4.Create and configure file systems

* 1. **Create, mount, unmount, and use vfat, ext4, and xfs file systems**

To create a new partition called **lv\_vol** with a size of **100MB**, type:

# **lvcreate --size 100M --name lv\_vol /dev/vg**

## **Ext4 File System**

To create an**ext4** file system (here called **/dev/vg/lv\_vol**), type:

# **mkfs.ext4 /dev/vg/lv\_vol**

mke2fs 1.42.9 (28-Dec-2013)

Filesystem label=

OS type: Linux

Block size=1024 (log=0)

Fragment size=1024 (log=0)

Stride=0 blocks, Stripe width=0 blocks

25688 inodes, 102400 blocks

5120 blocks (5.00%) reserved for the super user

First data block=1

Maximum filesystem blocks=33685504

13 block groups

8192 blocks per group, 8192 fragments per group

1976 inodes per group

Superblock backups stored on blocks:

8193, 24577, 40961, 57345, 73729

Allocating group tables: done

Writing inode tables: done

Creating journal (4096 blocks): done

Writing superblocks and filesystem accounting information: done

To mount this file system, type:

# **mount /dev/vg/lv\_vol /mnt**

To mount it permanently, edit the **/etc/fstab** file and add the following line:

**/dev/mapper/vg-lv\_vol /mnt ext4 defaults 1 2**

Note1: The last number (here **2**) is related to the **fsck**command: ‘**0**‘ means no **fsck** run at boot (very dangerous), ‘**1**‘ **fsck** is run first (root filesystem), ‘**2**‘ **fsck** is run just after the root filesystem. The second to last argument is in relation with the **dump** command (normally set at ‘**1**‘ for real filesystems, ‘**0**‘ for swap and NFS mounted filesystems).  
Note2: A best practice is to execute the **mount -a** command, each time you change something in the **/etc/fstab** file to detect any boot problem before it occurs.

To check an unmounted file system consistency, type:

# **fsck /dev/vg/lv\_vol**

fsck from util-linux 2.23.2

e2fsck 1.42.9 (28-Dec-2013)

/dev/mapper/vg-lv\_vol: clean, 11/25688 files, 8896/102400 block

To get details about a file system, type:

# **dumpe2fs /dev/vg/lv\_vol**

dumpe2fs 1.42.9 (28-Dec-2013)

Filesystem volume name:

Last mounted on:

Filesystem UUID: de13176f-6816-47e8-8742-2d543003d382

Filesystem magic number: 0xEF53

Filesystem revision #: 1 (dynamic)

Filesystem features: has\_journal ext\_attr resize\_inode dir\_index filetype needs\_recovery extent 64bit flex\_bg sparse\_super huge\_file uninit\_bg dir\_nlink extra\_isize

Filesystem flags: signed\_directory\_hash

Default mount options: user\_xattr acl

Filesystem state: clean

Errors behavior: Continue

Filesystem OS type: Linux

Inode count: 25688

Block count: 102400

Reserved block count: 5120

Free blocks: 93504

Free inodes: 25677

First block: 1

Block size: 1024

Fragment size: 1024

Group descriptor size: 64

Reserved GDT blocks: 256

Blocks per group: 8192

Fragments per group: 8192

Inodes per group: 1976

Inode blocks per group: 247

Flex block group size: 16

Filesystem created: Tue Jul 29 17:47:36 2014

Last mount time: Tue Jul 29 17:48:56 2014

Last write time: Tue Jul 29 17:48:56 2014

Mount count: 1

Maximum mount count: -1

Last checked: Tue Jul 29 17:47:36 2014

Check interval: 0 ()

Lifetime writes: 4447 kB

Reserved blocks uid: 0 (user root)

Reserved blocks gid: 0 (group root)

First inode: 11

Inode size: 128

Journal inode: 8

Default directory hash: half\_md4

Directory Hash Seed: 1780d5a3-72df-4cc4-ba19-4a6915c064e3

Journal backup: inode blocks

Journal features: journal\_64bit

Journal size: 4096k

Journal length: 4096

Journal sequence: 0x00000002

Journal start: 1

...

