Manage basic storage

Linux #redhat #storage #filesystem

Storage

3 types of storage that are attached to your system

- Local storage
  - Come pre-built, and pre-configures inside of your computer
- SAN (Storage Area Network)
  - Attaches to your computer through a fiber cable, fiber cables are now coming in with 8G of speed and 16 gigabit of speed.
- NAS (Network Attached Storage)
  - Anything that attaches to your computer through NFS or through an IP, or through a CAT5 cable. Windows Samba is a common example of NAS.

Disk Partition

Commands for disk partition

- `df` - Tells you your disk information, how much disk is used, where is it partitioned to?
- `fdisk` - Tells you how much the disk total size is and how many partitions has been created out of the disk.

From Monitor and manage Linux processes
Example using `df` command:

```
[user@localhost ~]$ df -h
```

- The `-h` option stands for "human readable" and specifies to change the units to be more readable to humans.

Output:

```
Filesystem            Size  Used Avail Use% Mounted on
devtmpfs              4.0M     0  4.0M   0% /dev
tmpfs                 5.1G     0  5.1G   0% /dev/shm
tmpfs                 2.1G  9.2M  2.1G   1% /run
/dev/mapper/rhel-root  17G  7.6G  9.5G  45% /
/dev/sda1             960M  304M  657M  32% /boot
tmpfs                 1.1G  100K  1.1G   1% /run/user/1000
```

- Please note: Your output might be different because your Linux system is configured with different disk space and a different volume name

From Access Linux files systems

The `fdisk` command

- You cannot run `fdisk` by itself (you have to specify an option)

Example using `fdisk` command:

```
[root@localhost ~]# fdisk -l
```

- The `-l` option stands for "list" and specifies to list the partition tables for the specified devices and then exit.

Output:

```
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: VBOX HARDDISK
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: 0xc8a29178

Device     Boot   Start      End  Sectors Size Id Type
/dev/sda1  *       2048  2099199  2097152   1G 83 Linux
/dev/sda2       2099200 41943039 39843840  19G 8e Linux LVM


Disk /dev/mapper/rhel-root: 17 GiB, 18249416704 bytes, 35643392 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


Disk /dev/mapper/rhel-swap: 2 GiB, 2147483648 bytes, 4194304 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
```

- It will list all your filesystem and where is it mounted.
- Why is there LVM on the 19G (`/dev/sda2`) drive? and why not on the 1G one (`/dev/sda1`)?
  - It's because sda1 has been mounted on `/boot`
    - You can verify this by running `df -h` and see where the `/dev/sda1` disk is mounted on.
    - For `/boot` or sometimes the swap you do not need an LVM filesystem.
- LVM allows you to create redundancy within your filesystem.
- Note the 20G disk is divided into two subdisks: a 17G disk for root, and a 2G disk for swap.

To create a new disk:

- Let's say you attach a new disk to your Linux OS, then you have to come here and run `fdisk` and specify the the name of the disk.
  Example using `fdisk` command:

```
[root@localhost ~]# fdisk /dev/sdb
```

- As the name `/dev/sda` is already taken (that is our principal disk), we can use the name `/dev/sdb`.
- When running this command it will give you more options on how you could partition that disk.

Adding Disk and Creating Partition

- Purpose? = Out of Space, Additional Apps, etc.
  - Maybe one partition is required for one application and the other partition is required for another application
- Commands for disk partition
  - `df`
  - `fdisk`

