Products

Solutions

Developers

Partners

Pricing

DigitalOcean

Log in

Sign up

Blog

Docs

Get Support

Contact Sales

Tutorials Questions Learning Paths For Businesses Product Docs

CONTENTS

// TUTORIAL //

How To Create RAID Arrays with mdadm on Ubuntu 18.04

Updated on October 20, 2022

Storage Block Storage Ubuntu 18.04

Justin Ellingwood

Not using Ubuntu 18.04?
Choose a different version or distribution.

Ubuntu 18.04

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, you will perform different RAID configurations that can be set up using an Ubuntu 18.04 server.

Prerequisites

To follow the steps in this guide, you will need:

- A non-**root** user with `sudo` privileges on an Ubuntu 18.04 server. To learn how to set up an account with these privileges, follow our Ubuntu 18.04 initial server setup guide.
- A basic understanding of RAID terminology and concepts. To learn more about RAID and what RAID level is right for you, read our introduction to RAID article.
- Multiple raw storage devices available on your server. The examples in this tutorial demonstrate how to configure various types of arrays on the server. As such, you will need some drives to configure.
- Depending on the array type, you will need **two to four storage devices**. These drives do not need to be formatted prior to following this guide.

> **Info:** Due to the inefficiency of RAID setups on virtual private servers, we don’t recommend deploying a RAID setup on DigitalOcean droplets. The efficiency of datacenter disk replication makes the benefits of a RAID negligible, relative to a setup on baremetal hardware. This tutorial aims to be a reference for a conventional RAID setup.

Resetting Existing RAID Devices (Optional)

You can skip this section for now if you have not yet set up any arrays. This guide will introduce a number of different RAID levels. If you wish to follow along and complete each RAID level for your devices, you will likely want to reuse your storage devices after each section. This specific section **Resetting Existing RAID Devices** can be referenced to reset your component storage devices prior to testing a new RAID level.

> **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 any data you need to retain prior to destroying the array.

Begin by finding the active arrays in the `/proc/mdstat` file:

```
$ 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>
```

Then unmount the array from the filesystem:

```
$ sudo umount /dev/ md0
```

Now stop and remove the array:

```
$ sudo mdadm --stop /dev/ md0
```

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

> **Warning:** 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 linux_raid_member disk
 sdb       100G linux_raid_member disk
sdc       100G                    disk
sdd       100G                    disk
vda        25G                    disk
├─vda1   24.9G ext4               part /
├─vda14     4M                    part
└─vda15   106M vfat               part /boot/efi
vdb       466K iso9660            disk
```

After discovering the devices used to create an array, zero their *superblock* which holds metadata for the RAID setup. Zeroing this removes the RAID metadata and resets them to normal:

```
$ sudo mdadm --zero-superblock /dev/ sda
$ sudo mdadm --zero-superblock /dev/ sdb
```

It’s recommended to also remove any persistent references to the array. Edit the `/etc/fstab` file and comment out or remove the reference to your array. You can comment it out by inserting a hashtag symbol `#` at the beginning of the line, using `nano` or your preferred text editor:

```
$ sudo nano /etc/fstab
```

Copy

/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
```

Copy

/etc/mdadm/mdadm.conf

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

Finally, update the `initramfs` again so that the early boot process does not try to bring an unavailable array online:

```
$ sudo update-initramfs -u
```

Copy

From here, 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 in terms of read/write and capacity.

- Things to keep in mind: Make sure that you have functional backups. A single device failure will destroy all data in the array.

Identifying the Component Devices

To start, 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       25G           disk
├─vda1   24.9G ext4     part /
├─vda14    4M           part
└─vda15  106M vfat     part /boot/efi
vdb      466K iso9660  disk
```

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

Creating the Array

To create a RAID 0 array with these components, pass them into the `mdadm --create` command. You will have to specify the device name you wish to create, the RAID level, and the number of devices. In this command example, you will be naming the device `/dev/md0`, and include the two disks that will build the array:

```
$ sudo mdadm --create --verbose /dev/md0 --level=0 --raid-devices=2 /dev/
```

Confirm 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>
```

This output reveals that the `/dev/md0` device was created in the RAID 0 configuration using the `/dev/sda` and `/dev/sdb` devices.

