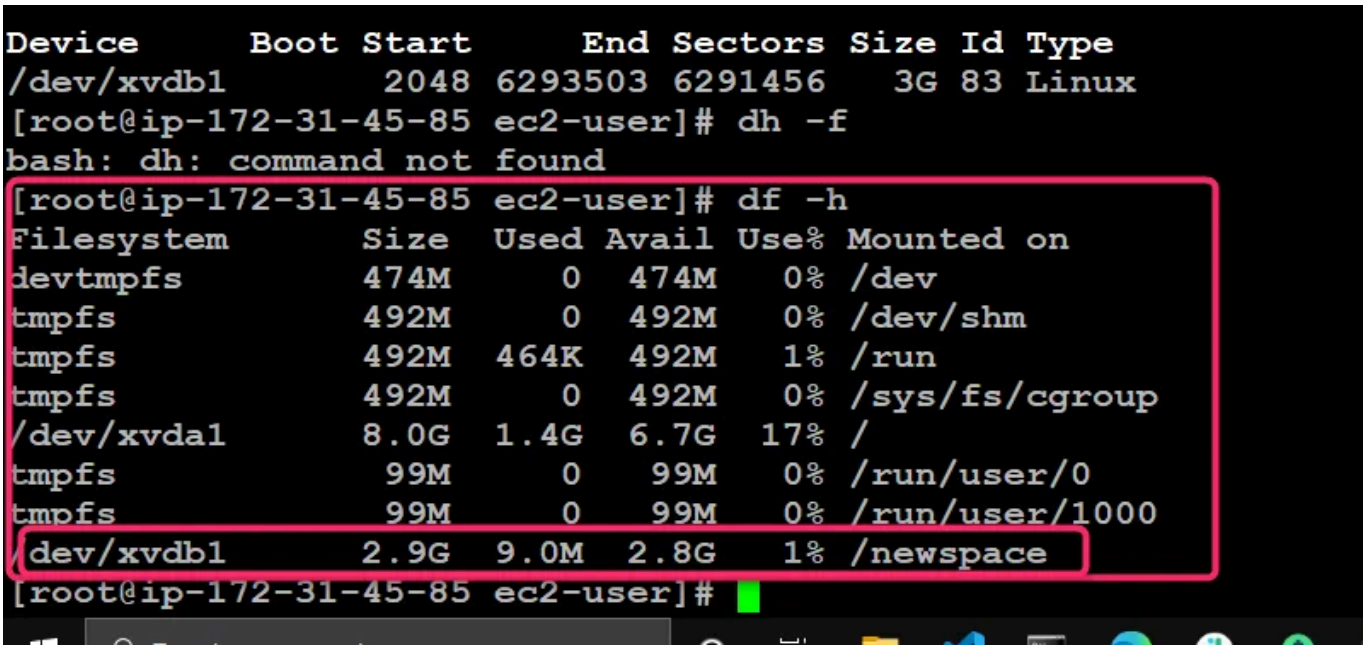


In Windows, the OS is stored in C:/ directory storage, and in Linux, the OS is stored in / “root” directory.

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.



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



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