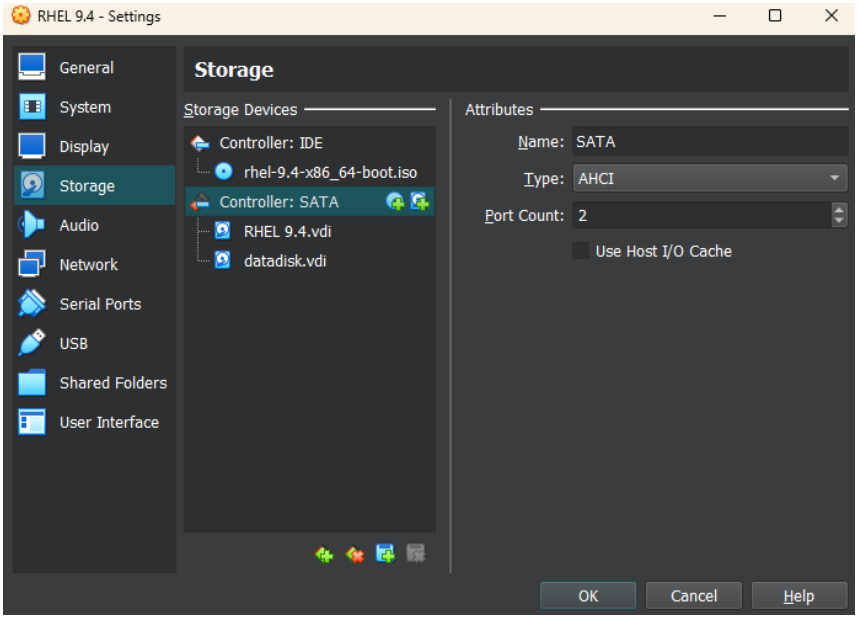
  \

Adding space in your VM

- Since for this note we do not have a physical server that we are working on, on which we would of course have to pull the server out, add a new disk if you have a slot and then bring the system up. But now since we are working on a virtual environment it's a lot easier to add a disk.

Steps to add a disk to your VM:

1. Power off your VM if running
2. Take a snapshot of your VM before adding a disk
3. Go to Settings -> Storage
4. Click on Controller: SATA
5. Click on add new storage attachment
6. Select Add Hard Disk
7. Click on create new disk
8. Leave the default VDI (VirtualBox Disk Image)
9. Leave Dynamically allocated default or if there is no option to leave this checked then check Pre-allocate Full Size instead
10. Pick a name for the disk (anything will work)
11. Select how much space you want to add
12. Click on Create or Finish
13. Once you create you'll see you have a new disk, now click OK, or Choose while selecting the new disk



- After adding a new disk your VM storage should look like this.
- Note the name of the new disk is "datadisk.vdi"

Create a partition with the `fdisk` utility

1. **Become root**
2. **Run** `fdisk -l`

Example using `fdisk` *command:*

```
[root@localhost ~]# fdisk -l
```

Output:

```
Disk /dev/sda: 20 GiB, 21474836480 bytes, 41943040 sectors
Disk model: VBOX HARDDISK
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: 0xc8a29178

Device     Boot   Start      End  Sectors Size Id Type
/dev/sda1  *       2048  2099199  2097152   1G 83 Linux
/dev/sda2       2099200 41943039 39843840  19G 8e Linux LVM


Disk /dev/sdb: 2 GiB, 2147483648 bytes, 4194304 sectors
Disk model: VBOX HARDDISK
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes


Disk /dev/mapper/rhel-root: 17 GiB, 18249416704 bytes, 35643392 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


Disk /dev/mapper/rhel-swap: 2 GiB, 2147483648 bytes, 4194304 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
```

- You'll notice the first disk that we added while we were building the system, this is for 20G and it is `/dev/sda`
  - It has two partitions, one for swap, and one for the regular Linux LVM
- The second disk that we added is right below the partitions from the first disk. You notice this disk has 2147 megabytes which is about 2G and has unite sectors and all that information there.
- The last two paragraphs of information is about the partition information about the first disk (rhel-root and rhel-swap partitions).

3. **Enter the `fdisk` utility**

Example using `fdisk` command:

```
[root@localhost ~]# fdisk /dev/sdb
```

- Note `/dev/sdb` is the name of the 2G disk we previously added to our VM

Output:

```


Welcome to fdisk (util-linux 2.37.4).
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 0x33fb7333.

Command (m for help):
```

- When you hit enter you'll notice it will bring up to the `fdisk` menu or portal.