Creating and Mounting the Filesystem

Next, create a filesystem on the array:

```
$ sudo mkfs.ext4 -F /dev/md0
```

Copy

Then, create a mount point to attach the new filesystem:

```
$ sudo mkdir -p /mnt/md0
```

Copy

You can mount the filesystem with the following command:

```
$ sudo mount /dev/md0 /mnt/md0
```

Copy

After, check whether the new space is available:

```
$ df -h -x devtmpfs -x tmpfs
```

Copy

```
Output
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        25G  1.4G   23G   6% /
/dev/vda15      105M  3.4M  102M   4% /boot/efi
 /dev/md0        196G   61M  186G   1% /mnt/md0
```

The new filesystem is now mounted and accessible.

Saving the Array Layout

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

```
$ 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 / tab
```

Your RAID 0 array will now automatically assemble and mount each boot.

You're now finished with your RAID set up. If you want to try a different RAID, follow the resetting instructions at the beginning of this tutorial to proceed with creating a new RAID array type.

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 between two storage devices.
- Things to keep in mind: Since two copies of the data are maintained, only half of the disk space will be usable.

Identifying the Component Devices

To start, 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       25G           disk
├─vda1  24.9G ext4      part /
├─vda14    4M           part
└─vda15  106M vfat      part /boot/efi
vdb      466K iso9660   disk
```

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

Creating the Array

To create a RAID 1 array with these components, pass them into the `mdadm --create` command. You will have to specify the device name you wish to create, the RAID level, and the number of devices. In this command example, you will be naming the device `/dev/md0`, and include the disks that will build the array:

```
$ sudo mdadm --create --verbose /dev/md0 --level=1 --raid-devices=2 /dev/
```

Copy

If the component devices you are using are not partitions with the `boot` flag enabled, you will likely receive the following warning. It is safe to respond with `y` and 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
```

Copy

```
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 spe

unused devices: <none>
```

In the first highlighted line, the `/dev/md0` device was created in the RAID 1 configuration using the `/dev/sda` and `/dev/sdb` devices. The second highlighted line reveals the progress on the mirroring. You can continue to the next step while this process completes.

Creating and Mounting the Filesystem

Next, create a filesystem on the array:

```
$ sudo mkfs.ext4 -F /dev/md0
```

Then, create a mount point to attach the new filesystem:

```
$ sudo mkdir -p /mnt/md0
```

You can mount the filesystem by running the following:

```
$ sudo mount /dev/md0 /mnt/md0
```

Check whether the new space is available:

```
$ df -h -x devtmpfs -x tmpfs
```

```
Output
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        25G  1.4G   23G   6% /
/dev/vda15      105M  3.4M  102M   4% /boot/efi
 /dev/md0         99G   60M   94G   1% /mnt/md0
```

The new filesystem is mounted and accessible.

Saving the Array Layout

To make sure that the array is reassembled automatically at boot, you have to adjust the `/etc/mdadm/mdadm.conf` file. You can automatically scan the active array and append the file with the following:

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

Afterward, 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 /
```

Your RAID 1 array will now automatically assemble and mount each boot.

You’re now finished with your RAID set up. If you want to try a different RAID, follow the resetting instructions at the beginning of this tutorial to proceed with creating a new RAID array type.

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.

Identifying the Component Devices

To start, 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       25G          disk
├─vda1  24.9G ext4     part /
├─vda14    4M          part
└─vda15  106M vfat     part /boot/efi
vdb      466K iso9660  disk
```

You have three disks without a filesystem, each 100G in size. These devices have been given the `/dev/sda`, `/dev/sdb`, and `/dev/sdc` identifiers for this session and will be the raw components you use to build the array.

