https://www.digitalocean.com/community/tutorials/how-to-manage-raid-arrays-with-mdadm-on-ubuntu-16-04#adding-a-spare-device-to-an-array

How To Manage RAID Arrays with mdadm on Ubuntu 16.04

PostedAugust 16, 2016 28.8kviews [STORAGE](https://www.digitalocean.com/community/tags/storage?type=tutorials) [BLOCK STORAGE](https://www.digitalocean.com/community/tags/block-storage?type=tutorials) [UBUNTU](https://www.digitalocean.com/community/tags/ubuntu?type=tutorials) [UBUNTU 16.04](https://www.digitalocean.com/community/tags/ubuntu-16-04?type=tutorials)

**Introduction**

RAID arrays provide increased performance and redundancy by combining individual disks into virtual storage devices in specific configurations. In Linux, the mdadm utility makes it easy to create and manage software RAID arrays.

In a previous guide, we covered [how to create RAID arrays with mdadm on Ubuntu 16.04](https://www.digitalocean.com/community/tutorials/how-to-create-raid-arrays-with-mdadm-on-ubuntu-16-04). In this guide, we will demonstrate how to manage RAID arrays on an Ubuntu 16.04 server. Managing RAID arrays is quite straight forward in most cases.

Prerequisites

To complete this guide, you will need access to a non-root sudo user. You can follow our [Ubuntu 16.04 initial server setup guide](https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-16-04) to set up an appropriate user.

As mentioned above, this guide will cover RAID array management. Follow our guide on [how to create RAID arrays with mdadm on Ubuntu 16.04](https://www.digitalocean.com/community/tutorials/how-to-create-raid-arrays-with-mdadm-on-ubuntu-16-04) to create one or more arrays before starting on this guide. This guide will assume that you have one or more arrays to operate on.

Querying for Information about RAID Devices

One of the most essential requirements for proper management is the ability to find information about the structure, component devices, and current state of the array.

To get detailed information about a RAID device, pass the RAID device with the -D or --detail option to mdadm:

* sudo mdadm -D /dev/md0

The important information about the array will be displayed:

Output

/dev/md0:

Version : 1.2

Creation Time : Mon Aug 8 21:19:06 2016

Raid Level : raid10

Array Size : 209584128 (199.88 GiB 214.61 GB)

Used Dev Size : 104792064 (99.94 GiB 107.31 GB)

Raid Devices : 4

Total Devices : 4

Persistence : Superblock is persistent

Update Time : Mon Aug 8 21:36:36 2016

State : active

Active Devices : 4

Working Devices : 4

Failed Devices : 0

Spare Devices : 0

Layout : near=2

Chunk Size : 512K

Name : mdadmwrite:0 (local to host mdadmwrite)

UUID : 0dc2e687:1dfe70ac:d440b2ac:5828d61d

Events : 18

Number Major Minor RaidDevice State

0 8 0 0 active sync set-A /dev/sda

1 8 16 1 active sync set-B /dev/sdb

2 8 32 2 active sync set-A /dev/sdc

3 8 48 3 active sync set-B /dev/sdd

From this view you can see the RAID level, the array size, the health of the individual pieces, the UUID of the array, and the component devices and their roles. The information provided in this view is all fairly well labeled.

To get the shortened details for an array, appropriate for adding to the /dev/mdadm/mdadm.conf file, you can pass in the --brief or -b flags with the detail view:

* sudo mdadm -Db /dev/md0

Output

ARRAY /dev/md0 metadata=1.2 name=mdadmwrite:0 UUID=0dc2e687:1dfe70ac:d440b2ac:5828d61d

To get a quick human-readable summary of a RAID device, use the -Q option to query it:

* sudo mdadm -Q /dev/md0

Output

/dev/md0: 199.88GiB raid10 4 devices, 0 spares. Use mdadm --detail for more detail.

This can be used to find the key info about a RAID device at a glance.

Getting Information about Component Devices

You can also use mdadm to query individual component devices.

The -Q option, when used with a component device, will tell you the array it is a part of and its role:

* sudo mdadm -Q /dev/sdc

Output

/dev/sdc: is not an md array

/dev/sdc: device 2 in 4 device active raid10 /dev/md0. Use mdadm --examine for more detail.

