https://www.digitalocean.com/community/tutorials/how-to-create-raid-arrays-with-mdadm-on-ubuntu-16-04

How To Create RAID Arrays with mdadm on Ubuntu 16.04

### Introduction

The mdadm utility can be used to create and manage storage arrays using Linux's software RAID capabilities. Administrators have great flexibility in coordinating their individual storage devices and creating logical storage devices that have greater performance or redundancy characteristics.

In this guide, we will go over a number of different RAID configurations that can be set up using an Ubuntu 16.04 server.

## Prerequisites

In order to complete the steps in this guide, you should have:

* **A non-root user with sudo privileges on an Ubuntu 16.04 server**: The steps in this guide will be completed with a sudo user. To learn how to set up an account with these privileges, follow our [Ubuntu 16.04 initial server setup guide](https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-16-04).
* **A basic understanding of RAID terminology and concepts**: While this guide will touch on some RAID terminology in passing, a more complete understanding is very useful. To learn more about RAID and to get a better understanding of what RAID level is right for you, read our [introduction to RAID article](https://www.digitalocean.com/community/tutorials/an-introduction-to-raid-terminology-and-concepts).
* **Multiple raw storage devices available on your server**: We will be demonstrating how to configure various types of arrays on the server. As such, you will need some drives to configure. If you are using DigitalOcean, you can use [Block Storage volumes](https://www.digitalocean.com/community/tutorials/how-to-use-block-storage-on-digitalocean) to fill this role. Depending on the array type, you will need at minimum between **two to four storage devices**.

## Resetting Existing RAID Devices

Throughout this guide, we will be introducing the steps to create a number of different RAID levels. If you wish to follow along, you will likely want to reuse your storage devices after each section. This section can be referenced to learn how to quickly reset your component storage devices prior to testing a new RAID level. Skip this section for now if you have not yet set up any arrays.

Warning

This process will completely destroy the array and any data written to it. Make sure that you are operating on the correct array and that you have copied off any data you need to retain prior to destroying the array.

Find the active arrays in the /proc/mdstat file by typing:

* cat /proc/mdstat

Output

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

md0 : active raid0 sdc[1] sdd[0]

209584128 blocks super 1.2 512k chunks

unused devices: <none>

Unmount the array from the filesystem:

* sudo umount /dev/md0

Then, stop and remove the array by typing:

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

Find the devices that were used to build the array with the following command:

Note

Keep in mind that the /dev/sd\* names can change any time you reboot! Check them every time to make sure you are operating on the correct devices.

* lsblk -o NAME,SIZE,FSTYPE,TYPE,MOUNTPOINT

Output

NAME SIZE FSTYPE TYPE MOUNTPOINT

sda 100G disk

sdb 100G disk

sdc 100G linux\_raid\_member disk

sdd 100G linux\_raid\_member disk

vda 20G disk

├─vda1 20G ext4 part /

└─vda15 1M part

After discovering the devices used to create an array, zero their superblock to reset them to normal:

* sudo mdadm --zero-superblock /dev/sdc
* sudo mdadm --zero-superblock /dev/sdd

You should remove any of the persistent references to the array. Edit the /etc/fstab file and comment out or remove the reference to your array:

* sudo nano /etc/fstab

/etc/fstab

. . .

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

Also, comment out or remove the array definition from the /etc/mdadm/mdadm.conf file:

* sudo nano /etc/mdadm/mdadm.conf

/etc/mdadm/mdadm.conf

. . .

# ARRAY /dev/md0 metadata=1.2 name=mdadmwrite:0 UUID=7261fb9c:976d0d97:30bc63ce:85e76e91

Finally, update the initramfs again:

* sudo update-initramfs -u

At this point, you should be ready to reuse the storage devices individually, or as components of a different array.

## Creating a RAID 0 Array

The RAID 0 array works by breaking up data into chunks and striping it across the available disks. This means that each disk contains a portion of the data and that multiple disks will be referenced when retrieving information.

* Requirements: minimum of **2 storage devices**
* Primary benefit: Performance
* Things to keep in mind: Make sure that you have functional backups. A single device failure will destroy all data in the array.

### Identify the Component Devices

To get started, find the identifiers for the raw disks that you will be using:

* lsblk -o NAME,SIZE,FSTYPE,TYPE,MOUNTPOINT

Output

NAME SIZE FSTYPE TYPE MOUNTPOINT

sda 100G disk

sdb 100G disk

vda 20G disk

├─vda1 20G ext4 part /

└─vda15 1M part

As you can see above, we have two disks without a filesystem, each 100G in size. In this example, these devices have been given the /dev/sda and /dev/sdb identifiers for this session. These will be the raw components we will use to build the array.

