Docker



(Audience Handbook)

## 

Index

* [Docker Commands](#_x15jk2gpw2s1)
  + [Docker version](#_59y4pn8amfex)
  + [Docker top](#_enwpjgqnza02)
  + [Docker ps](#_szm8rl48asz9)
  + [Docker stop](#_zadyydnbdxqm)
  + [Docker start](#_eegv2w32ofz5)
  + [Docker rmi](#_zg3vwas0kx7a)
  + [Docker search](#_60fk682d80k6)
  + [Docker Run](#_lpwg50navgpg)
    - [Detached vs foreground](#_kjzcl2eq5hrl)
    - [Managing /etc/hosts](#_kq6lld1rzjvf)
  + [CMD (default command or options)](#_4t3h5jelrs6s)
  + [ENTRYPOINT (default command to execute at runtime)](#_opamh2xoc7f3)
  + [Docker build](#_88w8wl9cmu32)
  + [Docker commit](#_q4nkd44fsbwj)
  + [Docker cp](#_ytgzdg1zdozl)
  + [Docker exec](#_ipa8hrkeufa3)
  + [Docker images](#_gyq9eh6x97pk)
  + [Docker info](#_omwqpunwsidp)
  + [Docker inspect](#_wujfbv5kkapt)
  + [Docker kill](#_fxackk62639b)
  + [Docker pull](#_r78bwrs4jwsg)
* [Assessment](#_9t9gz66bc1ie)

## 

## 

## 

## 

## 

## 

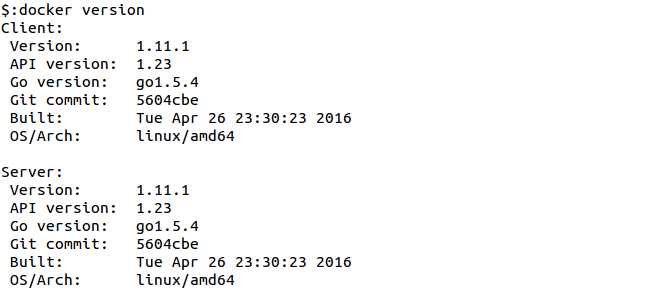
## 

# Docker Commands

## Docker version

It shows the Docker version.

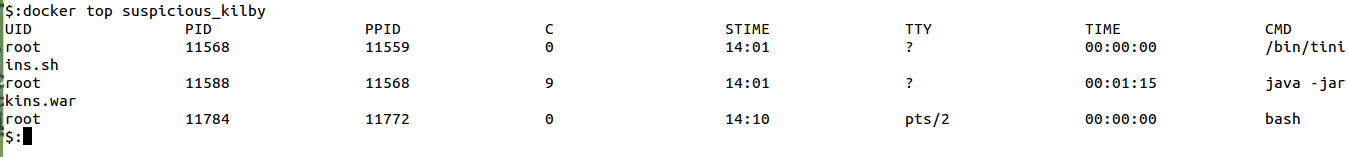
$: docker version



## Docker top

It displays the running processes of a container.

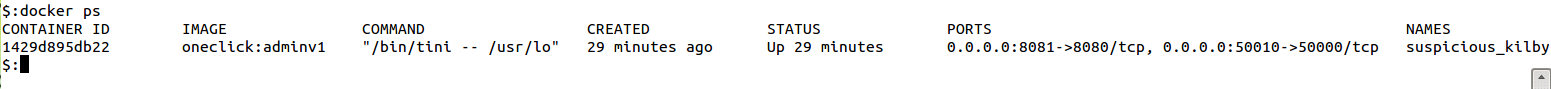
$: docker top suspicious\_kilby



## Docker ps

docker ps will show only running containers by default.

$: docker ps

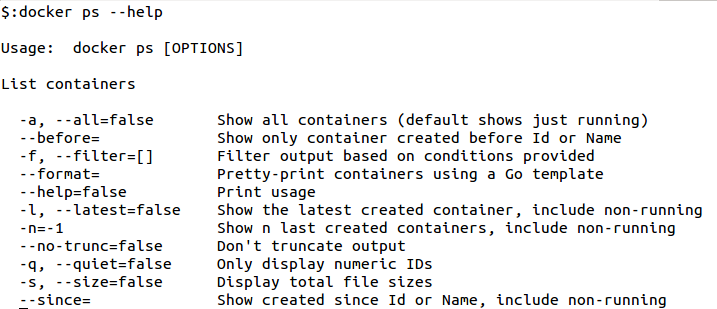


To see all containers:

$: docker ps -a



$: docker ps --help



## 

## 

## 

## 

## Docker stop

Stop one or more running containers. The main process inside the container will receive SIGTERM, and after a grace period, SIGKILL.

