

# LVM & RAID

## I. Soft RAID (Linux raid auto)

### I.1. Hiểu về các chuẩn RAID cứng

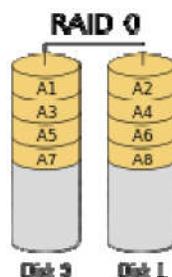
RAID (Redundant Arrays of Inexpensive Disks) là hình thức ghép nhiều ổ đĩa cứng vật lý thành một hệ thống ổ đĩa cứng logic có chức năng tăng tốc độ đọc/ghi dữ liệu hoặc nhằm tăng thêm sự an toàn của dữ liệu chứa trên hệ thống đĩa hoặc kết hợp cả hai yếu tố trên.

Về cơ bản, các ổ cứng logic này xuất hiện như những ổ cứng vật lý trong hệ điều hành.

Lần đầu tiên RAID được phát triển năm 1987 tại trường Đại học California tại Berkeley (Hoa Kỳ) với những đặc điểm chỉ ghép các phần đĩa cứng nhỏ hơn thông qua phần mềm để tạo ra một hệ thống đĩa dung lượng lớn hơn thay thế cho các ổ cứng dung lượng lớn giá đắt thời bấy giờ.

Mặc dù hiện nay không tồn tại nữa, nhưng Hội đồng tư vấn phát triển RAID (RAID Advisory Board: Viết tắt là RAB) đã ra thành lập tháng 7 năm 1992 để định hướng, lập ra các tiêu chuẩn, định dạng cho RAID. RAB đã phân ra các loại cấp độ RAID (level), các tiêu chuẩn phần cứng sử dụng RAID. RAB đã phân ra 7 loại cấp độ RAID từ cấp độ 0 đến cấp độ 6.

#### I.1.1. RAID 0 - Stripping



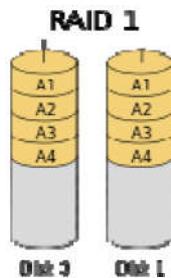
RAID 0 cần ít nhất 2 ổ đĩa. Tóm quát ta có  $n$  đĩa ( $n \geq 2$ ) và các đĩa là cùng loại.

Dữ liệu sẽ được chia ra nhiều phần bằng nhau để lưu trên từng đĩa. Như vậy mỗi đĩa sẽ chứa  $1/n$  dữ liệu.

- **Ưu điểm:** - Tăng dung lượng đĩa: bên ngoài thấy 1 HDD có dung lượng gấp  $n$  lần ổ đĩa đơn. - Tăng tốc độ đọc/ghi đĩa: mỗi đĩa chỉ cần phải đọc/ghi  $1/n$  lượng dữ liệu được yêu cầu. Lý thuyết thì tốc độ sẽ tăng  $n$  lần.

- **Nhược điểm:** - Tính an toàn thấp. Nếu một đĩa bị hư thì dữ liệu trên tất cả các đĩa còn lại sẽ không còn sử dụng được. Xác suất để mất dữ liệu sẽ tăng n lần so với dùng ổ đĩa đơn.

