# Use the Docker command line

To list available commands, either run docker with no parameters or execute docker help:

$ docker

Usage: docker [OPTIONS] COMMAND [arg...]

docker [ --help | -v | --version ]

A self-sufficient runtime for containers.

Options:

--config=~/.docker Location of client config files

-D, --debug Enable debug mode

-H, --host=[] Daemon socket(s) to connect to

-h, --help Print usage

-l, --log-level=info Set the logging level

--tls Use TLS; implied by --tlsverify

--tlscacert=~/.docker/ca.pem Trust certs signed only by this CA

--tlscert=~/.docker/cert.pem Path to TLS certificate file

--tlskey=~/.docker/key.pem Path to TLS key file

--tlsverify Use TLS and verify the remote

-v, --version Print version information and quit

Commands:

attach Attach to a running container

# […]

Depending on your Docker system configuration, you may be required to preface each docker command with sudo. To avoid having to use sudo with the docker command, your system administrator can create a Unix group called docker and add users to it.

For more information about installing Docker or sudo configuration, refer to the [installation](https://docs.docker.com/engine/installation/) instructions for your operating system.

## Environment variables

For easy reference, the following list of environment variables are supported by the docker command line:

* DOCKER\_API\_VERSION The API version to use (e.g. 1.19)
* DOCKER\_CONFIG The location of your client configuration files.
* DOCKER\_CERT\_PATH The location of your authentication keys.
* DOCKER\_DRIVER The graph driver to use.
* DOCKER\_HOST Daemon socket to connect to.
* DOCKER\_NOWARN\_KERNEL\_VERSION Prevent warnings that your Linux kernel is unsuitable for Docker.
* DOCKER\_RAMDISK If set this will disable ‘pivot\_root’.
* DOCKER\_TLS\_VERIFY When set Docker uses TLS and verifies the remote.
* DOCKER\_CONTENT\_TRUST When set Docker uses notary to sign and verify images. Equates to --disable-content-trust=false for build, create, pull, push, run.
* DOCKER\_CONTENT\_TRUST\_SERVER The URL of the Notary server to use. This defaults to the same URL as the registry.
* DOCKER\_TMPDIR Location for temporary Docker files.

Because Docker is developed using ‘Go’, you can also use any environment variables used by the ‘Go’ runtime. In particular, you may find these useful:

* HTTP\_PROXY
* HTTPS\_PROXY
* NO\_PROXY

These Go environment variables are case-insensitive. See the [Go specification](http://golang.org/pkg/net/http/) for details on these variables.

## Configuration files

By default, the Docker command line stores its configuration files in a directory called .docker within your $HOME directory. However, you can specify a different location via the DOCKER\_CONFIG environment variable or the --config command line option. If both are specified, then the --config option overrides the DOCKER\_CONFIG environment variable. For example:

docker --config ~/testconfigs/ ps

Instructs Docker to use the configuration files in your ~/testconfigs/ directory when running the ps command.

Docker manages most of the files in the configuration directory and you should not modify them. However, you can modify the config.json file to control certain aspects of how the docker command behaves.

Currently, you can modify the docker command behavior using environment variables or command-line options. You can also use options within config.json to modify some of the same behavior. When using these mechanisms, you must keep in mind the order of precedence among them. Command line options override environment variables and environment variables override properties you specify in a config.json file.

The config.json file stores a JSON encoding of several properties:

The property HttpHeaders specifies a set of headers to include in all messages sent from the Docker client to the daemon. Docker does not try to interpret or understand these header; it simply puts them into the messages. Docker does not allow these headers to change any headers it sets for itself.

The property psFormat specifies the default format for docker ps output. When the --format flag is not provided with the docker ps command, Docker’s client uses this property. If this property is not set, the client falls back to the default table format. For a list of supported formatting directives, see the [Formatting section in the docker ps documentation](https://docs.docker.com/engine/reference/commandline/ps/)

Once attached to a container, users detach from it and leave it running using the using CTRL-p CTRL-q key sequence. This detach key sequence is customizable using the detachKeys property. Specify a <sequence> value for the property. The format of the <sequence> is a comma-separated list of either a letter [a-Z], or the ctrl- combined with any of the following:

* a-z (a single lowercase alpha character )
* @ (at sign)
* [ (left bracket)
* \\ (two backward slashes)
* \_ (underscore)
* ^ (caret)

Your customization applies to all containers started in with your Docker client. Users can override your custom or the default key sequence on a per-container basis. To do this, the user specifies the --detach-keys flag with the docker attach, docker exec, docker run or docker start command.