### Create the Array

To create a RAID 0 array with these components, pass them in to the mdadm --create command. You will have to specify the device name you wish to create (/dev/md0 in our case), the RAID level, and the number of devices:

* sudo mdadm --create --verbose /dev/md0 --level=0 --raid-devices=2 /dev/sda /dev/sdb

You can ensure that the RAID was successfully created by checking the /proc/mdstat file:

* cat /proc/mdstat

Output

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

md0 : active raid0 sdb[1] sda[0]

209584128 blocks super 1.2 512k chunks

unused devices: <none>

As you can see in the highlighted line, the /dev/md0 device has been created in the RAID 0 configuration using the /dev/sda and /dev/sdb devices.

### Create and Mount the Filesystem

Next, create a filesystem on the array:

* sudo mkfs.ext4 -F /dev/md0

Create a mount point to attach the new filesystem:

* sudo mkdir -p /mnt/md0

You can mount the filesystem by typing:

* sudo mount /dev/md0 /mnt/md0

Check whether the new space is available by typing:

* df -h -x devtmpfs -x tmpfs

Output

Filesystem Size Used Avail Use% Mounted on

/dev/vda1 20G 1.1G 18G 6% /

/dev/md0 197G 60M 187G 1% /mnt/md0

The new filesystem is mounted and accessible.

### Save the Array Layout

To make sure that the array is reassembled automatically at boot, we will have to adjust the /etc/mdadm/mdadm.conf file. You can automatically scan the active array and append the file by typing:

* sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf

Afterwards, you can update the initramfs, or initial RAM file system, so that the array will be available during the early boot process:

* sudo update-initramfs -u

Add the new filesystem mount options to the /etc/fstab file for automatic mounting at boot:

* echo '/dev/md0 /mnt/md0 ext4 defaults,nofail,discard 0 0' | sudo tee -a /etc/fstab

Your RAID 0 array should now automatically be assembled and mounted each boot.

## Creating a RAID 1 Array

The RAID 1 array type is implemented by mirroring data across all available disks. Each disk in a RAID 1 array gets a full copy of the data, providing redundancy in the event of a device failure.

* Requirements: minimum of **2 storage devices**
* Primary benefit: Redundancy
* Things to keep in mind: Since two copies of the data are maintained, only half of the disk space will be usable

### Identify the Component Devices

To get started, find the identifiers for the raw disks that you will be using:

* lsblk -o NAME,SIZE,FSTYPE,TYPE,MOUNTPOINT

Output

NAME SIZE FSTYPE TYPE MOUNTPOINT

sda 100G disk

sdb 100G disk

vda 20G disk

├─vda1 20G ext4 part /

└─vda15 1M part

As you can see above, we have two disks without a filesystem, each 100G in size. In this example, these devices have been given the /dev/sda and /dev/sdb identifiers for this session. These will be the raw components we will use to build the array.

### Create the Array

To create a RAID 1 array with these components, pass them in to the mdadm --create command. You will have to specify the device name you wish to create (/dev/md0 in our case), the RAID level, and the number of devices:

* sudo mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/sda /dev/sdb

If the component devices you are using are not partitions with the boot flag enabled, you will likely be given the following warning. It is safe to type **y** to continue:

Output

mdadm: Note: this array has metadata at the start and

may not be suitable as a boot device. If you plan to

store '/boot' on this device please ensure that

your boot-loader understands md/v1.x metadata, or use

--metadata=0.90

mdadm: size set to 104792064K

Continue creating array? y

The mdadm tool will start to mirror the drives. This can take some time to complete, but the array can be used during this time. You can monitor the progress of the mirroring by checking the /proc/mdstat file:

* cat /proc/mdstat

Output

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

md0 : active raid1 sdb[1] sda[0]

104792064 blocks super 1.2 [2/2] [UU]

[====>................] resync = 20.2% (21233216/104792064) finish=6.9min speed=199507K/sec

unused devices: <none>

As you can see in the first highlighted line, the /dev/md0 device has been created in the RAID 1 configuration using the /dev/sda and /dev/sdb devices. The second highlighted line shows the progress on the mirroring. You can continue the guide while this process completes.

### Create and Mount the Filesystem

Next, create a filesystem on the array:

* sudo mkfs.ext4 -F /dev/md0

Create a mount point to attach the new filesystem:

* sudo mkdir -p /mnt/md0

You can mount the filesystem by typing:

* sudo mount /dev/md0 /mnt/md0

Check whether the new space is available by typing:

* df -h -x devtmpfs -x tmpfs

Output

Filesystem Size Used Avail Use% Mounted on

/dev/vda1 20G 1.1G 18G 6% /

/dev/md0 99G 60M 94G 1% /mnt/md0

The new filesystem is mounted and accessible.