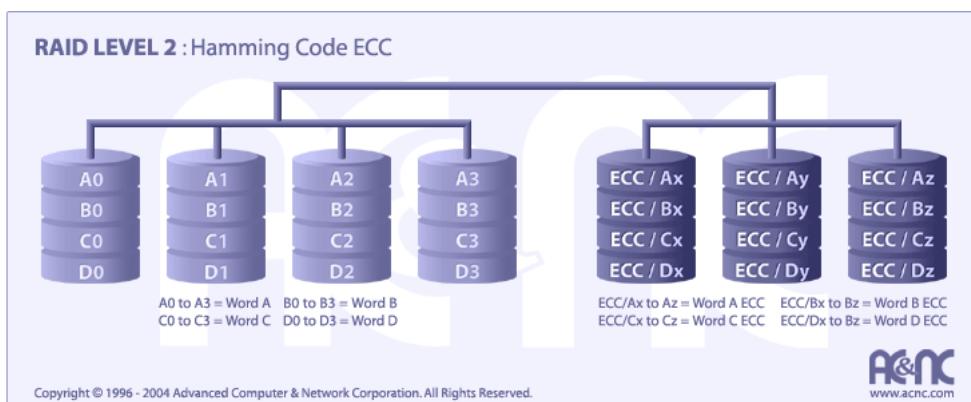
### I.1.2. RAID 1 - Mirror



Đây là dạng RAID cơ bản nhất có khả năng đảm bảo an toàn dữ liệu. Cũng giống như RAID 0, RAID 1 đòi hỏi ít nhất hai đĩa cứng để làm việc. Dữ liệu được ghi vào 2 ổ giống hệt nhau (Mirroring). Trong trường hợp một ổ bị trục trặc, ổ còn lại sẽ tiếp tục hoạt động bình thường.

- **Ưu điểm:** Có thể thay thế ổ đĩa bị hỏng mà không phải lo lắng đến vấn đề thông tin thất lạc. Đối với RAID 1, hiệu năng không phải là yếu tố hàng đầu nên chẳng có gì ngạc nhiên nếu nó không phải là lựa chọn số một cho những người say mê tốc độ. Tuy nhiên đối với những nhà quản trị mạng hoặc những ai phải quản lý nhiều thông tin quan trọng thì hệ thống RAID 1 là thứ không thể thiếu. Dung lượng cuối cùng của hệ thống RAID 1 bằng dung lượng của ổ đơn (hai ổ 80GB chạy RAID 1 sẽ cho hệ thống nhìn thấy duy nhất một ổ RAID 80GB).
- **Nhược điểm:** Chi phí đắt đỏ (Hiệu suất của hệ thống không đổi)

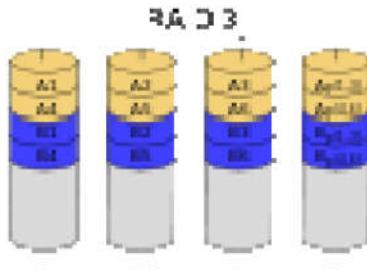
### I.1.3. RAID 2



RAID 2 gồm hai cụm ổ đĩa, cụm thứ nhất chứa các dữ liệu được phân tách giống như là RAID 0, cụm thứ hai chứa các mã ECC dành cho sửa chữa lỗi ở cụm thứ nhất. Sự hoạt

động của các ổ đĩa ở RAID 2 là đồng thời để đảm bảo rằng các dữ liệu được đọc đúng, chính do vậy chúng không hiệu quả bằng một số loại RAID khác nên ít được sử dụng.

#### I.1.4. RAID 3

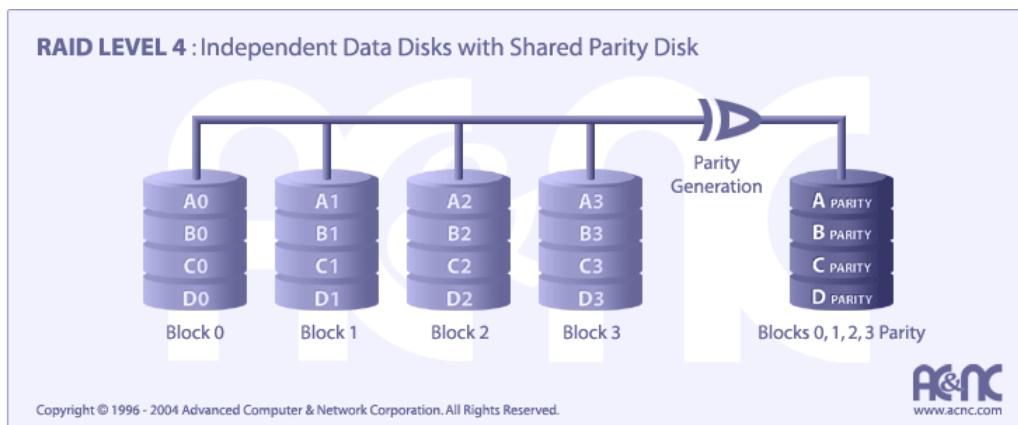


RAID 3 là sự cải tiến của RAID 0 nhưng có thêm (ít nhất) một ổ đĩa cứng chứa thông tin có thể khôi phục lại dữ liệu đã hư hỏng của các ổ đĩa cứng RAID 0.

