Without any further switches, this file system uses RAID0 for data (non-redundant) and RAID1 for metadata (redundant). When data is lost for some reason (e.g. failed sectors on your hard drive), btrfs can use metadata for trying to rebuild that data.

```
root@server1:~# mkfs.btrfs /dev/sdb /dev/sdc /dev/sdd

WARNING! - Btrfs Btrfs v0.19 IS EXPERIMENTAL
WARNING! - see http://btrfs.wiki.kernel.org before using

adding device /dev/sdc id 2
adding device /dev/sdd id 3
fs created label (null) on /dev/sdb
        nodesize 4096 leafsize 4096 sectorsize 4096 size 15.00GB
Btrfs Btrfs v0.19
root@server1:~#
```

If you want to use btrfs with just one hard drive and don't want metadata to be redundant (attention: this is dangerous - if your metadata is lost, your data is lost as well), you'd use the `-m single` switch (`-m` refers to metadata, `-d` to data):

```
mkfs.btrfs -m single /dev/sdb
```

If you want to do the same with multiple hard drives (i.e., non-redundant metadata), you'd use `-m raid0` instead of `-m single`:

```
mkfs.btrfs -m raid0 /dev/sdb /dev/sdc /dev/sdd
```

If you want data to be redundant and metadata to be non-redundant, you'd use the following command:

```
mkfs.btrfs -m raid0 -d raid1 /dev/sdb /dev/sdc /dev/sdd
```

If you want both data and metadata to be redundant, you'd use this command (RAID1 is the default for metadata, that's why we don't have to specify it here):

```
mkfs.btrfs -d raid1 /dev/sdb /dev/sdc /dev/sdd
```

It is also possible to use RAID10 (*-m raid10* or *-d raid10*), but then you need at least four hard drives. For RAID1, you need at least two hard drives, but it is not important that both drives have exactly the same size (which is another great thing about btrfs).

To get details about your filesystem, you can use...

```
btrfs filesystem show /dev/sdb
```

... which is equivalent to...

```
btrfs filesystem show /dev/sdc
```

... and...

```
btrfs filesystem show /dev/sdd
```

... because you can use any hard drive which is part of the btrfs file system.

```
root@server1:~# btrfs filesystem show /dev/sdb
failed to read /dev/sr0
Label: none  uuid: 21f33aaa-b2b3-464b-8cf1-0f8cc3689529
        Total devices 3 FS bytes used 28.00KB
        devid    3 size 5.00GB used 1.01GB path /dev/sdd
        devid    2 size 5.00GB used 1.01GB path /dev/sdc
        devid    1 size 5.00GB used 2.02GB path /dev/sdb

Btrfs Btrfs v0.19
root@server1:~#
```

To get a list of all btrfs file systems, just leave out the device:

```
btrfs filesystem show
```

```
root@server1:~# btrfs filesystem show
failed to read /dev/sr0
Label: none  uuid: 21f33aaa-b2b3-464b-8cf1-0f8cc3689529
        Total devices 3 FS bytes used 28.00KB
        devid    3 size 5.00GB used 1.01GB path /dev/sdd
        devid    2 size 5.00GB used 1.01GB path /dev/sdc
        devid    1 size 5.00GB used 2.02GB path /dev/sdb

Btrfs Btrfs v0.19
root@server1:~#
```

4 Mounting btrfs File Systems

Our btrfs file system can now be mounted like this:

```
mount /dev/sdb /mnt
```

Again, this is equivalent to...

```
mount /dev/sdc /mnt
```

... and:

```
mount /dev/sdd /mnt
```

In your */etc/fstab*, this would look as follows (if you want to have the file system mounted automatically at boot time):

```
vi /etc/fstab
```

```
[...]
```

```
/dev/sdb /mnt              btrfs   defaults 0       1
[...]
```

