

RHEL: XFS basic operations

Article Number: 144 | Rating: 1/5 from 1 votes | Last Updated: Sat, Jun 2, 2018 9:14 AM

RHEL: XFS basic operations

```
# Tested on RHEL 6 & 7

# XFS filesystem supports up to 16 EiB filesystems and up to 8 EiB and directory
structures
# holding tens of millions entries.
# On the other hand it is a single node filesystem. It comes integrated in RHEL 7,
in order
# to use it on RHEL 6 we need a subscription to the Scalable File System Add-ON.

# Although XFS file system supports up to 16 EiB, Red Hat only supports filesystems
up to
# 100 TiB

# Other limitations of XFS is that it's less suited for single threads creating and
deleting
# a large number of small files; in addition it uses about twice CPU resources that
ext4 so
# under those circumstances it is better to use ext4 filesystems.

# In general XFS is best suited for large systems with fast storage.

# As already indicated, to use XFS filesystems we need the "Scalable Filesystems"
group /

# 'xfsprogs' package

# Create an xfs filesystem
# -----
-----

mkfs.xfs /dev/sdd2
```

```

# If we know storage parameters, we can specify chunk size ('su', stripe unit)
and/or
# stripe width ('sw') in order to improve xfs fs performance. For example, to
create an xfs
# file system with a stripe-unit size of 32 KB and 4 units per stripe, we will
specify:

mkfs.xfs -d su=32k,sw=4 /dev/sdd2
meta-data=/dev/sdd2          isize=256    agcount=4, agsize=655352 blks
=                               sectsz=512   attr=2
data      =                   bsize=4096  blocks=2621408, imaxpct=25
=                               sunit=8     swidth=32 blks
naming    =version 2          bsize=4096  ascii-ci=0
log       =internal log      bsize=4096  blocks=2560, version=2
=                               sectsz=512   sunit=8 blks, lazy-count=1
realtime  =none              extsz=4096  blocks=0, rtextents=0

# External journal
# -----
-----

# By default xfs store journal internally. As synchronous metadata writes to the
journal
# must complete successfully before any associated data such a layout can lead to
disk
# contention. To improve performance we can consider placing journal on a separate
physical
# device. To create an external journal, use the "-l logdev=device,size=size"
option during

# xfs creation. If we omit the size parameter, mkfs.xfs selects a size based on the
size
# of the file system.

mkfs.xfs -l logdev=/dev/sde2 /dev/sdd2

# Mount an xfs F.S. without/with external journal
# -----

```

```

-----

# by editing /etc/fstab

/dev/datavg/lv_xfsdata    /myxfs    xfs    defaults    0 0

# or

/dev/datavg/lv_xfsdata    /myxfs    xfs    logdev=/dev/datavg/lvxfsjournal    0
0

# via 'mount' command

mount /dev/datavg/lv_xfsdata /myxfs

# or

mount -o logdev=/dev/datavg/lv_xfsjournal /dev/datavg/lv_xfsdata /myxfs

# Grow an xfs filesystem
# -----
-----

# 1.- We CANNOT grow an unmounted xfs
# 2.- An xfs filesystem CANNOT be shrunk

# We can increase the size of a XFS file system if there is enough space on
the underlying
# device. If necessary, increase the size of the logical volume (or disk partition
or LUN
# and make changes visible to the system).

df -h /myxfs
Filesystem                Size      Used Avail Use% Mounted on
/dev/sdf1                  2.0G      33M   2.0G   2% /myxfs

# To specify the final size of the xfs, we use the '-D' option. The size is
expressed in

```

```
# filesystem blocks. With a default block size of 4096 bytes, to grow our xfs up to 3 GiB:
```

```
# 3 GiB = 3221225472 byte = 786432 blocks
```

```
xfs_growfs -D 786432 /myxfs
meta-data=/dev/sdf1          isize=256    agcount=4, agsize=131530 blks
                =                sectsz=512   attr=2
data       =                bsize=4096   blocks=526120, imaxpct=25
                =                sunit=0      swidth=0 blks
naming     =version 2        bsize=4096   ascii-ci=0
log        =internal        bsize=4096   blocks=2560, version=2
                =                sectsz=512   sunit=0 blks, lazy-count=1
realtime   =none            extsz=4096   blocks=0, rtextents=0
data blocks changed from 526120 to 786432
```

```
df -h /myxfs
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdf1       3.0G  33M  3.0G   2% /myxfs
```

```
# To grow the xfs to the largest size possible:
```

```
xfs_growfs -d /myxfs
```

```
df -h /myxfs
Filesystem      Size  Used Avail Use% Mounted on
/dev/sdf1       10G   33M   10G   1% /myxfs
```

```
# Reduce an xfs filesystem
```

```
# -----  
-----
```

```
# Unfortunately, it is not possible to reduce an xfs filesystem
```

```
# Repair an xfs when file system is not cleanly unmounted, ...
# -----
-----

xfs_repair [ -l <logdev> ] /dev/sdd2

# An xfs file system with a dirty log cannot be repaired. To clear out the log you
can
# mount, then unmount. If this fails try '-L' option to xfs_repair to force clear
the
# log (as a last resort as it may result in a data corruption)

# Display/modify label and UUID of an xfs
# -----
-----

# Display existing label:

xfs_admin -l /dev/sde
label = ""

# Set a new label (filesystem has to be unmounted):

xfs_admin -L "NewLabel" /dev/sde
writing all SBs
new label = "NewLabel"

# Display existing UUID:

xfs_admin -u /dev/sde
UUID = 51b11165-c59d-44a6-8f4e-3616aaf79a4d

# Generate a new UUID (filesystem has to be unmounted):

xfs_admin -U generate /dev/sde
Clearing log and setting UUID
writing all SBs
new UUID = a05ff818-dc74-4d59-aff8-92b360c2a2ed
```

```
# Clear the UUID (filesystem has to be unmounted):

xfs_admin -U nil /dev/sde
Clearing log and setting UUID
writing all SBs
new UUID = 00000000-0000-0000-0000-000000000000

# Let's see what happens if we try to mount an xfs with a nil UUID:

mount /dev/sde /xfs01
mount: wrong fs type, bad option, bad superblock on /dev/sde,
missing codepage or helper program, or other error
In some cases useful info is found in syslog - try
dmesg | tail or so

dmesg | tail -1
XFS (sde): Filesystem has nil UUID - can't mount

# Defragmenting an xfs
# -----
-----

xfs_fsr /dev/sdd2

# To defragment a single file run xfs_fsr <path_to_file>

# If no option is given to 'xfs_fsr' command it will defragment all the xfs on the
server.
# Since this can potentially be a very long running operation the 'xfs_fsr' tool
will
# stop after a number of seconds specified with the '-t' option (by default 7200
seconds,
# this is 2 hours)
```

Posted - Sat, Jun 2, 2018 9:14 AM. This article has been viewed 17523 times.

Online URL: <http://kb.ictbanking.net/article.php?id=144>