You can get more detailed information by using the -E or --examine options:

* sudo mdadm -E /dev/sdc

Output

/dev/sdc:

Magic : a92b4efc

Version : 1.2

Feature Map : 0x0

Array UUID : 0dc2e687:1dfe70ac:d440b2ac:5828d61d

Name : mdadmwrite:0 (local to host mdadmwrite)

Creation Time : Mon Aug 8 21:19:06 2016

Raid Level : raid10

Raid Devices : 4

Avail Dev Size : 209584128 (99.94 GiB 107.31 GB)

Array Size : 209584128 (199.88 GiB 214.61 GB)

Data Offset : 131072 sectors

Super Offset : 8 sectors

Unused Space : before=130984 sectors, after=0 sectors

State : active

Device UUID : b0676ef0:73046e93:9d9c7bde:c80352bb

Update Time : Mon Aug 8 21:36:36 2016

Bad Block Log : 512 entries available at offset 72 sectors

Checksum : 8be1be96 - correct

Events : 18

Layout : near=2

Chunk Size : 512K

Device Role : Active device 2

Array State : AAAA ('A' == active, '.' == missing, 'R' == replacing)

This information is similar to that displayed when using the -D option with the array device, but focused on the component device's relationship to the array.

Reading the /proc/mdstat Information

To get detailed information about each of the assembled arrays on your server, check the /proc/mdstatfile. This is often the best way to find the current status of the active arrays on your system:

* cat /proc/mdstat

Output

Personalities : [raid6] [raid5] [raid4] [linear] [multipath] [raid0] [raid1] [raid10]

md0 : active raid10 sdd[3] sdc[2] sdb[1] sda[0]

209584128 blocks super 1.2 512K chunks 2 near-copies [4/4] [UUUU]

unused devices: <none>

The output here is quite dense, providing a lot of information in a small amount of space.

/proc/mdstat

Personalities : [raid6] [raid5] [raid4] [linear] [multipath] [raid0] [raid1] [raid10]

. . .

The **Personalities** line describes the different RAID levels and configurations that the kernel currently supports.

The line beginning with **md0** describes the beginning of a RAID device description. The indented line(s) that follow are also describe this device.

/proc/mdstat

. . .

md0 : active raid10 sdd[3] sdc[2] sdb[1] sda[0]

. . .

The first line state that the array is active (not faulty) and configured as RAID 10. Afterwards, the component devices that were used to build the array are listed. The numbers in the brackets describe the current "role" of the device in the array (this affects which copies of data the device is given).

/proc/mdstat

. . .

209584128 blocks super 1.2 512K chunks 2 near-copies [4/4] [UUUU]

. . .

The second line displayed in this example gives the number of blocks the virtual devices provides, the metadata version (1.2 in this example), and the chunk size of the array. Since this is a RAID 10 array, it also includes information about the layout of the array (this example has been configured to store two copies of each chunk of data in the "near" layout).

The last items in square brackets both represent currently available devices out of a healthy set. The first number in the numeric brackets indicates the size of a healthy array while the second number represents the currently available number of devices. The other brackets are a visual indication of the array health, with "U" representing healthy devices and "\_" representing faulty devices.

If your array is currently assembling or recovering, you might have another line that shows the progress. It would look something like this:

/proc/mdstat

. . .

[>....................] resync = 0.9% (2032768/209584128) finish=15.3min speed=225863K/sec

. . .

This this describes the operation currently being applied and the current progress in a number of different ways. It also provides the current speed and an estimated time until completion.

After you have a good idea of what arrays are currently running on your system, there are a number of actions you can take.

Stopping an Array

To stop an array, the first step is to unmount it.

Step outside of the mounted directory and unmount it by typing:

* cd ~
* sudo umount /mnt/md0

You can stop all active arrays by typing:

* sudo mdadm --stop --scan

If you want to stop a specific array, pass it to the mdadm --stop command:

* sudo mdadm --stop /dev/md0

This will stop the array. You will have to reassemble the array to access it again.

Starting an Array

To start all arrays defined in the configuration files or /proc/mdstat, type:

* sudo mdadm --assemble --scan

To start a specific array, you can pass it in as an argument to mdadm --assemble:

* sudo mdadm --assemble /dev/md0

This works if the array is defined in the configuration file.

If the correct definition for the array is missing from the configuration file, the array can still be started by passing in the component devices:

* sudo mdadm --assemble /dev/md0 /dev/sda /dev/sdb /dev/sdc /dev/sdd

Once the array is assembled, it can be mounted as usual:

* sudo mount /dev/md0 /mnt/md0

The array should now be accessible at the mount point.

Adding a Spare Device to an Array

Spare devices can be added to any arrays that offer redundancy (such as RAID 1, 5, 6, or 10). The spare will not be actively used by the array unless an active device fails. When this happens, the array will resync the data to the spare drive to repair the array to full health. Spares **cannot** be added to non-redundant arrays (RAID 0) because the array will not survive the failure of a drive.

To add a spare, simply pass in the array and the new device to the mdadm --add command:

* sudo mdadm /dev/md0 --add /dev/sde

If the array is not in a degraded state, the new device will be added as a spare. If the device is currently degraded, the resync operation will immediately begin using the spare to replace the faulty drive.

After you add a spare, update the configuration file to reflect your new device orientation:

* sudo nano /etc/mdadm/mdadm.conf

Remove or comment out the current line that corresponds to your array definition:

/etc/mdadm/mdadm.conf

. . .

