Ayub Roti
Technical Solutions Engineer, Serianu Limited.
The Content Herein is automatically generated

How To Remove Docker Containers, Images, Volumes, and Networks | Linuxize

Content Courtesy of
https://linuxize.com/post/how-to-remove-docker-images-containers-volumes-and-networks/
Docker allows you to quickly build, test and deploy applications as portable, self-sufficient containers that can virtually run everywhere.

Docker doesn't remove unused objects such as containers, images, volumes, and networks unless you explicitly tell it to do so. As you work with Docker, you can easily accumulate a large number of unused objects that consume significant disk space and clutter the output produced by the Docker commands.

This guide serves as a "cheat sheet" to help Docker users keep their system organized and to free disk space by removing unused Docker containers, images, volumes, and networks.

Removing All Unused Objects

The `docker system prune` command will remove all stopped containers, all dangling images, and all unused networks:

You'll be prompted to continue, use the `-f` or `--force` flag to bypass the prompt.

```
WARNING! This will remove:
        - all stopped containers
        - all networks not used by at least one container
        - all dangling images
        - all build cache
Are you sure you want to continue? [y/N]
```

If you also want to remove all unused volumes, pass the `--volumes` flag:

```
docker system prune --volumes
```

```
WARNING! This will remove:
        - all stopped containers
        - all networks not used by at least one container
        - all volumes not used by at least one container
        - all dangling images
        - all build cache
Are you sure you want to continue? [y/N] y
```

Removing Docker Containers

Docker containers are not automatically removed when you stop them unless you start the container using the `--rm` flag.

Remove one or more containers

To remove one or more Docker images use the `docker container rm` command followed by the ID of the containers you want to remove.

You can get a list of all active and inactive containers by passing the `-a` flag to the `docker container ls` command:

The output should look something like this:

```
CONTAINER ID        IMAGE                  COMMAND                  CREATED             STATUS
PORTS               NAMES
cc3f2ff51cab        centos                 "/bin/bash"              2 months ago        Created
competent_nightingale
cd20b396a061        solita/ubuntu-systemd   "/bin/bash -c 'exec …"   2 months ago        Exited (137) 2 months a
go                       systemd
fb62432cf3c1        ubuntu                 "/bin/bash"              3 months ago        Exited (130) 3 months a
go                       jolly_mirzakhani
```

Once you know the `CONTAINER ID` of the containers you want to delete, pass it to the `docker container rm` command. For example to remove the first two containers listed in the output above run:

```
docker container rm cc3f2ff51cab cd20b396a061
```

If you get an error similar to the following, it means that the container is running. You'll need to stop the container before removing it.

```
Error response from daemon: You cannot remove a running container fc983ebf4771d42a8bd0029df061cb74dc12cb174530b2
036987575b83442b47. Stop the container before attempting removal or force remove.
```

Remove all stopped containers

Before performing the removal command, you can get a list of all non-running (stopped) containers that will be removed using the following command:

```
docker container ls -a --filter status=exited --filter status=created
```

To remove all stopped containers use the `docker container prune` command:

You'll be prompted to continue, use the `-f` or `--force` flag to bypass the prompt.

```
WARNING! This will remove all stopped containers.
Are you sure you want to continue? [y/N] y
```

Remove containers using filters

The `docker container prune` command allows you to remove containers based on condition using the filtering flag `--filter`.

At the time of the writing of this article, the currently supported filters (https://docs.docker.com/engine/reference/commandline/container_prune) are `until` and `label`. You can use more than one filter

by passing multiple `--filter` flags.

For example to remove all images that are created more than 12 hours ago, run:

```
docker container prune --filter "until=12h"
```

Stop and remove all containers

You can get a list of all Docker containers on your system using the `docker container ls -aq` command.

To stop all running containers use the `docker container stop` command followed by a list of all containers IDs.

```
docker container stop $(docker container ls -aq)
```

Once all containers are stopped you can remove them using the `docker container stop` command followed by the containers ID list.

```
docker container rm $(docker container ls -aq)
```

Removing Docker Images

Remove one or more images

To remove one or more Docker images use the `docker images ls` command to find the ID of the images you want to remove.

The output should look something like this:

```
REPOSITORY              TAG                 IMAGE ID            CREATED             SIZE
centos                  latest              75835a67d134        7 days ago          200MB
ubuntu                  latest              2a4cca5ac898        2 months ago        111MB
linuxize/fedora         latest              a45d6dca3361        3 months ago        311MB
java                    8-jre               e44d62cf8862        3 months ago        311MB
```

Once you've located the images you want to remove, pass their `IMAGE ID` to the `docker image rm` command. For example to remove the first two images listed in the output above run:

```
docker image rm 75835a67d134 2a4cca5ac898
```

If you get an error similar to the following, it means that the image is used by an existing container. To remove the image you will have to remove the container first.

```
Error response from daemon: conflict: unable to remove repository reference "centos" (must force) - container cd2
0b396a061 is using its referenced image 75835a67d134
```

Remove dangling images

Docker provides a `docker image prune` command that can be used to remove dangled and unused

images.

A dangling image is an image that is not tagged and is not used by any container. To remove dangling images type:

You'll be prompted to continue, use the `-f` or `--force` flag to bypass the prompt.

```
WARNING! This will remove all dangling images.
Are you sure you want to continue? [y/N] y
```

When removing dangling images, if the images build by you are not tagged, they will be removed too.