Example using `m` command:

```
Command (m for help): m

Help:

  DOS (MBR)
   a   toggle a bootable flag
   b   edit nested BSD disklabel
   c   toggle the dos compatibility flag

  Generic
   d   delete a partition
   F   list free unpartitioned space
   l   list known partition types
   n   add a new partition
```

```
   p   print the partition table
   t   change a partition type
   v   verify the partition table
   i   print information about a partition

  Misc
   m   print this menu
   u   change display/entry units
   x   extra functionality (experts only)

  Script
   I   load disk layout from sfdisk script file
   O   dump disk layout to sfdisk script file

  Save & Exit
   w   write table to disk and exit
   q   quit without saving changes

  Create a new label
   g   create a new empty GPT partition table
   G   create a new empty SGI (IRIX) partition table
   o   create a new empty DOS partition table
   s   create a new empty Sun partition table
```

- Here is the list of different commands we have available to use
- The one that will be useful in this case is `n` because we are creating a new partition

Example using `n` command:

```
Command (m for help): n
   p   primary (0 primary, 0 extended, 4 free)
   e   extended (container for logical partitions)
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-4194303, default 2048):
Last sector, +/-sectors or +/-size{K,M,G,T,P} (2048-4194303, default 4194303):

Created a new partition 1 of type 'Linux' and of size 2 GiB.

Command (m for help):
```

- The first thing this command does is ask us what will be the type of the partition, primary or extended?
  - We are doing this with a new disk so we can select `p` for primary
- Then it asks for the primary partition number
  - We can leave it like that and hit Enter or type 1 (most of the time we want or primary partition to be number 1)
- Then it asks for the first sector
  - You can also leave it default to 2048
- Then it asks for the last sector
  - This is also fine, you can leave it default to 4194303
  - If you want to create 2 partitions from this disk, and you want each one to be 1G, then you could do `+1G` (type this at the moment that the Last sector question is prompted)
  - If you want the entire disk space for 1 partition you can just hit Enter.
- Now it's done. Created a new partition 1 of type 'Linux' and of size 2GiB.
  - But it will not be created until we enter `w` to write changes.

Example using `w` command:

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

- Now it IS done.

Example using `fdisk` command:

```
[root@localhost ~]# fdisk -l
```

- Run `fdisk -l` to verify the changes in the partitions of the new disk

Output:

```
...
Disk /dev/sdb: 2 GiB, 2147483648 bytes, 4194304 sectors
Disk model: VBOX HARDDISK
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: 0x33fb7333

Device     Boot Start     End Sectors Size Id Type
/dev/sdb1        2048 4194303 4192256   2G 83 Linux
...
```

- Now you can see the `/dev/sdb1` partition that you just created with the `fdisk` utility.\
- Automatically when you create one partition, the first partition on a disk, it assigns the number one.
- Now the volume is created but now we have to assign a filesystem type.
  - The latest filesystem that we have in Linux is 'xfs'.

Make a filesystem

Example using `mkfs` command:

```
[root@localhost ~]# mkfs.xfs /dev/sdb1
```

- The `mkfs` commands stands for "make filesystem"
- The `.xfs` extension specifies that the new filesystem to be created will be of the type 'xfs'
- Then a valome has to be specified (use the volume that you recently created)

Output:

```
meta-data=/dev/sdb1              isize=512    agcount=4, agsize=131008 blks
         =                       sectsz=512   attr=2, projid32bit=1
```

```
         =                       crc=1        finobt=1, sparse=1, rmapbt=0
         =                       reflink=1    bigtime=1 inobtcount=1 nrext64=0
data     =                       bsize=4096   blocks=524032, imaxpct=25
         =                       sunit=0      swidth=0 blks
naming   =version 2              bsize=4096   ascii-ci=0, ftype=1
log      =internal log           bsize=4096   blocks=16384, version=2
         =                       sectsz=512   sunit=0 blks, lazy-count=1
realtime =none                   extsz=4096   blocks=0, rtextents=0
```

- When you run this line you'll get all that message with all the details. how the file system created, what are the other information, size sector, and a whole bunch of stuff.
- Now the filesystem is created, but now we have to mount it to a certain mount point.