Giả sử dữ liệu A được phân tách thành 3 phần A1, A2, A3 (Xem hình minh họa RAID 3), khi đó dữ liệu được chia thành 3 phần chứa trên các ổ đĩa cứng 0, 1, 2 (giống như RAID 0). Phần ổ đĩa cứng thứ 3 chứa dữ liệu của tất cả để khôi phục dữ liệu có thể sẽ mất ở ổ đĩa cứng 0, 1, 2. Giả sử ổ đĩa cứng 1 hư hỏng, hệ thống vẫn hoạt động bình thường cho đến khi thay thế ổ đĩa cứng này. Sau khi gắn nóng ổ đĩa cứng mới, dữ liệu lại được khôi phục trở về ổ đĩa 1 như trước khi nó bị hư hỏng.

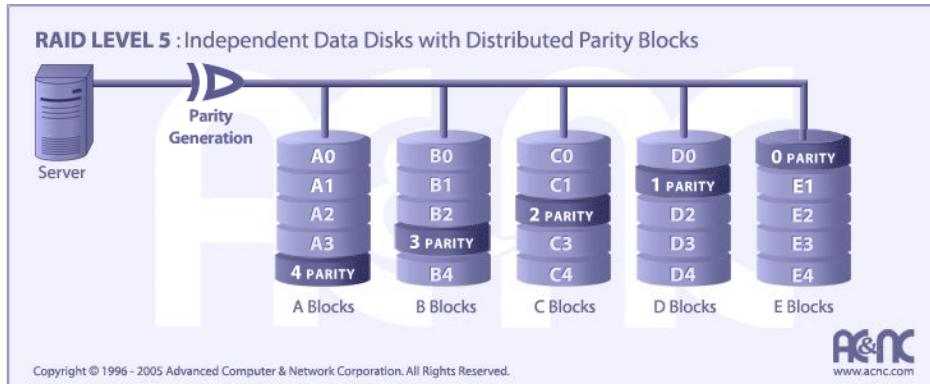
Yêu cầu tối thiểu của RAID 3 là có ít nhất 3 ổ đĩa cứng.

#### I.1.5. RAID 4



RAID 4 tương tự như RAID 3 nhưng ở một mức độ các khối dữ liệu lớn hơn chứ không phải đến từng byte. Chúng cũng yêu cầu tối thiểu 3 đĩa cứng (ít nhất hai đĩa dành cho chứa dữ liệu và ít nhất 1 đĩa dùng cho lưu trữ dữ liệu tổng thể)

### I.1.6. RAID 5



Đặc điểm của chuẩn này là dùng một phần không gian lưu trữ để lưu mã sửa sai.

Dữ liệu và bản sao lưu được chia lên tất cả các ổ cứng. Ví dụ có 8 đoạn dữ liệu được đánh số từ 1 đến 8 và 3 ổ đĩa cứng có dung lượng giống nhau. Đoạn dữ liệu số 1 và số 2 sẽ được ghi vào ổ đĩa 1 và 2 riêng rẽ, đoạn sao lưu của chúng được ghi vào ổ cứng 3. Đoạn số 3 và 4 được ghi vào ổ 1 và 3 với đoạn sao lưu tương ứng ghi vào ổ đĩa 2. Đoạn số 5, 6 ghi vào ổ đĩa 2 và 3, còn đoạn sao lưu được ghi vào ổ đĩa 1 và sau đó trích tự này lặp lại, đoạn số 7,8 được ghi vào ổ 1, 2 và đoạn sao lưu ghi vào ổ 3 như ban đầu. Như vậy RAID 5 vừa đảm bảo tốc độ có cải thiện, vừa giữ được tính an toàn cao. Dung lượng đĩa cứng cuối cùng bằng tổng dung lượng đĩa sử dụng trừ đi một ổ. Tức là nếu dùng 3 ổ 80GB thì dung lượng cuối cùng sẽ là 160GB.

Nhờ đó nếu hỏng một ổ cứng bất kỳ thì hệ thống vẫn hoạt động bình thường, tuy nhiên hệ thống sẽ lỗi nếu hỏng từ 2 ổ trở lên

Ưu điểm của chuẩn này là kết hợp ưu của hai chuẩn trước: an toàn hơn chuẩn 0, nhanh hơn chuẩn 1, nhược điểm cũng là kết hợp của hai chuẩn trên.