## **Xfs File System**

To create an**xfs** file system (here called **/dev/vg/lv\_vol**), type:

# **mkfs.xfs /dev/vg/lv\_vol**

meta-data=/dev/vg/lv\_vol        isize=256    agcount=4, agsize=6400 blks

         =                       sectsz=512   attr=2, projid32bit=1

         =                       crc=0

data     =                       bsize=4096   blocks=25600, imaxpct=25

         =                       sunit=0      swidth=0 blks

naming   =version 2              bsize=4096   ascii-ci=0 ftype=0

log      =internal log           bsize=4096   blocks=853, version=2

         =                       sectsz=512   sunit=0 blks, lazy-count=1

realtime =none                   extsz=4096   blocks=0, rtextents=0

To mount this file system, type:

# **mount /dev/vg/lv\_vol /mnt**

To mount it permanently, edit the **/etc/fstab** file and add the following line:

**/dev/mapper/vg-lv\_vol /mnt xfs defaults 1 2**

Note: The last number (here **2**) is related to the **fsck**command: ‘**0**‘ means no **fsck** run at boot (very dangerous), ‘**1**‘ **fsck** is run first (root filesystem), ‘**2**‘ **fsck** is run just after the root filesystem. The second to last argument is in relation with the **dump** command (normally set at ‘**1**‘ for real filesystems, ‘**0**‘ for swap and NFS mounted filesystems).  
Note2: A best practice is to execute the **mount -a** command, each time you change something in the **/etc/fstab** file to detect any boot problem before it occurs.

To repair an unmounted file system consistency, type:

# **xfs\_repair /dev/vg/lv\_vol**

Phase 1 - find and verify superblock...

Phase 2 - using internal log

        - zero log...

        - scan filesystem freespace and inode maps...

        - found root inode chunk

Phase 3 - for each AG...

        - scan and clear agi unlinked lists...

        - process known inodes and perform inode discovery...

        - agno = 0

        - agno = 1

        - agno = 2

        - agno = 3

        - process newly discovered inodes...

Phase 4 - check for duplicate blocks...

        - setting up duplicate extent list...

        - check for inodes claiming duplicate blocks...

        - agno = 0

        - agno = 1

        - agno = 2

        - agno = 3

Phase 5 - rebuild AG headers and trees...

        - reset superblock...

Phase 6 - check inode connectivity...

        - resetting contents of realtime bitmap and summary inodes

        - traversing filesystem ...

        - traversal finished ...

        - moving disconnected inodes to lost+found ...

Phase 7 - verify and correct link counts...

done

To get details about a mounted file system, type:

# **xfs\_info /dev/vg/lv\_vol**

meta-data=/dev/mapper/vg-lv\_vol isize=256    agcount=4, agsize=6400 blks

         =                       sectsz=512   attr=2, projid32bit=1

         =                       crc=0

data     =                       bsize=4096   blocks=25600, imaxpct=25

         =                       sunit=0      swidth=0 blks

naming   =version 2              bsize=4096   ascii-ci=0 ftype=0

log      =internal               bsize=4096   blocks=853, version=2

         =                       sectsz=512   sunit=0 blks, lazy-count=1

realtime =none                   extsz=4096   blocks=0, rtextents=0

## **Vfat File System**

To create an**vfat** file system (here called **/dev/vg/lv\_vol**), type:

# **mkfs.vfat /dev/vg/lv\_vol**

mkfs.fat 3.0.20 (12 Jun 2013)

unable to get drive geometry, using default 255/63

To mount this file system, type:

# **mount /dev/vg/lv\_vol /mnt**

To mount it permanently, edit the **/etc/fstab** file and add the following line:

**/dev/mapper/vg-lv\_vol /mnt vfat defaults 1 2**

To repair an unmounted file system consistency, type:

# **fsck.vfat /dev/vg/lv\_vol**

fsck.fat 3.0.20 (12 Jun 2013)

/dev/vg/lv\_vol: 0 files, 0/51091 clusters

## **Useful Tip**

The time to create a filesystem has been dramatically reduced between **Ext3** and **Ext4** (by a factor of 100). However, although the creation time is now very quick, a process in the background continues to work for several minutes (according to the size of the filesystem)!