# ARRAY /dev/md0 metadata=1.2 name=mdadmwrite:0 UUID=d81c843b:4d96d9fc:5f3f499c:6ee99294

Afterwards, append your current configuration:

* sudo mdadm --detail --brief /dev/md0 | sudo tee -a /etc/mdadm/mdadm.conf

The new information will be used by the mdadm utility to assemble the array.

Increasing the Number of Active Devices in an Array

It is possible to grow an array by increasing the number of active devices within the assembly. The exact procedure depends slightly on the RAID level you are using.

**With RAID 1 or 10**

Begin by adding the new device as a spare, just as demonstrated in the last section:

* sudo mdadm /dev/md0 --add /dev/sde

Find out the current number of RAID devices in the array:

* sudo mdadm --detail /dev/md0

Output

/dev/md0:

Version : 1.2

Creation Time : Wed Aug 10 15:29:26 2016

Raid Level : raid1

Array Size : 104792064 (99.94 GiB 107.31 GB)

Used Dev Size : 104792064 (99.94 GiB 107.31 GB)

Raid Devices : 2

Total Devices : 3

Persistence : Superblock is persistent

. . .

We can see that in this example, the array is configured to actively use two devices, and that the total number of devices available to the array is three (because we added a spare).

Now, reconfigure the array to have an additional active device. The spare will be used to satisfy the extra drive requirement:

* sudo mdadm --grow --raid-devices=3 /dev/md0

The array will begin to reconfigure with an additional active disk. To view the progress of syncing the data, type:

* cat /proc/mdstat

You can continue to use the device as the process completes.

**With RAID 5 or 6**

Begin by adding the new device as a spare, just as demonstrated in the last section:

* sudo mdadm /dev/md0 --add /dev/sde

Find out the current number of RAID devices in the array:

* sudo mdadm --detail /dev/md0

Output

/dev/md0:

Version : 1.2

Creation Time : Wed Aug 10 18:38:51 2016

Raid Level : raid5

Array Size : 209584128 (199.88 GiB 214.61 GB)

Used Dev Size : 104792064 (99.94 GiB 107.31 GB)

Raid Devices : 3

Total Devices : 4

Persistence : Superblock is persistent

. . .

We can see that in this example, the array is configured to actively use three devices, and that the total number of devices available to the array is four (because we added a spare).

Now, reconfigure the array to have an additional active device. The spare will be used to satisfy the extra drive requirement. When growing a RAID 5 or RAID 6 array, it is important to include an additional option called --backup-file. This should point to a location **off** the array where a backup file containing critical information will be stored.

Note

The backup file is only used for a very short but critical time during this process, after which it will be deleted automatically. Because the time when this is needed is very brief, you will likely never see the file on disk, but in the event that something goes wrong, it can be used to rebuild the array. [This post](http://www.spinics.net/lists/raid/msg47479.html) has some additional information if you would like to know more.

* sudo mdadm --grow --raid-devices=4 --backup-file=/root/md0\_grow.bak /dev/md0

The following output indicates that the critical section will be backed up:

Output

mdadm: Need to backup 3072K of critical section..

The array will begin to reconfigure with an additional active disk. To view the progress of syncing the data, type:

* cat /proc/mdstat

You can continue to use the device as this process completes.

After the reshape is complete, you will need to expand the filesystem on the array to utilize the additional space:

* sudo resize2fs /dev/md0

Your array should now have a filesystem that matches its capacity.

**With RAID 0**

Because RAID 0 arrays cannot have spare drives (there is no chance for a spare to rebuild a damaged RAID 0 array), we must add the new device at the same time that we grow the array.

First, find out the current number of RAID devices in the array:

* sudo mdadm --detail /dev/md0

Output

/dev/md0:

Version : 1.2

Creation Time : Wed Aug 10 19:17:14 2016

Raid Level : raid0

Array Size : 209584128 (199.88 GiB 214.61 GB)

Raid Devices : 2

Total Devices : 2

Persistence : Superblock is persistent

. . .

We can now increment the number of RAID devices in the same operation as the new drive addition:

* sudo mdadm --grow /dev/md0 --raid-devices=3 --add /dev/sdc

You will see output indicating that the array has been changed to RAID 4:

Output

mdadm: level of /dev/md0 changed to raid4

mdadm: added /dev/sdc

This is normal and expected. The array will transition back into RAID 0 when the data has been redistributed to all existing disks.

You can check the progress of the action by typing:

* cat /proc/mdstat

Once the sync is complete, resize the filesystem to use the additional space:

* sudo resize2fs /dev/md0

Your array should now have a filesystem that matches its capacity.

Removing a Device from an Array

Removing a drive from a RAID array is sometimes necessary if there is a fault or if you need to switch out the disk.