### I.1.7. RAID 6

Giống RAID 5 nhưng sử dụng gấp đôi không gian lưu trữ để lưu mã sửa sai nên đồng thời cho phép hỏng tối đa 2 ổ cứng.

## I.2. Các khái niệm khác

### I.2.1. Hot swap

Tráo đổi nóng là cơ chế cho phép thay “nóng” ổ cứng hỏng khi hệ thống đang hoạt động. Card RAID cho phép sinh lại dữ liệu hợp lệ trên ổ cứng mới thay ngay khi hệ thống vẫn đang hoạt động. Các chuẩn 1, 5, 6 hỗ trợ chức năng này.

### I.2.2. Active, Spare, Sync

Trên thực tế, thời gian đọc/ghi trên các ổ cứng chạy chuẩn 1, 5, 6 là tương đương nhau nên xác suất các ổ hỏng cùng lúc trong một khoảng thời gian gần là rất cao. Để đảm bảo tính an toàn, giảm thời gian/công sức quản trị (không cần lúc nào cũng theo dõi khay ổ cứng để kiểm tra tình trạng ổ), các chuẩn raid 1, 5, 6 có thêm khái niệm trạng thái hoạt động như sau:

- **Active:** Ổ cứng ở trạng thái này tham gia thực tế vào việc đọc/ghi dữ liệu, quyết định trực tiếp đến kích thước của ổ raid logic.
- **Spare:** Ổ cứng trong trạng thái ngủ/nằm chờ, không được cấp nguồn.
- **Sync:** Mỗi khi có một ổ active hỏng, ngay lập tức 1 ổ spare được kích hoạt theo dạng hot swap. Trước khi thực sự trở thành active nó cần được card raid chép dữ liệu hợp lệ lên. Quá trình này nhanh hay chậm tùy vào khối lượng dữ liệu thực.

### I.3. Soft RAID trong Linux

Hỗ trợ các chuẩn 0, 1, 5, 6 trong RAID cứng. Chỉ sử dụng lệnh **mdadm** để quản lý và file **/proc/mdstat** để chứa các thông tin real-time về các thiết bị raid. Tuy nhiên khác raid cứng là raid mềm cho phép ghép các phân vùng trên một hoặc nhiều ổ cứng khác nhau lại thành thiết bị raid (hoạt động ở mức phân vùng).

Các thiết bị raid sau khi được tạo có thể định dạng và ánh xạ sử dụng như các phân vùng thông thường.

- Sử dụng công cụ quản lý softRAID (**mdadm**) để nối 4 phân vùng mới tạo lại thành 1 phân vùng mới (**/dev/md1**) theo chuẩn RAID 6.

```
mdadm --help
mdadm --create --help
mdadm --create /dev/md1 --level=6 --raid-devices=4
/dev/hdc1 /dev/hdc2 /dev/sda1 /dev/sda2
cat /proc/mdstat
```

- Định dạng phân vùng mới theo chuẩn ext3 và ánh xạ vào thư mục **/raid**.

```
mkfs -t ext3 /dev/md1
mount -t ext3 /dev/md1 /raid
```

- Copy toàn bộ thư mục **/usr** vào **/raid** sau đó khai báo 1 phân vùng tham gia vào **/dev/md1** bị hỏng rồi loại bỏ nó ra khỏi hệ thống softRAID.

```
cp -r /usr /raid
df -h
mdadm --manage /dev/md1 --fail /dev/hdc1
cat /proc/mdstat
mdadm --manage /dev/md1 --remove /dev/hdc1
cat /proc/mdstat
```

- Lại cho phân vùng vừa bỏ ra vào trong hệ thống softRAID. Sau đó dùng lệnh watch "cat /proc/mdstat" để theo dõi quá trình rebuild tự động của hệ thống softRAID

```
mdadm --manage /dev/md1 --add /dev/hdc1  
watch "cat /proc/mdstat"
```