The property imagesFormat specifies the default format for docker images output. When the --format flag is not provided with the docker images command, Docker’s client uses this property. If this property is not set, the client falls back to the default table format. For a list of supported formatting directives, see the [Formatting section in the docker images documentation](https://docs.docker.com/engine/reference/commandline/images/)

Following is a sample config.json file:

{

"HttpHeaders": {

"MyHeader": "MyValue"

},

"psFormat": "table {{.ID}}\\t{{.Image}}\\t{{.Command}}\\t{{.Labels}}",

"imagesFormat": "table {{.ID}}\\t{{.Repository}}\\t{{.Tag}}\\t{{.CreatedAt}}",

"detachKeys": "ctrl-e,e"

}

### Notary

If using your own notary server and a self-signed certificate or an internal Certificate Authority, you need to place the certificate at tls/<registry\_url>/ca.crt in your docker config directory.

Alternatively you can trust the certificate globally by adding it to your system’s list of root Certificate Authorities.

## Help

To list the help on any command just execute the command, followed by the --help option.

$ docker run --help

Usage: docker run [OPTIONS] IMAGE [COMMAND] [ARG...]

Run a command in a new container

-a, --attach=[] Attach to STDIN, STDOUT or STDERR

--cpu-shares=0 CPU shares (relative weight)

...

## Option types

Single character command line options can be combined, so rather than typing docker run -i -t --name test busybox sh, you can write docker run -it --name test busybox sh.

### Boolean

Boolean options take the form -d=false. The value you see in the help text is the default value which is set if you do not specify that flag. If you specify a Boolean flag without a value, this will set the flag to true, irrespective of the default value.

For example, running docker run -d will set the value to true, so your container will run in “detached” mode, in the background.

Options which default to true (e.g., docker build --rm=true) can only be set to the non-default value by explicitly setting them to false:

$ docker build --rm=false .

### Multi

You can specify options like -a=[] multiple times in a single command line, for example in these commands:

$ docker run -a stdin -a stdout -i -t ubuntu /bin/bash

$ docker run -a stdin -a stdout -a stderr ubuntu /bin/ls

Sometimes, multiple options can call for a more complex value string as for -v:

$ docker run -v /host:/container example/mysql

Note: Do not use the -t and -a stderr options together due to limitations in the pty implementation. All stderr in pty mode simply goes to stdout.

### Strings and Integers

Options like --name="" expect a string, and they can only be specified once. Options like -c=0 expect an integer, and they can only be specified once.

# Ps Command

Usage: docker ps [OPTIONS]

List containers

Options:

-a, --all Show all containers (default shows just running)

-f, --filter value Filter output based on conditions provided (default [])

- exited=<int> an exit code of <int>

- label=<key> or label=<key>=<value>

- status=(created|restarting|running|paused|exited)

- name=<string> a container's name

- id=<ID> a container's ID

- before=(<container-name>|<container-id>)

- since=(<container-name>|<container-id>)

- ancestor=(<image-name>[:tag]|<image-id>|<image@digest>)

containers created from an image or a descendant.

--format string Pretty-print containers using a Go template

--help Print usage

-n, --last int Show n last created containers (includes all states) (default -1)

-l, --latest Show the latest created container (includes all states)

--no-trunc Don't truncate output

-q, --quiet Only display numeric IDs

-s, --size Display total file sizes

Running docker ps --no-trunc showing 2 linked containers.

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

4c01db0b339c ubuntu:12.04 bash 17 seconds ago Up 16 seconds 3300-3310/tcp webapp

d7886598dbe2 crosbymichael/redis:latest /redis-server --dir 33 minutes ago Up 33 minutes 6379/tcp redis,webapp/db

The docker ps command only shows running containers by default. To see all containers, use the -a (or --all) flag:

$ docker ps -a

docker ps groups exposed ports into a single range if possible. E.g., a container that exposes TCP ports 100, 101, 102 displays 100-102/tcp in the PORTS column.

## Filtering

The filtering flag (-f or --filter) format is a key=value pair. If there is more than one filter, then pass multiple flags (e.g. --filter "foo=bar" --filter "bif=baz")

The currently supported filters are:

* id (container’s id)
* label (label=<key> or label=<key>=<value>)
* name (container’s name)
* exited (int - the code of exited containers. Only useful with --all)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| status (created | restarting | running | paused | exited | dead) |

* ancestor (<image-name>[:<tag>], <image id> or <image@digest>) - filters containers that were created from the given image or a descendant.
* before (container’s id or name) - filters containers created before given id or name
* since (container’s id or name) - filters containers created since given id or name

|  |  |  |
| --- | --- | --- |
| isolation (default | process | hyperv) (Windows daemon only) |

* volume (volume name or mount point) - filters containers that mount volumes.
* network (network id or name) - filters containers connected to the provided network

#### Label

The label filter matches containers based on the presence of a label alone or a label and a value.