For a device to be removed, it must first be marked as "failed" within the array. You can check if there is a failed device by using mdadm --detail:

* sudo mdadm --detail /dev/md0

Output

/dev/md0:

Version : 1.2

Creation Time : Wed Aug 10 21:42:12 2016

Raid Level : raid5

Array Size : 209584128 (199.88 GiB 214.61 GB)

Used Dev Size : 104792064 (99.94 GiB 107.31 GB)

Raid Devices : 3

Total Devices : 3

Persistence : Superblock is persistent

Update Time : Thu Aug 11 14:10:43 2016

State : clean, degraded

Active Devices : 2

Working Devices : 2

Failed Devices : 1

Spare Devices : 0

Layout : left-symmetric

Chunk Size : 64K

Name : mdadmwrite:0 (local to host mdadmwrite)

UUID : bf7a711b:b3aa9440:40d2c12e:79824706

Events : 144

Number Major Minor RaidDevice State

0 0 0 0 removed

1 8 0 1 active sync /dev/sda

2 8 16 2 active sync /dev/sdb

0 8 32 - faulty /dev/sdc

The highlighted lines all indicate that a drive is no longer functioning (/dev/sdc in this example).

If you need to remove a drive that does not have a problem, you can manually mark it as failed with the --fail option:

* sudo mdadm /dev/md0 --fail /dev/sdc

Output

mdadm: set /dev/sdc faulty in /dev/md0

If you look at the output of mdadm --detail, you should see that the device is now marked faulty.

Once the device is failed, you can remove it from the array with mdadm --remove:

* sudo mdadm /dev/md0 --remove /dev/sdc

Output

mdadm: hot removed /dev/sdc from /dev/md0

You can then replace it with a new drive, using the same mdadm --add command that you use to add a spare:

* sudo mdadm /dev/md0 --add /dev/sdd

Output

mdadm: added /dev/sdd

The array will begin to recover by copying data to the new drive.

Deleting an Array

To destroy an array, including all data contained within, begin by following the process we used to stop an array.

First, unmount the filesystem:

* cd ~
* sudo umount /mnt/md0

Next, stop the array:

* sudo mdadm --stop /dev/md0

Afterwards, delete the array itself with the --remove command targeting the RAID device:

* sudo mdadm --remove /dev/md0

Once the array itself is removed, you should use mdadm --zero-superblock on each of the component devices. This will erase the md superblock, a header used by mdadm to assemble and manage the component devices as part of an array. If this is still present, it may cause problems when trying to reuse the disk for other purposes.

You can see that the superblock is still present in the array by checking out the FSTYPE column in the lsblk --fs output:

* lsblk --fs

Output

NAME FSTYPE LABEL UUID MOUNTPOINT

sda linux\_raid\_member mdadmwrite:0 bf7a711b-b3aa-9440-40d2-c12e79824706

sdb linux\_raid\_member mdadmwrite:0 bf7a711b-b3aa-9440-40d2-c12e79824706

sdc linux\_raid\_member mdadmwrite:0 bf7a711b-b3aa-9440-40d2-c12e79824706

sdd

vda

├─vda1 ext4 DOROOT 4f8b85db-8c11-422b-83c4-c74195f67b91 /

└─vda15

In this example, /dev/sda, /dev/sdb, and /dev/sdc were all part of the array and are still labeled as such.

Remove the labels by typing:

* sudo mdadm --zero-superblock /dev/sda /dev/sdb /dev/sdc

Next, make sure you remove or comment out any references to the array in the /etc/fstab file:

* sudo nano /etc/fstab

/etc/fstab

. . .

# /dev/md0 /mnt/md0 ext4 defaults,nofail,discard 0 0

Save and close the file when you are finished.

Remove or comment out any references to the array from the /etc/mdadm/mdadm.conf file as well:

* nano /etc/mdadm/mdadm.conf

/etc/mdadm/mdadm.conf

# ARRAY /dev/md0 metadata=1.2 name=mdadmwrite:0 UUID=bf7a711b:b3aa9440:40d2c12e:79824706

Save and close the file when you are finished.

Update the initramfs by typing:

* sudo update-initramfs -u

This should remove the device from the early boot environment.

Conclusion

Linux's mdadm utility makes it fairly easy to manage arrays once you understand the conventions it uses and the places where you can look for information. This guide is in no ways exhaustive, but serves to introduce some of the management tasks that you might need to perform on a day-to-day basis.

Once you're comfortable creating and managing RAID arrays with mdadm, there are a number of different directions you can explore next. Volume management layers like LVM integrate tightly with RAID and allow you to flexibly partition space into logical volumes. Similarly, LUKS and dm-crypt encryption is commonly used to encrypt the RAID devices prior to writing the filesystem. Linux allows all of these technologies to be used together to enhance your storage capabilities.