**Chú ý:** Sau khi bổ xung thêm ổ đĩa vào RAID (chỉ thêm vào với RAID 5, 6) sử dụng lệnh resize2fs để cập nhật lại RAID

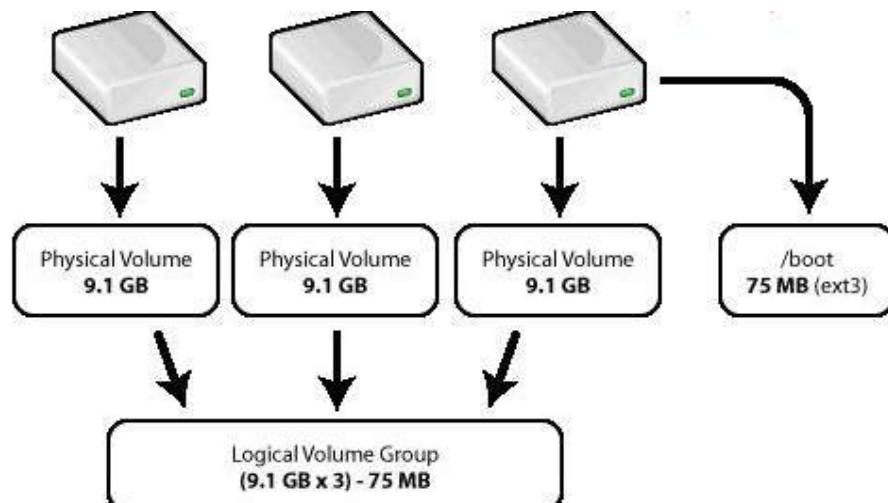
```
resize2fs /dev/md0
```

## II. LVM - Logical Volume Manager

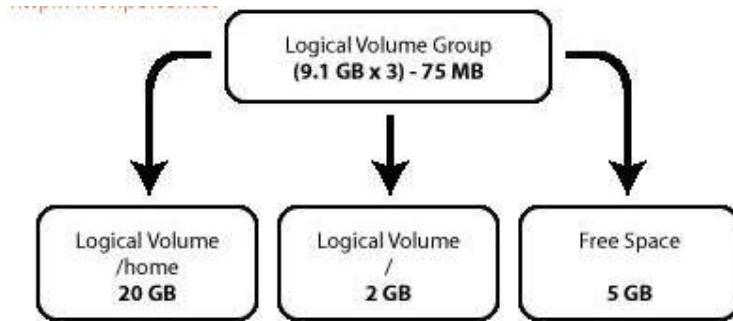
LVM là một phương pháp cho phép ấn định không gian đĩa cứng thành những Logical Volume khiếun cho việc thay đổi kích thước trở lên dễ dàng ( so với partition ). Với kỹ thuật **Logical Volume Manager** (LVM) có thể thay đổi kích thước mà không cần phải sửa lại partition table của OS. Điều này thực sự hữu ích với những trường hợp đã sử dụng hết phần bộ nhớ còn trống của partition và muốn mở rộng dung lượng của nó.

Một số khái niệm cơ bản sử dụng trong LVM

- **Physical Volume:** Là một cách gọi khác của partition trong kỹ thuật LVM, là những thành phần cơ bản được sử dụng bởi LVM. Một Physical Volume không thể mở rộng ra ngoài phạm vi một ổ đĩa.
- **Logical Volume Group:** Nhiều Physical Volume trên những ổ đĩa khác nhau được kết hợp lại thành một Logical Volume Group, với LVM Logical Volume Group được xem như một ổ đĩa ảo.



- **Logical Volumes:** Logical Volume Group được chia nhỏ thành nhiều Logical Volume, mỗi Logical Volume có ý nghĩa tương tự như partition. Nó được dùng cho các mount point và được format với những định dạng khác nhau như ext2, ext3 ...



khi dung lượng của Logical Volume được sử dụng hết có thể đưa thêm ổ đĩa mới bổ sung cho Logical Volume Group và do đó tăng được dung lượng của Logical Volume.

- **Physical Extent:** là một đại lượng thể hiện một khối dữ liệu dùng làm đơn vị tính dung lượng của Logical Volume

Một điểm cần lưu ý là boot loader không thể đọc **/boot** khi nó nằm trên Logical Volume Group. Do đó không thể sử dụng kỹ thuật LVM với /boot mount point.