Run...

```
df -h
```

... to see your new file system:

```
root@server1:~# df -h
Filesystem                Size  Used Avail Use% Mounted on
udev                      489M  4.0K  489M   1% /dev
tmpfs                     200M  308K  199M   1% /run
none                      5.0M     0  5.0M   0% /run/lock
none                      498M     0  498M   0% /run/shm
none                      100M     0  100M   0% /run/user
/dev/mapper/server1-root   27G  1.1G   25G   5% /
/dev/sda1                 228M   29M  188M  14% /boot
/dev/sdb                   15G   56K   10G   1% /mnt
root@server1:~#
```

The command...

```
btrfs filesystem df /mnt
```

... gives you some more details about your data and metadata (e.g. RAID levels):

```
root@server1:~# btrfs filesystem df /mnt
Data, RAID1: total=1.00GB, used=0.00
Data: total=8.00MB, used=0.00
System, RAID1: total=8.00MB, used=4.00KB
System: total=4.00MB, used=0.00
Metadata, RAID1: total=1.00GB, used=24.00KB
Metadata: total=8.00MB, used=0.00
root@server1:~#
```

5 Using Compression With btrfs

btrfs file systems can make use of zlib (default) and lzo compression which means that compressible files will be stored in compressed form on the hard drive which saves space. zlib has a higher compression ratio while lzo is faster and takes less cpu load. Using compression, especially lzo compression, can improve the throughput preformance. Please note that btrfs will not compress files that have already been compressed ar application level (such as videos, music, images, etc.).

You can mount a btrfs file system with lzo compression as follows:

```
mount -o compress=lzo /dev/sdb /mnt
```

For zlib compression, you'd either use...

```
mount -o compress=zlib /dev/sdb /mnt
```

... or...

```
mount -o compress /dev/sdb /mnt
```

... since zlib is the default compression algorithm.

In */etc/fstab*, this would look as follows:

```
vi /etc/fstab
```

```
[...]
/dev/sdb /mnt              btrfs   defaults,compress=lzo 0       1
[...]
```

6 Rescuing A Dead btrfs File System

If you have a dead btrfs file system, you can try to mount it with the *recovery* mount option which will try to seek for a usable copy of the tree root:

```
mount -o recovery /dev/sdb /mnt
```

7 Resizing btrfs File Systems Online

btrfs file systems can be resized online, i.e., there's no need to unmount the partition or to reboot into a rescue system.

To decrease our /mnt volume by 2GB, we run:

```
btrfs filesystem resize -2g /mnt
```

(Instead of *g* for GB, you cam also use *m* for MB, e.g.

```
btrfs filesystem resize -500m /mnt
```

)

```
root@server1:~# btrfs filesystem resize -2g /mnt
Resize '/mnt' of '-2g'
root@server1:~#
```

Let's take a look at our */mnt* partition...

```
df -h
```

... and we should see that it has a size of 13GB instead of 15GB:

```
root@server1:~# df -h
Filesystem                Size  Used Avail Use% Mounted on
udev                      489M  4.0K  489M   1% /dev
tmpfs                     200M  308K  199M   1% /run
none                      5.0M     0  5.0M   0% /run/lock
none                      498M     0  498M   0% /run/shm
none                      100M     0  100M   0% /run/user
/dev/mapper/server1-root   27G  1.1G   25G   5% /
/dev/sda1                 228M   29M  188M  14% /boot
/dev/sdb                   13G  312K   10G   1% /mnt
root@server1:~#
```

To increase the `/mnt` partition by 1GB, run:

```
btrfs filesystem resize +1g /mnt
```

```
df -h
```

```
root@server1:~# df -h
Filesystem                Size  Used Avail Use% Mounted on
udev                      489M  4.0K  489M   1% /dev
tmpfs                     200M  308K  199M   1% /run
none                      5.0M     0  5.0M   0% /run/lock
none                      498M     0  498M   0% /run/shm
none                      100M     0  100M   0% /run/user
/dev/mapper/server1-root   27G  1.1G   25G   5% /
/dev/sda1                 228M   29M  188M  14% /boot
/dev/sdb                   14G  312K   10G   1% /mnt
root@server1:~#
```