This process called **ext4lazyinit** creates the remaining index nodes which are used to reference leaf nodes, kind of pointers to data on the filesystem.

Although definitively not required for the **RHCSA** exam, you can create an **ext4** filesystem without this behavior by typing:

# **mkfs.ext4 -E lazy\_itable\_init=0,lazy\_journal\_init=0 /dev/vg/lv\_vol**

Source: [Red Hat](https://people.redhat.com/mskinner/rhug/q1.2016/dm-cache.pdf).

## Additional Resources

You can also watch **Ralph Nyberg**‘s video about [Creating file systems and making them accessible (19min/2015)](https://www.youtube.com/watch?v=iLQJIDC6MAM).  
Beyond the exam objective, you can read a page about [dealing with XFS filesystem corruption](http://serverfault.com/questions/777299/proper-way-to-deal-with-corrupt-xfs-filesystems) or [converting an existing root filesystem to a LVM partition](http://thegeekdiary.com/centos-rhel-converting-an-existing-root-filesystem-to-lvm-partition/).  
**Daniel Aleksandersen** wrote an article discussing [What to do when a Linux system is running out of available inodes](https://www.ctrl.blog/entry/how-to-all-out-of-inodes).

* 1. **Mount and unmount CIFS and NFS network file systems**

## NFS network file system

To mount and unmount **NFS** network file systems, you need to [set up a NFS server](https://www.certdepot.net/rhel7-provide-nfs-network-shares-specific-clients/).

Install the **NFS** client package:

# **yum install -y nfs-utils**

Let’s assume that the **/home/tools** directory is exported by the **nfsserver** server.  
If no working **DNS**, add an entry in the **/etc/hosts** file with the **nfsserver** name and its IP address.

Activate at boot and start the **nfs-idmap** service (**RHEL 7.0** only):

# **systemctl enable nfs-idmap && systemctl start nfs-idmap**

Note: The **nfs-idmap** service is only required by **NFSv4** when setting **ACL** by names or to display user/group names. It doesn’t allow you any **UID/GID mismatches** between clients and server.  
All permission checks are still done with the **UID/GID** used by the **server**.

Activate at boot and start the **nfs-client** target (**RHEL 7.1** and after):

# **systemctl enable nfs-client.target && systemctl start nfs-client.target**

Edit the **/etc/fstab** file and add the following line:

**nfsserver:/home/tools /mnt nfs4 defaults 0 0**

Execute the **/etc/fstab** file configuration:

# **mount -a**

To check the current configuration, type:

# **mount | grep nfsserver**

nfsserver:/home/tools on /mnt type nfs4 (rw,relatime,vers=4.0,rsize=131072,wsize=131072,namlen=255,hard,proto=tcp,port=0,timeo=600,retrans=2,sec=sys,clientaddr=192.168.1.42,local\_lock=none,addr=192.168.1.49)

To unmount the **NFS** mounted directory, remove the previous line from the **/etc/fstab** file and type:

# **umount /mnt**

Note: if you get a message like **“/mnt: device is busy”**, to check that you are not in the mounted directory and no process is using it, type:

# **fuser /mnt**

## CIFS network file system

To mount and unmount **CIFS** network file systems, you need [to set up a CIFS file server](https://www.certdepot.net/rhel7-provide-smb-network-shares/).

Install the **Samba** client packages:

# **yum install -y cifs-utils**

# **yum install -y samba-client**

Let’s assume that the **/shared** directory is exported by the **smbserver** server.  
If no working **DNS**, add an entry in the **/etc/hosts** file with the **smbserver** name and its IP address.

Edit the **/etc/fstab** file and add the following line:

**//smbserver/shared /mnt cifs rw,username=user01,password=pass 0 0**

Execute the **/etc/fstab** file configuration:

# **mount -a**

To check the current configuration, type:

# **mount | grep smbserver**

//smbserver/shared on /mnt type cifs (rw,relatime,vers=1.0,cache=strict,username=user01,domain=MYSERVER,uid=0,noforceuid,gid=0,noforcegid,addr=192.168.1.48,unix,posixpaths,serverino,acl,rsize=1048576,wsize=65536,actimeo=1)

To unmount the **CIFS** mounted directory, remove the previous line from the **/etc/fstab** file and type:

# **umount /mnt**

To learn more about the **Automounter**, go to the [LDAP client configuration tutorial](https://www.certdepot.net/rhel7-configure-system-use-existing-ldap-directory-service-user-group-information/).