### **II.1. Đăng ký sử dụng trong LVM với các phân vùng vật lý**

**pvcreate** <danh sách tên các phân vùng vật lý>

### **II.2. Quản lý các VG**

<b>vgcreate</b>	Tạo 1 VG mới từ các phân vùng vật lý đã đăng ký sử dụng
<b>vgdisplay</b>	Kiểm tra thông tin về các VG đã có
<b>vgextend</b>	Thêm một phân vùng vật lý mới vào VG đã có
<b>vgreduce</b>	Loại một phân vùng vật lý trong VG đã có nếu đủ chỗ trống
<b>vgrename</b>	Đổi tên VG
<b>vgremove</b>	Hủy VG

### **II.3. Quản lý các LV**

<b>lvcreate</b>	Tạo 1 LV mới trong 1 VG đã có
<b>lvdisplay</b>	Kiểm tra các thông tin về các LV đã có
<b>lvresize</b>	Thay đổi kích thước các LV (lưu ý về tham số <b>-r</b> và nếu giảm kích thước LV thì LV này phải umount trước đó nếu không sẽ bị hỏng hệ thống file)
<b>lvrename</b>	Đổi tên LV

<b>lvremove</b>	Hủy LV
-----------------	--------

## II.4. Định danh các LV

Đường dẫn tuyệt đối đến mỗi LV được xác định như sau:

```
/dev/mapper/<tên VG chứa LV>-<tên LV>
```

Đường dẫn này có thể truy cập theo một liên kết mềm khác như sau:

```
/dev/<tên VG chứa LV>/<tên LV>
```

## II.5. Ví dụ sử dụng LVM

Giả sử thư mục **/home** nằm trên partition **/dev/hde5** đã sử dụng hết không gian trống, cần phải tăng kích thước phần partition dành cho thư mục này bằng cách lắp thêm một ổ cứng mới **/dev/hdf**.

- Việc đầu tiên là backup toàn bộ dữ liệu trên thư mục **/home**, việc này có thể dễ dàng thực hiện bằng nhiều công cụ nhưng đơn giản nhất là dùng lệnh tar.

```
tar czf home.tar.gz /home/*
```

- Sau đó vào chế độ single user bằng lệnh.

```
init 1
```

- Gỡ bỏ ánh xạ tới thư mục **/home**.

```
umount /home
```

- Xem tóm tắt các partition trên HDH đang dùng.

```
fdisk -l /dev/hde
```

```
Disk /dev/hde: 4311 MB, 4311982080 bytes  
16 heads, 63 sectors/track, 8355 cylinders  
Units = cylinders of 1008 * 512 = 516096 bytes
```

```
Device Boot Start End Blocks Id System
```

```
/dev/hde1 1 4088 2060320+ fd Linux raid autodetect
```

```
/dev/hde2 4089 5713 819000 83 Linux
```

```
/dev/hde3 5714 6607 450576 83 Linux
```

```
/dev/hde4 6608 8355 880992 5 Extended
```

```
/dev/hde5 6608 7500 450040+ 83 Linux
```

- Thông tin trả về cho thấy partition cần sửa là dạng extend partition và nằm phía dưới cùng. Sửa lại partition type của partition này về loại **Linux LVM (8e)** bằng lệnh **fdisk**.

```
fdisk /dev/hde
```

```
The number of cylinders for this disk is set to 8355.
```

```
There is nothing wrong with that, but this is larger than
1024,
and could in certain setups cause problems with:
1) software that runs at boot time (e.g., old versions of
LILO)
2) booting and partitioning software from other OSs
(e.g., DOS FDISK, OS/2 FDISK)
```

Command (m for help):

Gõ **t** để thay đổi partition type, nhập giá trị **8e** để đặt partition này về dạng **LVM** **Linux**.

```
Command (m for help): t
```

```
Partition number (1-5): 5
```

```
Hex code (type L to list codes): 8e
```

```
Changed system type of partition 5 to 8e (Linux LVM)
```

Command (m for help):

Gõ **p** để xem lại thông tin partition table sau khi sửa đổi.

```
Command (m for help): p
```

```
Disk /dev/hde: 4311 MB, 4311982080 bytes
16 heads, 63 sectors/track, 8355 cylinders
Units = cylinders of 1008 * 512 = 516096 bytes
```

Device	Boot	Start	End	Blocks	Id	System
/dev/hde1	1	4088	2060320+	fd	Linux raid	autodetect
/dev/hde2	4089	5713	819000	83	Linux	
/dev/hde3	5714	6607	450576	83	Linux	
/dev/hde4	6608	8355	880992	5	Extended	
/dev/hde5	6608	7500	450040+	8e	Linux	LVM

Command (m for help): w

Gõ **w** để save thông tin partition table đã sửa đổi.