Creating the Array

To create a RAID 5 array with these components, pass them into the `mdadm --create` command. You will have to specify the device name you wish to create, the RAID level, and the number of devices. In this command example, you will be naming the device `/dev/md0`, and include the disks that will build the array:

```
$ sudo mdadm --create --verbose /dev/md0 --level=5 --raid-devices=3 /dev/ Copy dev
```

The `mdadm` tool will start to configure the array. It 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                                                      Copy
```

```
Output
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [raid10]
md0 : active raid5 sdc[3] sdb[1] sda[0]
      209582080 blocks super 1.2 level 5, 512k chunk, algorithm 2 [3/2] [UU_]
      [>....................]  recovery =  0.9% (957244/104791040) finish=18.0min sp

unused devices: <none>
```

In the first highlighted line, the `/dev/md0` device was created in the RAID 5 configuration using the `/dev/sda`, `/dev/sdb` and `/dev/sdc` devices. The second highlighted line shows the progress of the build.

> **Warning:** Due to the way that `mdadm` builds RAID 5 arrays, while the array is still building, the number of spares in the array will be inaccurately reported. This means that you must wait for the array to finish assembling before updating the `/etc/mdadm/mdadm.conf` file. If you update the configuration file while the array is still building, the system will have incorrect information about the array state and will be unable to assemble it automatically at boot with the correct name.

You can continue the guide while this process completes.

Creating and Mounting the Filesystem

Next, create a filesystem on the array:

```
$ sudo mkfs.ext4 -F /dev/md0
```

Copy

Create a mount point to attach the new filesystem:

```
$ sudo mkdir -p /mnt/md0
```

Copy

You can mount the filesystem with the following:

```
$ sudo mount /dev/md0 /mnt/md0
```

Copy

Check whether the new space is available:

```
$ df -h -x devtmpfs -x tmpfs
```

Copy

```
Output
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        25G  1.4G   23G   6% /
/dev/vda15      105M  3.4M  102M   4% /boot/efi
 /dev/md0        197G   60M  187G    1% /mnt/md0
```

The new filesystem is mounted and accessible.

Saving the Array Layout

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

> **Warning:** As mentioned previously, before you adjust the configuration, check again to make sure the array has finished assembling. Completing the following steps before the array is built will prevent the system from assembling the array correctly on reboot.

You can monitor the progress of the mirroring by checking the `/proc/mdstat` file:

```
$ cat /proc/mdstat                                                    Copy
```

```
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>
```

This output reveals that the rebuild is complete. Now, you can automatically scan the active array and append the file:

```
$ sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf         Copy
```

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                                              Copy
```

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 / Copy tab
```

Your RAID 5 array will now automatically assemble and mount each boot.

You're now finished with your RAID set up. If you want to try a different RAID, follow the resetting instructions at the beginning of this tutorial to proceed with creating a new RAID array type.

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 disks worth of capacity will be used for parity. RAID 6 can suffer from very poor performance when in a degraded state.

Identifying the Component Devices

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

```
$ lsblk -o NAME,SIZE,FSTYPE,TYPE,MOUNTPOINT
```

Copy

```
Output
NAME     SIZE FSTYPE   TYPE MOUNTPOINT
 sda      100G          disk
 sdb      100G          disk
 sdc      100G          disk
 sdd      100G          disk
vda       25G          disk
├─vda1  24.9G ext4     part /
├─vda14    4M          part
└─vda15  106M vfat     part /boot/efi
vdb      466K iso9660  disk
```

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

Creating the Array

To create a RAID 6 array with these components, pass them into the `mdadm --create` command. You have to specify the device name you wish to create, the RAID level, and the number of devices. In this following command example, you will be naming the device `/dev/md0` and include the disks that will build the array :

```
$ sudo mdadm --create --verbose /dev/md0 --level=6 --raid-devices=4 /dev/
```

Copy

The `mdadm` tool will start to configure the array. It 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
```

Copy

```
Output
Personalities : [linear] [multipath] [raid0] [raid1] [raid6] [raid5] [raid4] [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 spee

unused devices: <none>
```

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 of the build. You can continue the guide while this process completes.