### Save the Array Layout

To make sure that the array is reassembled automatically at boot, we will have to adjust the /etc/mdadm/mdadm.conf file. You can automatically scan the active array and append the file by typing:

* sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf

Afterwards, you can update the initramfs, or initial RAM file system, so that the array will be available during the early boot process:

* sudo update-initramfs -u

Add the new filesystem mount options to the /etc/fstab file for automatic mounting at boot:

* echo '/dev/md0 /mnt/md0 ext4 defaults,nofail,discard 0 0' | sudo tee -a /etc/fstab

Your RAID 1 array should now automatically be assembled and mounted each boot.

## Creating a RAID 5 Array

The RAID 5 array type is implemented by striping data across the available devices. One component of each stripe is a calculated parity block. If a device fails, the parity block and the remaining blocks can be used to calculate the missing data. The device that receives the parity block is rotated so that each device has a balanced amount of parity information.

* Requirements: minimum of **3 storage devices**
* Primary benefit: Redundancy with more usable capacity.
* Things to keep in mind: While the parity information is distributed, one disk's worth of capacity will be used for parity. RAID 5 can suffer from very poor performance when in a degraded state.

### Identify the Component Devices

To get started, find the identifiers for the raw disks that you will be using:

* lsblk -o NAME,SIZE,FSTYPE,TYPE,MOUNTPOINT

Output

NAME SIZE FSTYPE TYPE MOUNTPOINT

sda 100G disk

sdb 100G disk

sdc 100G disk

vda 20G disk

├─vda1 20G ext4 part /

└─vda15 1M part

As you can see above, we have three disks without a filesystem, each 100G in size. In this example, these devices have been given the /dev/sda, /dev/sdb, and /dev/sdc identifiers for this session. These will be the raw components we will use to build the array.

### Create the Array

To create a RAID 5 array with these components, pass them in to the mdadm --create command. You will have to specify the device name you wish to create (/dev/md0 in our case), the RAID level, and the number of devices:

* sudo mdadm --create --verbose /dev/md0 --level=5 --raid-devices=3 /dev/sda /dev/sdb /dev/sdc

The mdadm tool will start to configure the array (it actually uses the recovery process to build the array for performance reasons). This can take some time to complete, but the array can be used during this time. You can monitor the progress of the mirroring by checking the /proc/mdstat file:

* cat /proc/mdstat

Output

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

md0 : active raid5 sdc[3] sdb[1] sda[0]

209584128 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/2] [UU\_]

[===>.................] recovery = 15.6% (16362536/104792064) finish=7.3min speed=200808K/sec

unused devices: <none>

As you can see in the first highlighted line, the /dev/md0 device has been created in the RAID 5 configuration using the /dev/sda, /dev/sdb and /dev/sdc devices. The second highlighted line shows the progress on the build. You can continue the guide while this process completes.

### Create and Mount the Filesystem

Next, create a filesystem on the array:

* sudo mkfs.ext4 -F /dev/md0

Create a mount point to attach the new filesystem:

* sudo mkdir -p /mnt/md0

You can mount the filesystem by typing:

* sudo mount /dev/md0 /mnt/md0

Check whether the new space is available by typing:

* df -h -x devtmpfs -x tmpfs

Output

Filesystem Size Used Avail Use% Mounted on

/dev/vda1 20G 1.1G 18G 6% /

/dev/md0 197G 60M 187G 1% /mnt/md0

The new filesystem is mounted and accessible.

### Save the Array Layout

To make sure that the array is reassembled automatically at boot, we will have to adjust the /etc/mdadm/mdadm.conf file.

Before you adjust the configuration, check again to make sure the array has finished assembling. Because of the way that mdadm builds RAID 5 arrays, if the array is still building, the number of spares in the array will be inaccurately reported:

* cat /proc/mdstat

Output

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

md0 : active raid5 sdc[3] sdb[1] sda[0]

209584128 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/3] [UUU]

unused devices: <none>

The output above shows that the rebuild is complete. Now, we can automatically scan the active array and append the file by typing:

* sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf

Afterwards, you can update the initramfs, or initial RAM file system, so that the array will be available during the early boot process:

* sudo update-initramfs -u

Add the new filesystem mount options to the /etc/fstab file for automatic mounting at boot:

* echo '/dev/md0 /mnt/md0 ext4 defaults,nofail,discard 0 0' | sudo tee -a /etc/fstab

Your RAID 5 array should now automatically be assembled and mounted each boot.