Làm tương tự với phần ổ cứng bạn đưa thêm vào

- Khởi tạo **Physical Volume** trên những partition vừa tạo để sử dụng cho LVM bằng lệnh **pvcreate**, lưu ý quá trình này sẽ xoá sạch những dữ liệu đã có trên các partition.

```
pvcreate /dev/hde5
```

```
        pvcreate -- physical volume "/dev/hde5" successfully
        created
pvcreate /dev/hdf1
        pvcreate -- physical volume "/dev/hdf1" successfully
        created
```

- Kiểm tra lại **Physical Volume** vừa tạo bằng lệnh xem thông tin về **Physical Volume**: **pvs** hoặc **pvdisplay**
- Thực hiện update thông tin cấu hình LVM vào file **/etc/lvmtab** bằng lệnh.

#### **vgscan**

```
        vgscan -- reading all physical volumes (this may take a
        while...)
```

- Tổng hợp 2 **physical volumes** thành một đơn vị duy nhất gọi là **Volume Group**, khi đó LVM sẽ làm cho OS nhìn **Volume Group** như một ổ cứng mới.

#### **vgcreate lvm-hde /dev/hdf1 /dev/hde5**

```
        Volume group "lvm-hde" successfully created
```

- Câu lệnh trên thực hiện việc tổng hợp 2 **Physical Volumes** **/dev/hdf1** và **/dev/hde5** thành một **Volume Group lvm-hde** dùng lệnh **vgdisplay** để kiểm tra và lấy thông tin về **Volume Group** vừa tạo.

#### **vgdisplay lvm-hde**

```
        --- Volume group ---
        VG Name lvm-hde
        VG Access read/write
        VG Status available/resizable
        VG # 0
        MAX LV 256
        Cur LV 0
        Open LV 0
        MAX LV Size 255.99 GB
        Max PV 256
        Cur PV 2
        Act PV 2
        VG Size 848 MB
        PE Size 4 MB
        Total PE 212
        Alloc PE / Size 0 / 0
        Free PE / Size 212 / 848 MB
        VG UUID W7bgLB-1AFW-wtKi-wZET-jDJF-8VYD-snUaSZ
```

Lưu ý kích thước của Volume Group được tính theo đơn vị PE ( physical extents).  
Như ở đây **Volume Group** này có 212 PE.

- Từ **Volume Group** có thể tạo các **Logical Volume** để phục vụ cho phù hợp với mục đích sử dụng bằng lệnh **lvcreate**

Ý nghĩa của **Logical Volume** gần tương tự như partition:

```
lvcreate -l 212 lvm-hde -n lvm0
```

```

Logical volume "lvm0" created
tạo file system cho Logical Volume lvm0 vừa tạo
mkfs -t ext3 /dev/lvm-hde/lvm0
mke2fs 1.32 (09-Nov-2002)
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
108640 inodes, 217088 blocks
10854 blocks (5.00%) reserved for the super user
First data block=0
7 block groups
32768 blocks per group, 32768 fragments per group
15520 inodes per group
Superblock backups stored on blocks:
32768, 98304, 163840
Writing inode tables: done
Creating journal (4096 blocks): done
Writing superblocks and filesystem accounting information:
done
This filesystem will be automatically checked every 38
mounts or
180 days, whichever comes first. Use tune2fs -c or -i to
override.