Creating and Mounting the Filesystem

Next, create a filesystem on the array:

```
$ sudo mkfs.ext4 -F /dev/md0
```

Copy

Create a mount point to attach the new filesystem:

```
$ sudo mkdir -p /mnt/md0
```

Copy

You can mount the filesystem with the following:

```
$ sudo mount /dev/md0 /mnt/md0
```

Check whether the new space is available:

```
$ df -h -x devtmpfs -x tmpfs
```

```
Output
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        25G  1.4G   23G   6% /
/dev/vda15      105M  3.4M  102M   4% /boot/efi
 /dev/md0        197G   60M  187G   1% /mnt/md0
```

The new filesystem is mounted and accessible.

Saving the Array Layout

To make sure that the array is reassembled automatically at boot, you 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 /
```

Your RAID 6 array will now automatically assemble and mount each boot.

You're now finished with your RAID set up. If you want to try a different RAID, follow the resetting instructions at the beginning of this tutorial to proceed with creating a new RAID array type.

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. You 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 as follows:

- **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 of 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 halfway 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, and 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.

Identifying the Component Devices

To start, 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        25G          disk
├─vda1  24.9G ext4     part /
├─vda14    4M          part
└─vda15  106M vfat     part /boot/efi
vdb       466K iso9660  disk
```

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

Creating the Array

To create a RAID 10 array with these components, pass them into the `mdadm --create` command. You have to specify the device name you wish to create, the RAID level, and the number of devices. In this following command example, you will be naming the device `/dev/md0` and include the disks that will build the array:

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/ Copy /de
```

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 afterward.

For instance, to create an array that has three copies in the offset layout, the command would include the following:

```
$ sudo mdadm --create --verbose /dev/md0 --level=10 --layout=o3 --raid-d Copy =4
```

The `mdadm` tool will start to configure the array. It 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 sp

unused devices: <none>
```

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 (two copies in the near configuration). The third highlighted area shows the progress on the build. You can continue the guide while this process completes.

Creating and Mounting 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 with the following:

```
$ sudo mount /dev/md0 /mnt/md0
```

Check whether the new space is available:

```
$ df -h -x devtmpfs -x tmpfs
```

```
Output
Filesystem      Size  Used Avail Use% Mounted on
/dev/vda1        25G  1.4G   23G   6% /
/dev/vda15      105M  3.4M  102M   4% /boot/efi
 /dev/md0        197G   60M  187G   1% /mnt/md0
```

The new filesystem is mounted and accessible.

Saving the Array Layout

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

```
$ sudo mdadm --detail --scan | sudo tee -a /etc/mdadm/mdadm.conf            Copy
```

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                                                   Copy
```

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 / Copy tab
```

Your RAID 10 array will now automatically assemble and mount each boot.

Conclusion

In this guide, you learned 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 can learn how to perform day-to-day management with `mdadm`. Our guide on how to manage RAID arrays with `mdadm` on Ubuntu can help get you started.

Thanks for learning with the DigitalOcean Community. Check out our offerings for compute, storage, networking, and managed databases.

Learn more about us →

About the authors

Justin Ellingwood Author

Still looking for an answer? Ask a question

Search for more help

Was this helpful? Yes No

Comments

6 Comments

Leave a comment...

This textbox defaults to using `Markdown` to format your answer.

You can type `!ref` in this text area to quickly search our full set of tutorials, documentation & marketplace offerings and insert the link!

Sign In or Sign Up to Comment

hexforshort • December 18, 2018

If anyone is having trouble getting the RAID arrays to persist after rebooting, try running the `mdadm` command against a partition instead of the device. Meaning, format the drives as Ext.4 ahead of time and then point it to `/dev/sdb1` and `/dev/sdc1` instead of `/dev/sdb` and `/dev/sdc`.

Show replies   Reply

Thomrou • December 14, 2019

Thank you very much @hexforshort I was indeed also very surprised to lose a few TB after rebooting.