## Creating a RAID 6 Array

The RAID 6 array type is implemented by striping data across the available devices. Two components of each stripe are calculated parity blocks. If one or two devices fail, the parity blocks and the remaining blocks can be used to calculate the missing data. The devices that receive the parity blocks are rotated so that each device has a balanced amount of parity information. This is similar to a RAID 5 array, but allows for the failure of two drives.

* Requirements: minimum of **4 storage devices**
* Primary benefit: Double redundancy with more usable capacity.
* Things to keep in mind: While the parity information is distributed, two disk's worth of capacity will be used for parity. RAID 6 can suffer from very poor performance when in a degraded state.

### Identify the Component Devices

To get started, find the identifiers for the raw disks that you will be using:

* lsblk -o NAME,SIZE,FSTYPE,TYPE,MOUNTPOINT

Output

NAME SIZE FSTYPE TYPE MOUNTPOINT

sda 100G disk

sdb 100G disk

sdc 100G disk

sdd 100G disk

vda 20G disk

├─vda1 20G ext4 part /

└─vda15 1M part

As you can see above, we have four disks without a filesystem, each 100G in size. In this example, these devices have been given the /dev/sda, /dev/sdb, /dev/sdc, and /dev/sdd identifiers for this session. These will be the raw components we will use to build the array.

### Create the Array

To create a RAID 6 array with these components, pass them in to the mdadm --create command. You will have to specify the device name you wish to create (/dev/md0 in our case), the RAID level, and the number of devices:

* sudo mdadm --create --verbose /dev/md0 --level=6 --raid-devices=4 /dev/sda /dev/sdb /dev/sdc /dev/sdd

The mdadm tool will start to configure the array (it actually uses the recovery process to build the array for performance reasons). This can take some time to complete, but the array can be used during this time. You can monitor the progress of the mirroring by checking the /proc/mdstat file:

* cat /proc/mdstat

Output

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

md0 : active raid6 sdd[3] sdc[2] sdb[1] sda[0]

209584128 blocks super 1.2 level 6, 512k chunk, algorithm 2 [4/4] [UUUU]

[>....................] resync = 0.6% (668572/104792064) finish=10.3min speed=167143K/sec

unused devices: <none>

As you can see in the first highlighted line, the /dev/md0 device has been created in the RAID 6 configuration using the /dev/sda, /dev/sdb, /dev/sdc and /dev/sdd devices. The second highlighted line shows the progress on the build. You can continue the guide while this process completes.

### Create and Mount the Filesystem

Next, create a filesystem on the array:

* sudo mkfs.ext4 -F /dev/md0

Create a mount point to attach the new filesystem:

* sudo mkdir -p /mnt/md0

You can mount the filesystem by typing:

* sudo mount /dev/md0 /mnt/md0

Check whether the new space is available by typing:

* df -h -x devtmpfs -x tmpfs

Output

Filesystem Size Used Avail Use% Mounted on

/dev/vda1 20G 1.1G 18G 6% /

/dev/md0 197G 60M 187G 1% /mnt/md0

The new filesystem is mounted and accessible.

### Save the Array Layout

To make sure that the array is reassembled automatically at boot, we will have to adjust the /etc/mdadm/mdadm.conf file. We can automatically scan the active array and append the file by typing:

* sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf

Afterwards, you can update the initramfs, or initial RAM file system, so that the array will be available during the early boot process:

* sudo update-initramfs -u

Add the new filesystem mount options to the /etc/fstab file for automatic mounting at boot:

* echo '/dev/md0 /mnt/md0 ext4 defaults,nofail,discard 0 0' | sudo tee -a /etc/fstab

Your RAID 6 array should now automatically be assembled and mounted each boot.

## Creating a Complex RAID 10 Array

The RAID 10 array type is traditionally implemented by creating a striped RAID 0 array composed of sets of RAID 1 arrays. This nested array type gives both redundancy and high performance, at the expense of large amounts of disk space. The mdadm utility has its own RAID 10 type that provides the same type of benefits with increased flexibility. It is not created by nesting arrays, but has many of the same characteristics and guarantees. We will be using the mdadm RAID 10 here.

* Requirements: minimum of **3 storage devices**
* Primary benefit: Performance and redundancy
* Things to keep in mind: The amount of capacity reduction for the array is defined by the number of data copies you choose to keep. The number of copies that are stored with mdadm style RAID 10 is configurable.