$: docker stop <container-name>

Example:

$: docker stop suspicious\_kilby

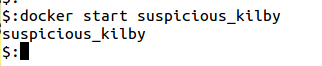
## stop.pngDocker start

Start one or more stopped containers.

$: docker start <container-name>

Example:

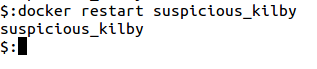
$: docker start suspicious\_kilby



Docker restart

Restart a container

$: docker restart <container-name>



Docker rm

Remove one or more containers

$:docker rm consul

This will remove the container referenced under the link consul.

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

This will remove the underlying link between /webapp and the /redis containers removing all network communication.

$: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.

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

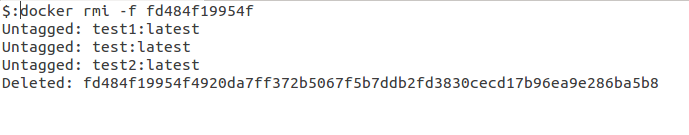
This command will delete all stopped containers. The command docker ps -a -q will return all existing container IDs and pass them to the rm command which will delete them. Any running containers will not be deleted.

## Docker rmi

You can remove an image using its short or long ID, its tag, or its digest. If an image has one or more tag referencing it, you must remove all of them before the image is removed. Digest references are removed automatically when an image is removed by tag.

$:docker rmi test1  
Untagged: test1:latest

If you use the -f flag and specify the image’s short or long ID, then this command untags and removes all images that match the specified ID.



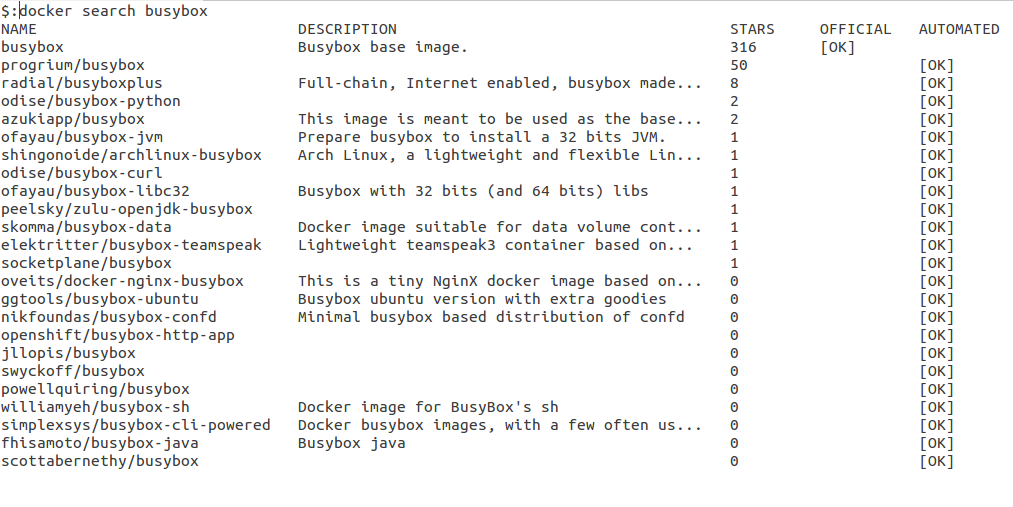
## 

## Docker search

Search the Docker Hub for images.

This example displays images with a name containing ‘busybox’:

$:docker search busybox



## Docker Run

Docker runs processes in isolated containers. When an operator executes docker run, she starts a process with its own file system, its own networking, and its own isolated process tree. The Image which starts the process may define defaults related to the binary to run, the networking to expose, and more, but docker run gives final control to the operator who starts the container from the image. That's the main reasonrun has more options than any other docker command.

The basic docker run command takes this form:

$:docker run [OPTIONS] IMAGE[:TAG] [COMMAND] [ARG...]

The list of [OPTIONS] breaks down into two groups:

1. Settings exclusive to operators, including:
   * Detached or Foreground running,
   * Container Identification,
   * Network settings, and
   * Runtime Constraints on CPU and Memory
   * Privileges and LXC Configuration
2. Settings shared between operators and developers, where operators can override defaults developers set in images at build time.