In addition to the previous answer in my case I proceed as follows to fix this issue (RAID 10 with 4x HDD SATA 4TB):

1. First remove all existing partition on every disk (with fdisk or parted)
2. Create partition of maximum size on each disk In my case I used **parted**

```
sudo parted /dev/sda
> mklabel gpt
> print (to get the disk info for the next command)
> mkpart primary 0 1024K 4000GB
> quit
```

No need to update /etc/fstab (this will happen later with the RAID config, in the tutorial) Now disks and partitions should look like this:

```
lsblk -o NAME,SIZE,FSTYPE,TYPE,MOUNTPOINT
NAME          SIZE  FSTYPE   TYPE  MOUNTPOINT
sda           3.7T           disk
└─sda1        3.7T           part
```

```
sdb           3.7T         disk
└─sdb1        3.7T         part
sdc           3.7T         disk
└─sdc1        3.7T         part
sdd           3.7T         disk
└─sdd1        3.7T         part
nvme0n1       477G         disk
├─nvme0n1p1   511M vfat    part /boot/efi
├─nvme0n1p2   476G ext4    part /
├─nvme0n1p3   511M swap    part [SWAP]
└─nvme0n1p4     1M iso9660 part
```

3. Format all the partitions for all the “disks” you want to use

```
sudo mkfs.ext4 /dev/sda1
[...]
```

4. For mdadm create provide each partition (instead of device)

```
sudo mdadm --create --verbose /dev/md0 --level=10 --raid-devices=4 /dev/sda1
```

5. Follow the tutorial At the end of the tutorial you can restart (no matter if the disk are still resyncing, this will resume after reboot). Then check the RAID configuration with the following command.

```
sudo mdadm -D /dev/md0
/dev/md0:
[...]
    Number   Major   Minor   RaidDevice State
       0       8        1        0      active sync set-A   /dev/sda1
       1       8       17        1      active sync set-B   /dev/sdb1
       2       8       33        2      active sync set-A   /dev/sdc1
       3       8       49        3      active sync set-B   /dev/sdd1
```

Hope it helps, Cheers

Show replies Reply

richardmeyer596 • June 6, 2019

WARNING! Following this guide, once you reboot, your data WILL disappear and you’ll have to rebuild the array!

Follow the instructions from @hexforshort. Format the drives as ext4 first, then point mdadm to the partitions instead of the drives.

I lost a alot of time and valuable data because of this problem :(. Whoever wrote this tutorial should update it to prevent this from happening to other people.

Reply

francescotubeadv • April 11, 2019

solved by deleting the partitions with fdisk, or it could be done as hexforshort said

Reply

francescotubeadv • April 5, 2019

I followed all the steps for RAID 1, but after reboot /dev/md0 doesn’t exist anymore…

`update-initramfs -u` gave these warnings:

```
update-initramfs: Generating /boot/initrd.img-4.15.0-47-generic
cryptsetup: WARNING: failed to detect canonical device of /dev/md3
cryptsetup: WARNING: could not determine root device from /etc/fstab
W: Possible missing firmware /lib/firmware/ast_dp501_fw.bin for module ast
I: The initramfs will attempt to resume from /dev/nvme1n1p4
I: (UUID=0e1f0b39-1504-44b9-8e2f-5026d566a325)
I: Set the RESUME variable to override this.
```

Reply

SmoothHacker • February 2, 2019

I'm having difficulties accessing the RAID array in a non-root user. I'm unable to make any modifications to the file system (i.e. make/edit files). Any help would be greatly appreciated!

Reply

This work is licensed under a Creative Commons Attribution-NonCommercial- ShareAlike 4.0 International License.

Try DigitalOcean for free

Click below to sign up and get **$200 of credit** to try our products over 60 days!