* 1. **Extend existing logical volumes**

This tutorial requires **200MB** of available disk space.

Before extending any unencrypted logical volume, let’s create one of **100MB** called **lv\_vol** in the **vg** volume group:

# **lvcreate --size 100M --name lv\_vol vg**

Only unencrypted **Ext4** and **XFS** file systems can be extended, not **vfat**.

Note1: In this tutorial, the **lvextend** and **lvreduce** commands are used to respectively increase and decrease the size of a logical volume. For brevity’s sake, the **lvresize** command is not discussed here, but the same operations can be done in a slightly different way with this command (see comments).

Note2: With **RHEL 7.3**, the **NetworkManager** service got two new directives in its unit file: **ProtectSystem=true** and **ProtectHome=read-only**. These directives forbid some changes made to system (**/usr**, **/boot**, **/root**, **/run/user**) and **/home** directories, mainly the creation of symbolic links to them (see details [here](http://www.spinics.net/lists/centos/msg163489.html)). For most operations, this shouldn’t change anything, but it’s better to know that.

## **Ext4 logical volume extension**

To create an **ext4** file system on the previously created logical volume, type:

# **mkfs.ext4 /dev/vg/lv\_vol**

mke2fs 1.42.9 (28-Dec-2013)

Filesystem label=

OS type: Linux

Block size=1024 (log=0)

Fragment size=1024 (log=0)

Stride=0 blocks, Stripe width=0 blocks

25688 inodes, 102400 blocks

5120 blocks (5.00%) reserved for the super user

First data block=1

Maximum filesystem blocks=33685504

13 block groups

8192 blocks per group, 8192 fragments per group

1976 inodes per group

Superblock backups stored on blocks:

        8193, 24577, 40961, 57345, 73729

Allocating group tables: done

Writing inode tables: done

Creating journal (4096 blocks): done

Writing superblocks and filesystem accounting information: done

Optionally, mount the new file system on **/mnt**:

# **mount /dev/vg/lv\_vol /mnt**

To allocate all the space available in the volume group to an unencrypted **Ext4**-formatted logical volume (here **/dev/vg/lv\_vol**), type:

# **lvextend -l +100%FREE -r /dev/vg/lv\_vol**

Note: The file system is automatically extended (**-r**) without any need to unmount it.

To extend the size of an unencrypted **Ext4**-formatted logical volume (here **/dev/vg/lv\_vol**) by **50MB**, type:

# **lvextend --size +50M -r /dev/vg/lv\_vol**

  Rounding size to boundary between physical extents: 52.00 MiB

Extending logical volume lv\_vol to 152.00 MiB

Logical volume lv\_vol successfully resized

resize2fs 1.42.9 (28-Dec-2013)

Filesystem at /dev/mapper/vg-lv\_vol is mounted on /mnt; on-line resizing required

old\_desc\_blocks = 1, new\_desc\_blocks = 2

The filesystem on /dev/mapper/vg-lv\_vol is now 155648 blocks long.

**Although there is no reason to do so**, you can still do it in two steps:

# **lvextend --size +50M /dev/vg/lv\_vol**

  Rounding size to boundary between physical extents: 52.00 MiB

  Extending logical volume lv\_vol to 152.00 MiB

  Logical volume lv\_vol successfully resized

# **resize2fs /dev/vg/lv\_vol**

resize2fs 1.42.9 (28-Dec-2013)

Filesystem at /dev/vg/lv\_vol is mounted on /mnt; on-line resizing required

old\_desc\_blocks = 1, new\_desc\_blocks = 2

The filesystem on /dev/vg/lv\_vol is now 155648 blocks long.