Remove all unused images

To remove all images which are not referenced by any existing container, not just the dangling ones, use the `prune` command with the `-a` flag:

```
WARNING! This will remove all images without at least one container associated to them.
Are you sure you want to continue? [y/N] y
```

Remove images using filters

With the `docker image prune` command, you can also remove images based on a certain condition using the filtering flag `--filter`.

At the time of the writing of this article, the currently supported filters (https://docs.docker.com/engine/reference/commandline/image_prune/#examples) are `until` and `label`. You can use more than one filter by passing multiple `--filter` flags.

For example to remove all images that are created more than 12 hours ago, run:

```
docker image prune -a --filter "until=12h"
```

Removing Docker Volumes

Remove one or more volumes

To remove one or more Docker volumes use the `docker volume ls` command to find the ID of the volumes you want to remove.

The output should look something like this:

```
DRIVER              VOLUME NAME
local               4e12af8913af888ba67243dec78419bf18adddc3c7a4b2345754b6db64293163
local               terano
```

Once you've found the `VOLUME NAME` of the volumes you want to remove, pass them to the `docker volume rm` command. For example to remove the first volume listed in the output above run:

```
docker volume rm 4e12af8913af888ba67243dec78419bf18adddc3c7a4b2345754b6db64293163
```

If you get an error similar to the following, it means that the volume is used by an existing container. To remove the volume you will have to remove the container first.

```
Error response from daemon: remove 4e12af8913af888ba67243dec78419bf18adddc3c7a4b2345754b6db64293163: volume is i
n use - [c7188935a38a6c3f9f11297f8c98ce9996ef5ddad6e6187be62bad3001a66c8e]
```

Remove all unused volumes

To remove all unused volumes use the `docker image prune` command:

You'll be prompted to continue, use the `-f` or `--force` flag to bypass the prompt.

```
WARNING! This will remove all local volumes not used by at least one container.
Are you sure you want to continue? [y/N]
```

Removing Docker Networks

Remove one or more networks

To remove one or more Docker networks use the `docker network ls` command to find the ID of the networks you want to remove.

The output should look something like this:

```
NETWORK ID          NAME                DRIVER              SCOPE
107b8ac977e3        bridge              bridge              local
ab998267377d        host                host                local
c520032c3d31        my-bridge-network   bridge              local
9bc81b63f740        none                null                local
```

Once you've located the networks you want to remove, pass their `NETWORK ID` to the `docker network rm` command. For example to remove the network with the name `my-bridge-network` run:

```
docker network rm c520032c3d31
```

If you get an error similar to the following, it means that the network is used by an existing container. To remove the network you will have to remove the container first.

```
Error response from daemon: network my-bridge-network id 6f5293268bb91ad2498b38b0bca970083af87237784017be24ea208
d2233c5aa has active endpoints
```

Remove all unused network

Use the `docker network prune` command to remove all unused networks.

You'll be prompted to continue, use the `-f` or `--force` flag to bypass the prompt.

```
WARNING! This will remove all networks not used by at least one container.
Are you sure you want to continue? [y/N]
```

Remove networks using filters

With the `docker network prune` command you can remove networks based on condition using the filtering flag `--filter`.

At the time of the writing of this article, the currently supported filters (https://docs.docker.com/engine/reference/commandline/network_prune/) are `until` and `label`. You can use more than one filter by passing multiple `--filter` flags.

For example to remove all networks that are created more than 12 hours ago, run:

```
docker network prune -a --filter "until=12h"
```

Conclusion

In this guide, we have shown you some of the common commands for removing Docker containers, images, volumes, and networks.

You should also check out the official Docker documentation (https://docs.docker.com/).

If you have any question, please leave a comment below.

docker rm | Docker Documentation

Content Courtesy of https://docs.docker.com/engine/reference/commandline/rm/

docker rm

Estimated reading time: 2 minutes

Description

Remove one or more containers

Usage

```
docker rm [OPTIONS] CONTAINER [CONTAINER...]
```

Options

Name, shorthand	Default	Description
`--force , -f`		Force the removal of a running container (uses SIGKILL)
`--link , -l`		Remove the specified link
`--volumes , -v`		Remove the volumes associated with the container

Parent command

Command	Description
docker (/engine/reference/commandline/docker)	The base command for the Docker CLI.

Examples

Remove a container

This will remove the container referenced under the link `/redis`.

```
$ docker rm /redis

/redis
```

Remove a link specified with `--link` on the default bridge network

This will remove the underlying link between `/webapp` and the `/redis` containers on the default bridge network, removing all network communication between the two containers. This does not apply when `--link` is used with user-specified networks.

```
$ docker rm --link /webapp/redis

/webapp/redis
```

Force-remove a running container

This command will force-remove a running container.

```
$ docker rm --force redis

redis
```

The main process inside the container referenced under the link `redis` will receive `SIGKILL`, then the container will be removed.

Remove all stopped containers

```
$ docker rm $(docker ps -a -q)
```

This command will delete all stopped containers. The command `docker ps -a -q` will return all existing container IDs and pass them to the `rm` command which will delete them. Any running containers will not be deleted.

Remove a container and its volumes

```
$ docker rm -v redis
redis
```

This command will remove the container and any volumes associated with it. Note that if a volume was specified with a name, it will not be removed.

Remove a container and selectively remove volumes

```
$ docker create -v awesome:/foo -v /bar --name hello redis
hello
$ docker rm -v hello
```

In this example, the volume for `/foo` will remain intact, but the volume for `/bar` will be removed. The same behavior holds for volumes inherited with `--volumes-from`.

References

https://linuxize.com/post/how-to-remove-docker-images-containers-volumes-and-networks/
https://docs.docker.com/engine/reference/commandline/rm/