Sign up

Popular Topics

Ubuntu

Linux Basics

JavaScript

Python

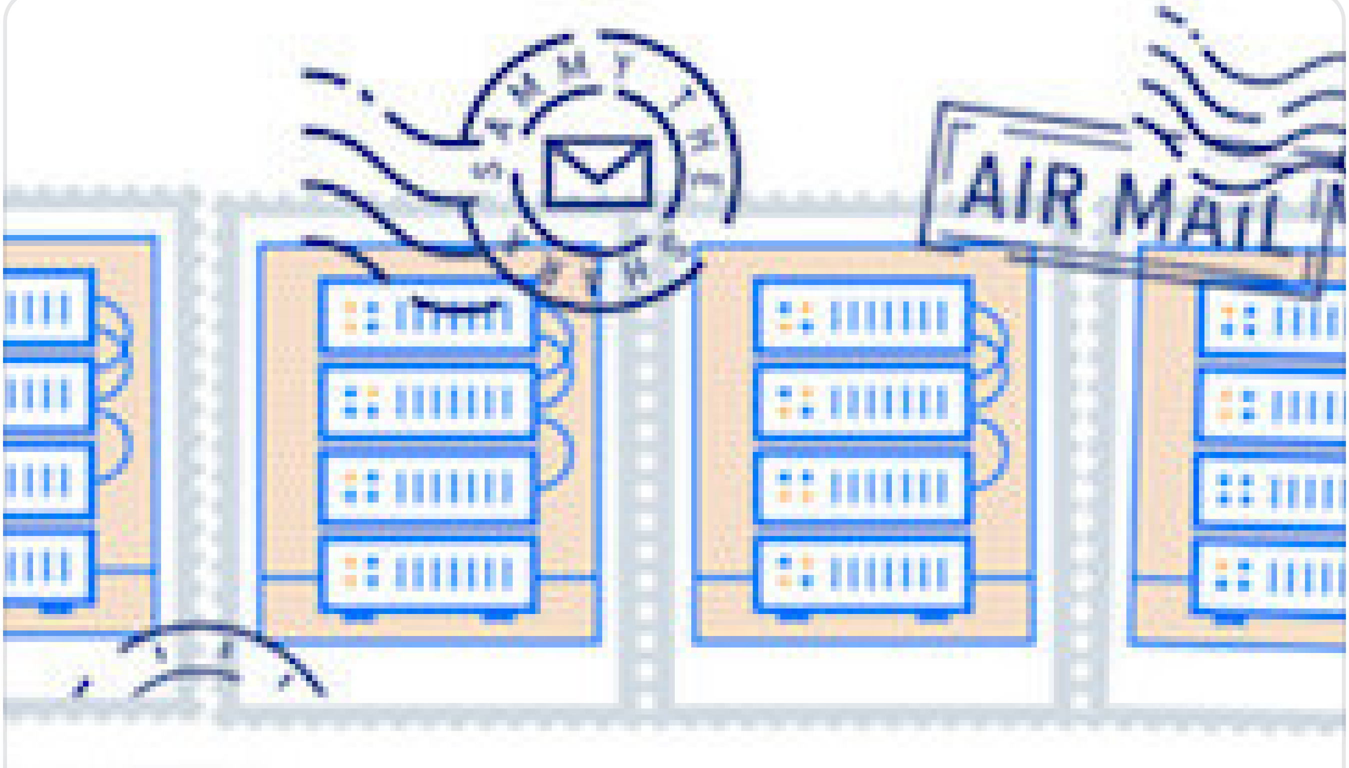
MySQL

Docker

Kubernetes

All tutorials →

Talk to an expert →



Get our biweekly newsletter

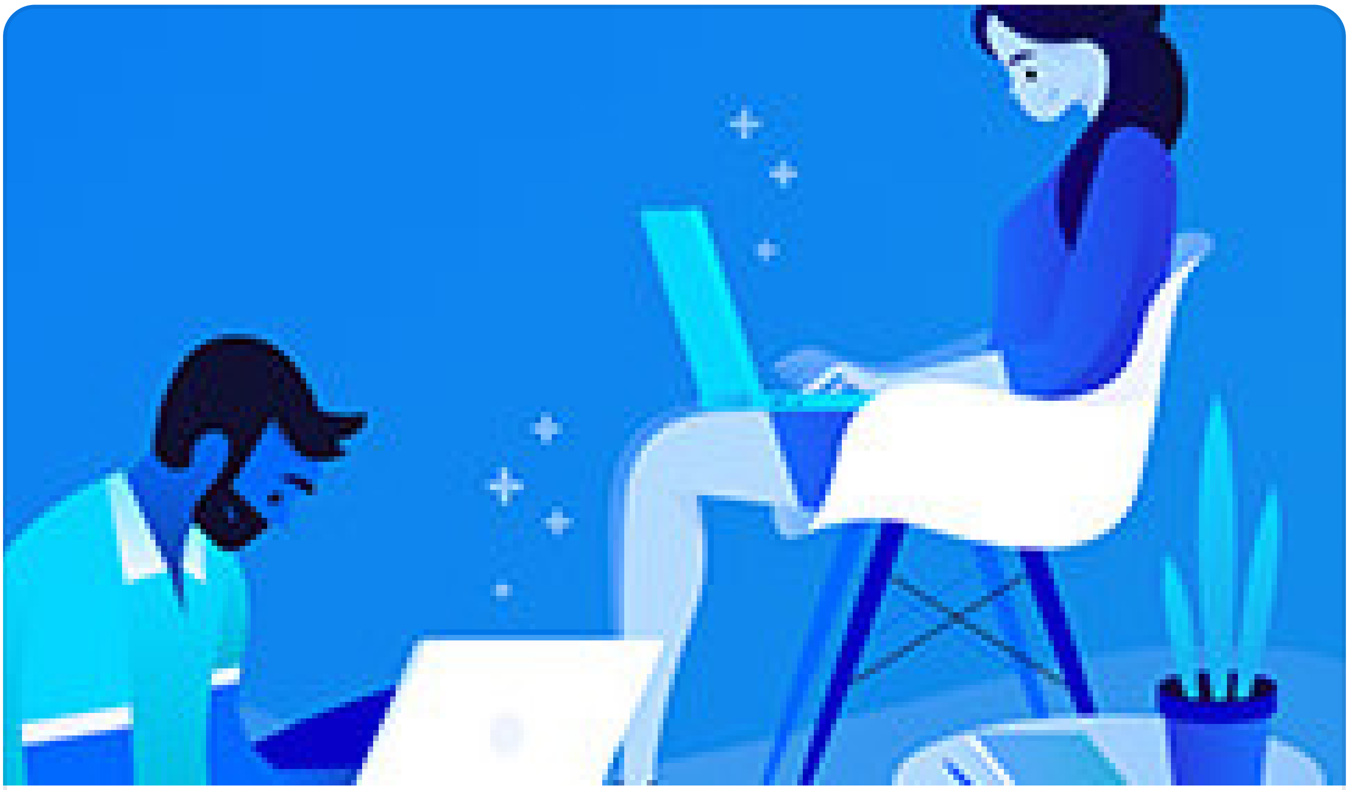
Sign up for Infrastructure as a Newsletter.

Sign up →

Hollie's Hub for Good

Working on improving health and education, reducing inequality, and spurring economic growth? We'd like to help.

Learn more →



Become a contributor

Get paid to write technical tutorials and select a tech-focused charity to receive a matching donation.

Learn more →

Featured Tutorials

Kubernetes Course Learn Python 3 Machine Learning in Python