Together, the docker run [OPTIONS] give the operator complete control over runtime behavior, allowing them to override all defaults set by the developer during docker build and nearly all the defaults set by the Docker runtime itself.

### Detached vs foreground

When starting a Docker container, you must first decide if you want to run the container in the background in a "detached" mode or in the default foreground mode:

-d=false: Detached mode: Run container in the background, print new container id

In foreground mode (the default when -d is not specified), docker run can start the process in the container and attach the console to the process's standard input, output, and standard error. It can even pretend to be a TTY (this is what most command line executables expect) and pass along signals.

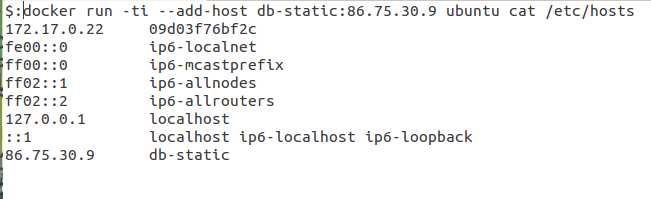


For interactive processes (like a shell), you must use -i -t together in order to allocate a tty for the container process. Specifying -t is however forbidden when the client standard output is redirected or pipe, such as in: echo test | docker run -i busybox cat.

### Managing /etc/hosts

Your container will have lines in /etc/hosts which define the hostname of the container itself as well as localhost and a few other common things. The --add-host flag can be used to add additional lines to /etc/hosts.

$:docker run -ti --add-host db-static:86.75.30.9 ubuntu cat /etc/hosts



## CMD (default command or options)

This command is optional because the person who created the IMAGE may have already provided a default COMMAND using the Dockerfile CMD instruction. As the operator (the person running a container from the image), you can override that CMD instruction just by specifying a new COMMAND.

If the image also specifies an ENTRYPOINT then the CMD or COMMAND get appended as arguments to the ENTRYPOINT.

## ENTRYPOINT (default command to execute at runtime)

The ENTRYPOINT of an image is similar to a COMMAND because it specifies what executable to run when the container starts, but it is (purposely) more difficult to override. The ENTRYPOINT gives a container its default nature or behavior, so that when you set an ENTRYPOINT you can run the container *as if it were that binary*, complete with default options, and you can pass in more options via the COMMAND. But, sometimes an operator may want to run something else inside the container, so you can override the default ENTRYPOINT at runtime by using a string to specify the new ENTRYPOINT. Here is an example of how to run a shell in a container that has been set up to automatically run something else (like /usr/bin/redis-server):

$:docker run -i -t --entrypoint /bin/bash example/redis

Examples of how to pass more parameters to that ENTRYPOINT:

$:docker run -i -t --entrypoint /bin/bash example/redis -c ls -l  
$:docker run -i -t --entrypoint /usr/bin/redis-cli example/redis --help