```

- Ánh xạ (mount) lại thư mục **/home** và backup lại dữ liệu.
- Sửa lại file **/etc/fstab** để thư mục **/home** được tự động mount mỗi lần boot hệ thống

```
/dev/lvm-hde/lvm0 /home ext3 defaults 1 2
```

### **III. Thực hành**

#### **III.1. Bài 1: Ôn tập về phân vùng**

Thêm 1 ổ cứng 100GiB vào máy ảo (Ổ mới có tên **/dev/sdb**)

Với ổ cứng mới và **/dev/sda**, tạo thêm 10 phân vùng logic với yêu cầu như sau:

- 1 Phân vùng đầu có kích thước 60GiB
- 9 Phân vùng sau có kích thước 300MiB

Khởi động lại máy ảo để active các phân vùng mới tạo

#### **III.2. Bài 2: Thực hành soft raid (1)**

Chạy lệnh sau để tạo thiết bị raid **/dev/md0** theo chuẩn 5, sử dụng 3 phân vùng có trạng thái active, 1 phân vùng có trạng thái spare:

```
mdadm --create /dev/md0 \
--level 5 \
```

```
--raid-devices 3 \
--spare-devices 1 /dev/sd{a,b}{6,7}
```

Chạy hai lệnh sau để kiểm tra trạng thái của thiết bị mới tạo, so sánh hai output của hai lệnh

```
cat /proc/mdstat
mdadm --detail /dev/md0
```

Định dạng **/dev/md0** sau đó ánh xạ thiết bị này vào thư mục **/lab/raid**, kiểm tra kích thước ánh xạ bằng lệnh **df -B miB**. Cuối cùng hủy bỏ ánh xạ này, sau đó hủy bỏ thiết bị trên bằng lệnh

```
mdadm --stop /dev/md0
```

### **III.3. Bài 3: Thực hành soft raid (2)**

Tạo thiết bị **/dev/md3** được ghép bởi các phân vùng vật lý 300MiB theo chuẩn 5 có kích thước sau khi ánh xạ là 1.2GiB trong đó có sử dụng hai phân vùng làm spare. Số phân vùng active phải tự xác định.

Báo lỗi một phân vùng vật lý trong **/dev/md3**, loại bỏ phân vùng lỗi ra khỏi **/dev/md3** và thêm một phân vùng khác vào **/dev/md3**. Trước và sau mỗi bước này cần chạy lệnh kiểm tra tình trạng **/dev/md3** để thấy được sự thay đổi tình trạng của các phân vùng trong **/dev/md3** khi hot swap được thực hiện.

Ánh xạ tự động **/dev/md3** vào **/lab/raid** bằng “đường dẫn tuyệt đối” rồi kiểm tra lại bằng lệnh **mount -a** xem có thực hiện được không? Khởi động lại máy để xem ánh xạ tự động này có thực hiện không? Giải thích kết quả nhận được.

### **III.4. Bài 4: Thực hành LVM (1)**

Ghép hai phân vùng **/dev/sd{a,b}5** lại thành 1 VG có tên vglab có kích thước PE là 16MiB như sau:

```
pvcreate /dev/sd{a,b}5
vgcreate -s 16M vglab /dev/sd{a,b}5
```

Tạo 1 LV có tên **lvlab1** có kích thước 200MiB trong vglab

```
lvcreate -n lvlab1 -L 200M vglab
```

Tạo 1 LV có tên **lvlab2** có kích thước là toàn bộ phần trống còn lại trong vglab sau đó ánh xạ LV này vào **/lab/lvm**

```
lvcreate -n lvlab2 -l 100%FREE vglab
mkfs -t ext4 /dev/vglab/lvlab2
mkdir -p /lab/lvm
mount /dev/mapper/vglab-lvlab2 /lab/lvm
```

Thêm hai phân vùng `/dev/sd{a,b}14` vào `vg_netlab` sau đó tăng kích thước `lv_root` lên toàn bộ phần còn trống của `vg_netlab`

```
pvccreate /dev/sd{a,b}14  
vgextend vg_netlab /dev/sd{a,b}14  
lvresize -r -l +100%FREE /dev/vg_netlab/lv_root
```

### ***III.5. Bài 5: Thực hành LVM (2)***

Giảm kích thước LV **lvlab2** đi 500MiB. Vào chế độ cứu hộ, không chạy lệnh **chroot**, giảm kích thước `lv_root` đi 1GiB. Chú ý phải **umount** tất cả các ánh xạ chứa `/mnt/sysimage` trước khi chạy lệnh `lvresize`