## **Ext4 logical volume reduction**

**Conversely**, to reduce the size of a logical volume (here by **50MB**), you have to follow these steps:  
Unmount the file system (here **/dev/vg/lv\_vol**):

# **umount /dev/vg/lv\_vol**

Reduce the size of the logical volume (here **/dev/vg/lv\_vol**) and the associated file system at the same time (**-r**):

# **lvreduce --size -50M -r /dev/vg/lv\_vol**

  Rounding size to boundary between physical extents: 48.00 MiB

fsck from util-linux 2.23.2

/dev/mapper/vg-lv\_vol: 11/37544 files (9.1% non-contiguous), 10390/155648 blocks

resize2fs 1.42.9 (28-Dec-2013)

Resizing the filesystem on /dev/mapper/vg-lv\_vol to 106496 (1k) blocks.

The filesystem on /dev/mapper/vg-lv\_vol is now 106496 blocks long.

  Reducing logical volume lv\_vol to 104.00 MiB

  Logical volume lv\_vol successfully resized

Mount the file system (here **/dev/vg/lv\_vol**):

# **mount /dev/vg/lv\_vol /mnt**

## **XFS logical volume extension**

To create a **XFS** file system on the previously created logical volume called **lv\_vol** (see **prerequisites**), type:

# **mkfs.xfs /dev/vg/lv\_vol**

meta-data=/dev/vg/lv\_vol isize=256 agcount=4, agsize=6400 blks

= sectsz=512 attr=2, projid32bit=1

= crc=0

data = bsize=4096 blocks=25600, imaxpct=25

= sunit=0 swidth=0 blks

naming =version 2 bsize=4096 ascii-ci=0 ftype=0

log =internal log bsize=4096 blocks=853, version=2

= sectsz=512 sunit=0 blks, lazy-count=1

realtime =none extsz=4096 blocks=0, rtextents=0

Mount the new file system on **/mnt**:

# **mount /dev/vg/lv\_vol /mnt**

To extend the size of an unencrypted **XFS**-formatted logical volume (here **/dev/vg/lv\_vol**) by **50MB**, type:

# **lvextend --size +50M -r /dev/vg/lv\_vol**

Rounding size to boundary between physical extents: 52.00 MiB

Extending logical volume lv\_vol to 152.00 MiB

Logical volume lv\_vol successfully resized

meta-data=/dev/mapper/vg-lv\_vol isize=256 agcount=4, agsize=6400 blks

= sectsz=512 attr=2, projid32bit=1

= crc=0

data = bsize=4096 blocks=25600, imaxpct=25

= sunit=0 swidth=0 blks

naming =version 2 bsize=4096 ascii-ci=0 ftype=0

log =internal bsize=4096 blocks=853, version=2

= sectsz=512 sunit=0 blks, lazy-count=1

realtime =none extsz=4096 blocks=0, rtextents=0

data blocks changed from 25600 to 38912

Note1: The file system is automatically extended (**-r**) without any need to unmount it.  
Note2: **You can’t reduce a XFS file system even though you unmount it. You have to back it up (with tar or another tool), drop it and recreate it.**

**Although there is no reason to do so**, you can still do it in two steps:

# **lvextend --size +50M /dev/vg/lv\_vol**

  Rounding size to boundary between physical extents: 52.00 MiB

  Extending logical volume lv\_vol to 152.00 MiB

  Logical volume lv\_vol successfully resized

# **xfs\_growfs /mnt**

meta-data=/dev/mapper/vg-lv\_vol isize=256    agcount=4, agsize=6400 blks

         =                       sectsz=512   attr=2, projid32bit=1

         =                       crc=0

data     =                       bsize=4096   blocks=25600, imaxpct=25

         =                       sunit=0      swidth=0 blks

naming   =version 2              bsize=4096   ascii-ci=0 ftype=0

log      =internal               bsize=4096   blocks=853, version=2

         =                       sectsz=512   sunit=0 blks, lazy-count=1

realtime =none                   extsz=4096   blocks=0, rtextents=0

data blocks changed from 25600 to 38912

Note: The **xfs\_growfs** command accepts only a mounted **XFS** filesystem as parameter.

## **Additional Resources**

Although outside of the **RHCSA** objectives, [TheGeekDiary](http://thegeekdiary.com/) website offers an interesting tutorial about [shrinking the LVM root file system](http://thegeekdiary.com/centos-rhel-how-to-shrink-lvm-root-file-system/).