Getting started with Go Intro to Kubernetes

DigitalOcean Products

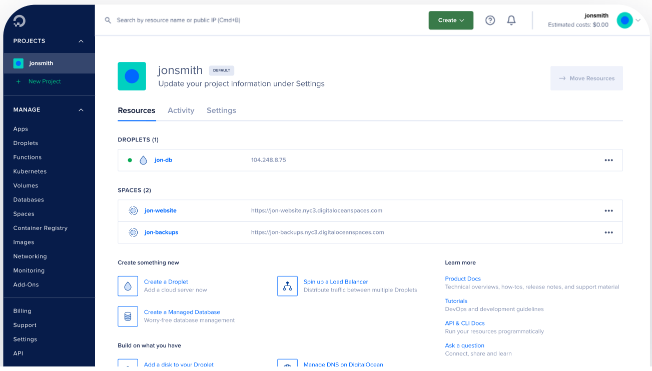
Cloudways Virtual Machines Managed Databases Managed Kubernetes

Block Storage Object Storage Marketplace VPC Load Balancers

Welcome to the developer cloud

DigitalOcean makes it simple to launch in the cloud and scale up as you grow — whether you're running one virtual machine or ten thousand.

Learn more



Get started for free

Sign up and get $200 in credit for your first 60 days with DigitalOcean.

Get started

This promotional offer applies to new accounts only.

Company

Products

Community

Solutions

Contact

© 2024 DigitalOcean, LLC. Sitemap.