Mount the volume

- Create a directory under root and call it "data"
  - `# mkdir /data`
- Use the `mount` command to mount the volume into that new directory

Example using `mount` command:

```
[root@localhost ~]# mount /dev/sdb1 /data
```

- We are mounting that partition that we created to out filesystem, to our directory that we just created as `/data`
- How do you verify the volume was mounted
  - run `df -h` to list the filesystems

Output from `df -h`:

```
Filesystem             Size  Used Avail Use% Mounted on
devtmpfs               4.0M     0  4.0M   0% /dev
tmpfs                  5.1G     0  5.1G   0% /dev/shm
tmpfs                  2.1G  9.2M  2.1G   1% /run
/dev/mapper/rhel-root   17G  7.2G  9.8G  43% /
/dev/sda1              960M  411M  550M  43% /boot
tmpfs                  1.1G   96K  1.1G   1% /run/user/1000
/dev/sr0               991M  991M     0 100% /run/media/mmarin/RHEL-9-4-0-BaseOS-x86_64
/dev/sdb1              2.0G   47M  1.9G   3% /data
```

- Now you can see the new `/dev/sdb1` filesystem from the partition that we just created
- Now we could write to it whatever that we wanted to do.
- Now every time your system reboots, it will not mount this. So how can you make it enabled that it will get mounted at a system reboot?

Enable to mount the volume (on boot)

- You have to add an entry into the `fstab` file.
  - `fstab` has all the information about your partitions.

Example using `vi` command:

```
[root@localhost ~]# vi /etc/fstab
```

- Opening the file editor with `vi` to edit the `fstab` file

File editor:

```
#
# /etc/fstab
# Created by anaconda on Mon Jun  3 18:07:09 2024
#
# Accessible filesystems, by reference, are maintained under '/dev/disk/'.
# See man pages fstab(5), findfs(8), mount(8) and/or blkid(8) for more info.
#
# After editing this file, run 'systemctl daemon-reload' to update systemd
# units generated from this file.
#
/dev/mapper/rhel-root   /                       xfs     defaults        0 0
UUID=cd303319-8ed9-4d5b-8812-88970a2fa4ab /boot                   xfs     defaults        0 0
/dev/mapper/rhel-swap   none                    swap    defaults        0 0

/dev/sdb1       /data   xfs     defaults        0       0
```

- Come down all the way to the bottom, hit O to add a new line then type:
  - `/dev/sdb1 /data xfs defaults 0 0` (every fields is separated by a Tab key)
    - First is the partition
    - Next is where it is being mounted to
    - Then the type of filesystem for that partition
    - Then the rest leave that default with the `defaults` keyword
    - Then add a 0
    - Finally add another 0
      - These are the options that it would run filesystem, check when the system boots or not.
- Note that if you made a mistake you will have a problem booting your system. Make sure you are typing it correctly

Verify it was mounted on boot

- After adding the previous line to the `fstab` file you can run `init 6`
  - The `init` command is a systemd system and service manager
    - The `6` option specifies to automount units provide automount capabilities, for on demand mounting of file systems as well as parallelized boot-up.
  - When you run this command, your system will reboot
- Run `df -h` to check if the partition was mounted during boot
  - If it appears the process was completed successfully and you can start using that partition

In case you want to unmount it

- Use the `umount` command and select the directory to which the volume is mounted

Example using `umount` command:

```
[root@localhost ~]# umount /data
```

- Remember that `/data` was the directory that we created in order to mount our `/dev/sdb1` partition
- Now if you run `df -h` the `/dev/sdb1` filesystem will no longer appear

If you want to mount the volume again
Example using `mount` command:

```
[root@localhost ~]# mount -a
```

- The `-a` option stands for "all" and specifies to mount all filesystems (of the given types) mentioned in `/etc/fstab` (except for those whose line contains the `noauto` keyword).