## 

## 

## 

## 

## 

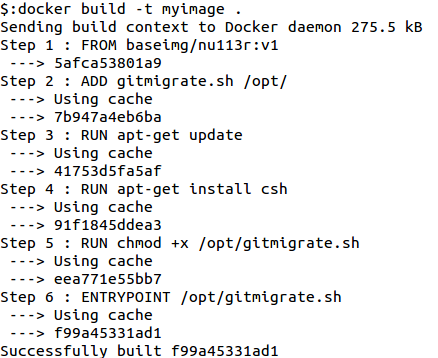
## 

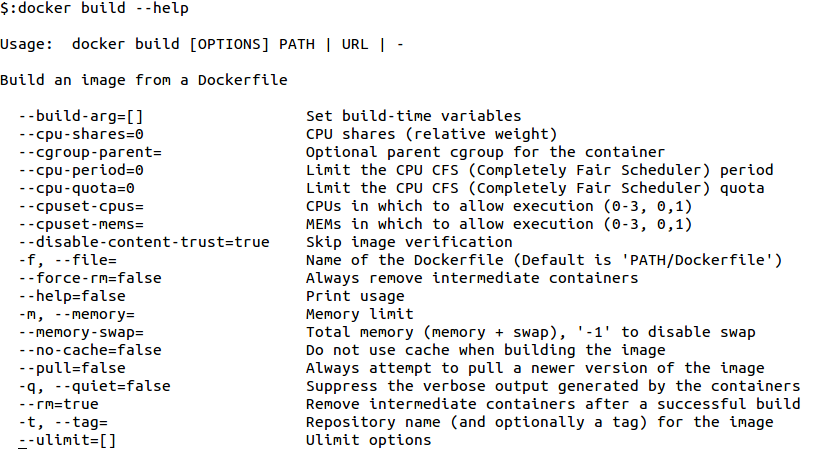
## 

## Docker build

Builds Docker images from a Dockerfile and a “context”. A build’s context is the files located in the specified PATH or URL. The build process can refer to any of the files in the context. For example, your build can use an ADD instruction to reference a file in the context.

$: docker build -t myimage .



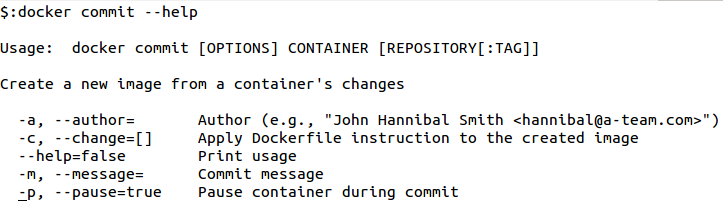


## Docker commit

It can be useful to commit a container’s file changes or settings into a new image. This allows you debug a container by running an interactive shell, or to export a working dataset to another server. Generally, it is better to use Dockerfiles to manage your images in a documented and maintainable way.

$: docker commit 7e5617021 myimage

commitex.png



## 

## 

## 

## 

## 

## 

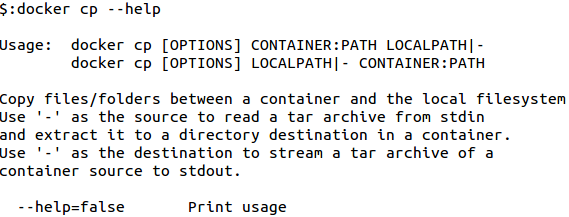
## 

## 

## Docker cp

The docker cp utility copies the contents of SRC\_PATH to the DEST\_PATH. You can copy from the container’s file system to the local machine or the reverse, from the local filesystem to the container. If - is specified for either the SRC\_PATH or DEST\_PATH, you can also stream a tar archive from STDIN or to STDOUT. The CONTAINER can be a running or stopped container. The SRC\_PATH or DEST\_PATH can be a file or directory.

$: docker cp myfile mycontainer:/home/



## 

## Docker exec

The docker exec command runs a new command in a running container.

The command started using docker exec only runs while the container’s primary process (PID 1) is running, and it is not restarted if the container is restarted.

If the container is paused, then the docker exec command will fail with an error.

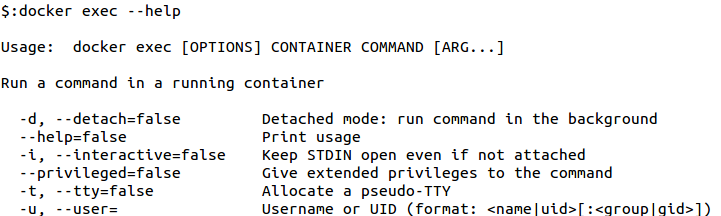
$: docker exec -d mycontainer touch /tmp/myfile

This will create a new file /tmp/myfile inside the running container mycontainer, in the background.

$: docker exec -it mycontainer bash

This will create a new Bash session in the container mycontainer.

$: docker exec --help

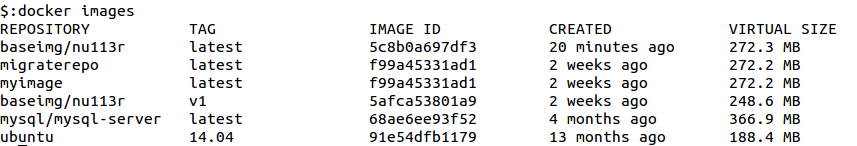


## 

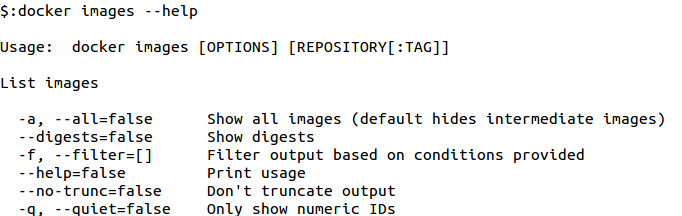
## Docker images

The default docker images will show all top level images, their repository and tags, and their size.