The following filter matches containers with the color label regardless of its value.

$ docker ps --filter "label=color"

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

673394ef1d4c busybox "top" 47 seconds ago Up 45 seconds nostalgic\_shockley

d85756f57265 busybox "top" 52 seconds ago Up 51 seconds high\_albattani

The following filter matches containers with the color label with the blue value.

$ docker ps --filter "label=color=blue"

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

d85756f57265 busybox "top" About a minute ago Up About a minute high\_albattani

#### Name

The name filter matches on all or part of a container’s name.

The following filter matches all containers with a name containing the nostalgic\_stallman string.

$ docker ps --filter "name=nostalgic\_stallman"

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

9b6247364a03 busybox "top" 2 minutes ago Up 2 minutes nostalgic\_stallman

You can also filter for a substring in a name as this shows:

$ docker ps --filter "name=nostalgic"

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

715ebfcee040 busybox "top" 3 seconds ago Up 1 seconds i\_am\_nostalgic

9b6247364a03 busybox "top" 7 minutes ago Up 7 minutes nostalgic\_stallman

673394ef1d4c busybox "top" 38 minutes ago Up 38 minutes nostalgic\_shockley

#### Exited

The exited filter matches containers by exist status code. For example, to filter for containers that have exited successfully:

$ docker ps -a --filter 'exited=0'

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

ea09c3c82f6e registry:latest /srv/run.sh 2 weeks ago Exited (0) 2 weeks ago 127.0.0.1:5000->5000/tcp desperate\_leakey

106ea823fe4e fedora:latest /bin/sh -c 'bash -l' 2 weeks ago Exited (0) 2 weeks ago determined\_albattani

48ee228c9464 fedora:20 bash 2 weeks ago Exited (0) 2 weeks ago tender\_torvalds

#### Killed containers

You can use a filter to locate containers that exited with status of 137 meaning a SIGKILL(9) killed them.

$ docker ps -a --filter 'exited=137'

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

b3e1c0ed5bfe ubuntu:latest "sleep 1000" 12 seconds ago Exited (137) 5 seconds ago grave\_kowalevski

a2eb5558d669 redis:latest "/entrypoint.sh redi 2 hours ago Exited (137) 2 hours ago sharp\_lalande

Any of these events result in a 137 status:

* the init process of the container is killed manually
* docker kill kills the container
* Docker daemon restarts which kills all running containers

#### Status

The status filter matches containers by status. You can filter using created, restarting, running, paused, exited and dead. For example, to filter for running containers:

$ docker ps --filter status=running

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

715ebfcee040 busybox "top" 16 minutes ago Up 16 minutes i\_am\_nostalgic

d5c976d3c462 busybox "top" 23 minutes ago Up 23 minutes top

9b6247364a03 busybox "top" 24 minutes ago Up 24 minutes nostalgic\_stallman

To filter for paused containers:

$ docker ps --filter status=paused

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

673394ef1d4c busybox "top" About an hour ago Up About an hour (Paused) nostalgic\_shockley

#### Ancestor

The ancestor filter matches containers based on its image or a descendant of it. The filter supports the following image representation:

* image
* image:tag
* image:tag@digest
* short-id
* full-id

If you don’t specify a tag, the latest tag is used. For example, to filter for containers that use the latest ubuntu image:

$ docker ps --filter ancestor=ubuntu

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

919e1179bdb8 ubuntu-c1 "top" About a minute ago Up About a minute admiring\_lovelace

5d1e4a540723 ubuntu-c2 "top" About a minute ago Up About a minute admiring\_sammet

82a598284012 ubuntu "top" 3 minutes ago Up 3 minutes sleepy\_bose

bab2a34ba363 ubuntu "top" 3 minutes ago Up 3 minutes focused\_yonath

Match containers based on the ubuntu-c1 image which, in this case, is a child of ubuntu:

$ docker ps --filter ancestor=ubuntu-c1

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

919e1179bdb8 ubuntu-c1 "top" About a minute ago Up About a minute admiring\_lovelace

Match containers based on the ubuntu version 12.04.5 image:

$ docker ps --filter ancestor=ubuntu:12.04.5

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

82a598284012 ubuntu:12.04.5 "top" 3 minutes ago Up 3 minutes sleepy\_bose

The following matches containers based on the layer d0e008c6cf02 or an image that have this layer in it’s layer stack.