To increase the partition to the max. available space, run:

```
btrfs filesystem resize max /mnt
```

```
df -h
```

```
root@server1:~# df -h
Filesystem                 Size  Used Avail Use% Mounted on
udev                       489M  4.0K  489M   1% /dev
tmpfs                      200M  308K  199M   1% /run
none                       5.0M     0  5.0M   0% /run/lock
none                       498M     0  498M   0% /run/shm
none                       100M     0  100M   0% /run/user
/dev/mapper/server1-root    27G  1.1G   25G   5% /
/dev/sda1                  228M   29M  188M  14% /boot
/dev/sdb                    15G  312K   10G   1% /mnt
root@server1:~#
```

8 Adding/Deleting Hard Drives To/From A btrfs File System

Now we want to add */dev/sde* to our btrfs file system. While the file system is mounted to */mnt*, we simply run:

```
btrfs device add /dev/sde /mnt
```

Let's take a look at the file system afterwards:

```
btrfs filesystem show /dev/sdb
```

```
root@server1:~# btrfs filesystem show /dev/sdb
failed to read /dev/sr0
Label: none  uuid: 21f33aaa-b2b3-464b-8cf1-0f8cc3689529
        Total devices 4 FS bytes used 156.00KB
        devid    4 size 5.00GB used 0.00 path /dev/sde
        devid    3 size 5.00GB used 1.01GB path /dev/sdd
```

```
        devid    2 size 5.00GB used 1.01GB path /dev/sdc
        devid    1 size 5.00GB used 2.02GB path /dev/sdb

Btrfs Btrfs v0.19
root@server1:~#
```

As you see, */dev/sde* has been added, but no space is being used on that device. If you are using a RAID level other than 0, you should now do a filesystem balance so that data and metadata get spread over all four devices:

```
btrfs filesystem balance /mnt
```

(Another syntax for the same command would be:

```
btrfs balance start /mnt
```

)

```
root@server1:~# btrfs filesystem balance /mnt
Done, had to relocate 5 out of 5 chunks
root@server1:~#
```

Let's take a look at our file system again:

```
btrfs filesystem show /dev/sdb
```

```
root@server1:~# btrfs filesystem show /dev/sdb
failed to read /dev/sr0
Label: none  uuid: 21f33aaa-b2b3-464b-8cf1-0f8cc3689529
        Total devices 4 FS bytes used 28.00KB
        devid    4 size 5.00GB used 512.00MB path /dev/sde
        devid    3 size 5.00GB used 32.00MB path /dev/sdd
        devid    2 size 5.00GB used 512.00MB path /dev/sdc
        devid    1 size 5.00GB used 36.00MB path /dev/sdb
```

```
Btrfs Btrfs v0.19
root@server1:~#
```

As you can see, data/metadata has been moved to */dev/sde*.

To delete an **intact** hard drive, e.g. */dev/sdc*, from the btrfs file system online, you can simply run:

```
btrfs device delete /dev/sdc /mnt
```

(This automatically does a rebalance of data/metadata, if necessary.)

While...

```
btrfs filesystem show /dev/sdb
```

... still lists */dev/sdc*, the output of...

```
df -h
```

... shows the reduced size of the file system.

To remove a **failed** hard drive, unmount the file system first:

```
umount /mnt
```

Mount it in degraded mode:

```
mount -o degraded /dev/sdb /mnt
```

Remove the failed hard drive. If you use a RAID level that requires a certain number of hard drives (e.g. two for RAID1 and four for RAID10), you might have to add an intact replacement drive because you cannot go below the minimum number of required drives.

If you have to add a replacement drive (e.g. `/dev/sdf`), do it as follows:

```
btrfs device add /dev/sdf /mnt
```

Only if you are sure you have enough intact drives do you run the following command to complete the replacement:

```
btrfs device delete missing /mnt
```

Next >>

Sub pages

A Beginner's Guide To btrfs
A Beginner's Guide To btrfs - Page 2

Suggested articles