By default, two copies of each data block will be stored in what is called the "near" layout. The possible layouts that dictate how each data block is stored are:

* **near**: The default arrangement. Copies of each chunk are written consecutively when striping, meaning that the copies of the data blocks will be written around the same part of multiple disks.
* **far**: The first and subsequent copies are written to different parts the storage devices in the array. For instance, the first chunk might be written near the beginning of a disk, while the second chunk would be written half way down on a different disk. This can give some read performance gains for traditional spinning disks at the expense of write performance.
* **offset**: Each stripe is copied, offset by one drive. This means that the copies are offset from one another, but still close together on the disk. This helps minimize excessive seeking during some workloads.

You can find out more about these layouts by checking out the "RAID10" section of this man page:

* man 4 md

You can also find this man page online [here](http://manpages.ubuntu.com/manpages/xenial/man4/md.4.html).

### Identify the Component Devices

To get started, find the identifiers for the raw disks that you will be using:

* lsblk -o NAME,SIZE,FSTYPE,TYPE,MOUNTPOINT

Output

NAME SIZE FSTYPE TYPE MOUNTPOINT

sda 100G disk

sdb 100G disk

sdc 100G disk

sdd 100G disk

vda 20G disk

├─vda1 20G ext4 part /

└─vda15 1M part

As you can see above, we have four disks without a filesystem, each 100G in size. In this example, these devices have been given the /dev/sda, /dev/sdb, /dev/sdc, and /dev/sdd identifiers for this session. These will be the raw components we will use to build the array.

### Create the Array

To create a RAID 10 array with these components, pass them in to the mdadm --create command. You will have to specify the device name you wish to create (/dev/md0 in our case), the RAID level, and the number of devices.

You can set up two copies using the near layout by not specifying a layout and copy number:

* sudo mdadm --create --verbose /dev/md0 --level=10 --raid-devices=4 /dev/sda /dev/sdb /dev/sdc /dev/sdd

If you want to use a different layout, or change the number of copies, you will have to use the --layout=option, which takes a layout and copy identifier. The layouts are **n** for near, **f** for far, and **o** for offset. The number of copies to store is appended afterwards.

For instance, to create an array that has 3 copies in the offset layout, the command would look like this:

* sudo mdadm --create --verbose /dev/md0 --level=10 --layout=o3 --raid-devices=4 /dev/sda /dev/sdb /dev/sdc /dev/sdd

The mdadm tool will start to configure the array (it actually uses the recovery process to build the array for performance reasons). This can take some time to complete, but the array can be used during this time. You can monitor the progress of the mirroring by checking the /proc/mdstat file:

* 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]

[===>.................] resync = 18.1% (37959424/209584128) finish=13.8min speed=206120K/sec

unused devices: <none>

As you can see in the first highlighted line, the /dev/md0 device has been created in the RAID 10 configuration using the /dev/sda, /dev/sdb, /dev/sdc and /dev/sdd devices. The second highlighted area shows the layout that was used for this example (2 copies in the near configuration). The third highlighted area shows the progress on the build. You can continue the guide while this process completes.

### Create and Mount the Filesystem

Next, create a filesystem on the array:

* sudo mkfs.ext4 -F /dev/md0

Create a mount point to attach the new filesystem:

* sudo mkdir -p /mnt/md0

You can mount the filesystem by typing:

* sudo mount /dev/md0 /mnt/md0

Check whether the new space is available by typing:

* df -h -x devtmpfs -x tmpfs

Output

Filesystem Size Used Avail Use% Mounted on

/dev/vda1 20G 1.1G 18G 6% /

/dev/md0 197G 60M 187G 1% /mnt/md0

The new filesystem is mounted and accessible.

### Save the Array Layout

To make sure that the array is reassembled automatically at boot, we will have to adjust the /etc/mdadm/mdadm.conf file. We can automatically scan the active array and append the file by typing:

* sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf

Afterwards, you can update the initramfs, or initial RAM file system, so that the array will be available during the early boot process:

* sudo update-initramfs -u

Add the new filesystem mount options to the /etc/fstab file for automatic mounting at boot:

* echo '/dev/md0 /mnt/md0 ext4 defaults,nofail,discard 0 0' | sudo tee -a /etc/fstab

Your RAID 10 array should now automatically be assembled and mounted each boot.

## Conclusion

In this guide, we demonstrated how to create various types of arrays using Linux's mdadm software RAID utility. RAID arrays offer some compelling redundancy and performance enhancements over using multiple disks individually.

Once you have settled on the type of array needed for your environment and created the device, you will need to learn how to perform day-to-day management with mdadm. Our guide on [how to manage RAID arrays with mdadm on Ubuntu 16.04](https://www.digitalocean.com/community/tutorials/how-to-manage-raid-arrays-with-mdadm-on-ubuntu-16-04) can help get you started.