* 1. **Create and configure set-GID directories for collaboration**

Let’s assume two users belonging to the **team** group, **user01** and **user02**, who want to share a directory called **shared**.

Create the **team** group:

# groupadd -g 50000 team

Create the **shared** directory:

# mkdir /home/shared

Change the ownership of the directory:

# chown nobody:team /home/shared

Assign the set group ID bit (**SGID**) to the directory:

# chmod g+s /home/shared

Allow the members of the **team** group to write into the **shared** directory:

# chmod g+w /home/shared

Remove the permissions for all other users:

# chmod o-rwx /home/shared

Note: The three last commands can be resumed in only one to choose among these:

# chmod g+ws,o-rwx /home/shared

# chmod 2770 /home/shared

Create the two users and assign them the **team** group in addition to their own group:

# useradd -G team user01

# useradd -G team user02

Note: This can be done in two steps:

# useradd user0X; usermod -aG team user0X

Check the configuration:

# su - user01

$ cd /home/shared

$ touch nothing

$ ls -l

total 0

-rw-rw-r--. 1 user01 team 0 Nov 12 09:45 nothing

Finally, if you want the **team** group members to be able to see each other’s files but not to delete them, type:

# chmod +t /home/shared

* 1. **Create and manage Access Control Lists (ACLs)**

When basic file permissions are not enough, you can use ACL.

ACL stands for Access Control Lists.

## **Prerequisites**

However, before doing this, you have to check if the partition permits ACLs.  
To check that ACLs work, type:

# mount

/dev/mapper/vg\_root-lv\_root on / type ext4 **(rw)**

In this case, you have to edit the **/etc/fstab** file, add **“,acl”** after the **defaults** or **rw** option and, then, remount the partition:

# mount -o remount /

## **ACL Configuration**

To allow **read/write** access to the user **bob** on the file called **f** (**-m** for modify, **u** for user, **rw-** for read/write access), type:

# setfacl **-m** **u**:**bob**:**rw-** **f**

To request access control list status on the same file **f**, type:

# getfacl **f**

# file: **f**

# owner: root

# group: root

user::rw-

user:**bob**:**rw-**

group::r--

mask::rw-

other::r--

To remove permissions allowed to the user **bob** (**-x** for remove, **u** for user), type:

# setfacl **-x** **u**:**bob** **f**

To remove all the ACLs on a file called **f** (**-b** for remove-all), type:

# setfacl **-b** **f**

To allow **read/execute** permissions to the group called **team** on a directory **dir** and all the files inside (**-R** for recursive, **-m** for modify, **g** for group, **r-x** for read/execute access), type:

# setfacl **-R** **-m** **g**:**team**:**r-x** **dir**

To get the result, type:

# getfacl **dir**

# file: **dir**

# owner: root

# group: root

user::rwx

group::r-x

group:**team**:**r-x**

mask::r-x

other::r-x

## **Addition Resources**

You can watch **Ralph Nyberg**‘s video about  [Configuring ACLs (18min/2015)](https://www.youtube.com/watch?v=jpNnCn8EsHU).  
Also, the **setfacl** man page is a good source of information.

* 1. **Diagnose and correct file permission problems**

## **cannot access a file/folder**

Check: with ls -al to see if you have permissions. If you see + at the end of permissions for the file use getfacl   
Correct: with chmod or setfacl

## **Can not delete a file**

Check: sticky bit / not root user  
Correct: with chmod

Check: use lsattr   
Correct: with chattr

## **Can not navigate directories**

Check: to see if directory has executable permission for your user or group or for world  
Correct: chmod +X /dirName – to allow everyone to browse directory – can be used recursively

## **Can not access file/folder after file move**

cp – does not preserve acls – [See previous Objective on copying ACL between files](https://greplog.com/rhcsa-objective04-05-create-and-manage-access-control-lists-acls/)  
mv – does preserve acls

## **Application cannot write from directory**

Check to see if it works as expected after temporary setting selinux to permissive

If you have any more you would like to add please leave a comment and I will update article