$: docker images



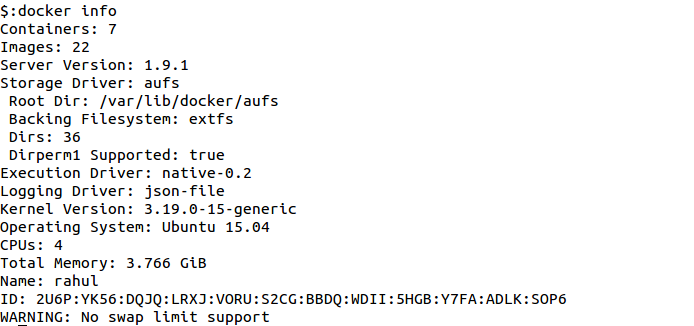
$: docker images --help



## Docker info

This command displays system wide information regarding the Docker installation. Information displayed includes the kernel version, number of containers and images. The number of images shown is the number of unique images. The same image tagged under different names is counted only once.

$: docker info



## 

## 

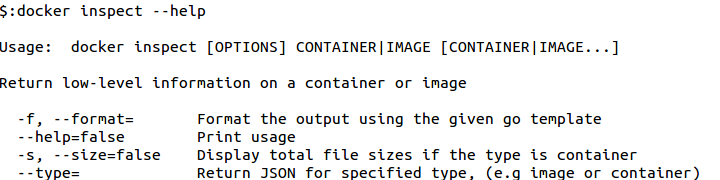
## 

## 

## Docker inspect

By default, this will render all results in a JSON array. If the container and image have the same name, this will return container JSON for unspecified type. If a format is specified, the given template will be executed for each result.

$: docker inspect --help



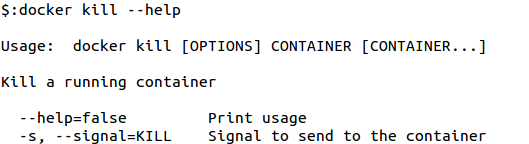
## Docker kill

Kill one or more running containers, the main process inside the container will be sent SIGKILL, or any signal specified with option --signal.

$: docker kill mycontainer



$: docker kill --help

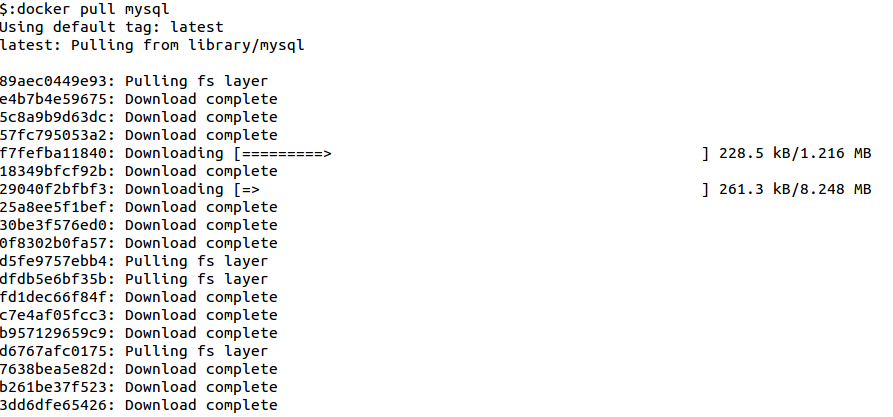


## Docker pull

Most of your images will be created on top of a base image from the Docker Hub registry.

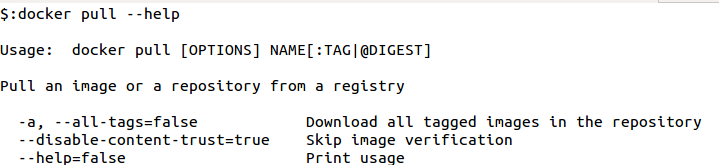
Docker Hub contains many pre-built images that you can pull and try without needing to define and configure your own.

To download a particular image, or set of images (i.e., a repository), use docker pull.



$: docker pull mysql

$: docker pull --help



# Assessment

1. Create a docker container for mongodb.
2. Install nginx on docker and serve two virtual hosts.