$ docker ps --filter ancestor=d0e008c6cf02

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

82a598284012 ubuntu:12.04.5 "top" 3 minutes ago Up 3 minutes sleepy\_bose

#### Before

The before filter shows only containers created before the container with given id or name. For example, having these containers created:

$ docker ps

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

9c3527ed70ce busybox "top" 14 seconds ago Up 15 seconds desperate\_dubinsky

4aace5031105 busybox "top" 48 seconds ago Up 49 seconds focused\_hamilton

6e63f6ff38b0 busybox "top" About a minute ago Up About a minute distracted\_fermat

Filtering with before would give:

$ docker ps -f before=9c3527ed70ce

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

4aace5031105 busybox "top" About a minute ago Up About a minute focused\_hamilton

6e63f6ff38b0 busybox "top" About a minute ago Up About a minute distracted\_fermat

#### Since

The since filter shows only containers created since the container with given id or name. For example, with the same containers as in before filter:

$ docker ps -f since=6e63f6ff38b0

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

9c3527ed70ce busybox "top" 10 minutes ago Up 10 minutes desperate\_dubinsky

4aace5031105 busybox "top" 10 minutes ago Up 10 minutes focused\_hamilton

#### Volume

The volume filter shows only containers that mount a specific volume or have a volume mounted in a specific path:

$ docker ps --filter volume=remote-volume --format "table {{.ID}}\t{{.Mounts}}"

CONTAINER ID MOUNTS

9c3527ed70ce remote-volume

$ docker ps --filter volume=/data --format "table {{.ID}}\t{{.Mounts}}"

CONTAINER ID MOUNTS

9c3527ed70ce remote-volume

#### Network

The network filter shows only containers that are connected to a network with a given name or id.

The following filter matches all containers that are connected to a network with a name containing net1.

$ docker run -d --net=net1 --name=test1 ubuntu top

$ docker run -d --net=net2 --name=test2 ubuntu top

$ docker ps --filter network=net1

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

9d4893ed80fe ubuntu "top" 10 minutes ago Up 10 minutes test1

The network filter matches on both the network’s name and id. The following example shows all containers that are attached to the net1 network, using the network id as a filter;

$ docker network inspect --format "{{.ID}}" net1

8c0b4110ae930dbe26b258de9bc34a03f98056ed6f27f991d32919bfe401d7c5

$ docker ps --filter network=8c0b4110ae930dbe26b258de9bc34a03f98056ed6f27f991d32919bfe401d7c5

CONTAINER ID IMAGE COMMAND CREATED STATUS PORTS NAMES

9d4893ed80fe ubuntu "top" 10 minutes ago Up 10 minutes test1

## Formatting

The formatting option (--format) pretty-prints container output using a Go template.

Valid placeholders for the Go template are listed below:

| Placeholder | Description |
| --- | --- |
| .ID | Container ID |
| .Image | Image ID |
| .Command | Quoted command |
| .CreatedAt | Time when the container was created. |
| .RunningFor | Elapsed time since the container was started. |
| .Ports | Exposed ports. |
| .Status | Container status. |
| .Size | Container disk size. |
| .Names | Container names. |
| .Labels | All labels assigned to the container. |
| .Label | Value of a specific label for this container. For example '{{.Label "com.docker.swarm.cpu"}}' |
| .Mounts | Names of the volumes mounted in this container. |

When using the --format option, the ps command will either output the data exactly as the template declares or, when using the table directive, includes column headers as well.

The following example uses a template without headers and outputs the ID and Command entries separated by a colon for all running containers:

$ docker ps --format "{{.ID}}: {{.Command}}"

a87ecb4f327c: /bin/sh -c #(nop) MA

01946d9d34d8: /bin/sh -c #(nop) MA

c1d3b0166030: /bin/sh -c yum -y up

41d50ecd2f57: /bin/sh -c #(nop) MA

To list all running containers with their labels in a table format you can use:

$ docker ps --format "table {{.ID}}\t{{.Labels}}"

CONTAINER ID LABELS

a87ecb4f327c com.docker.swarm.node=ubuntu,com.docker.swarm.storage=ssd

01946d9d34d8

c1d3b0166030 com.docker.swarm.node=debian,com.docker.swarm.cpu=6

41d50ecd2f57 com.docker.swarm.node=fedora,com.docker.swarm.cpu=3,com.docker.swarm.storage=ssd

\*\*\*\*\*\*\*\*\*\*