15 Docker Commands You Should Know

Part 5 of Learn Enough Docker to be Useful

Jeff Hale

Feb 5, 2019 · 10 min read ★

In this article we'll look at 15 Docker CLI commands you should know. If you haven't yet, check out the rest of this series on Docker concepts, the ecosystem, Dockerfiles, and keeping your images slim. In Part 6 we'll explore data with Docker. I've got a series on Kubernetes in the works too, so follow me to make sure you don't miss the fun!

There are about a billion Docker commands (give or take a billion). The Docker docs are extensive, but overwhelming when you're just getting started. In this article I'll highlight the key commands for running vanilla Docker.

Fruit theme

At risk of taking the food metaphor thread running through these articles too far, let's use a fruit theme. Veggies provided sustenance in the article on slimming down our images. Now tasty fruits will give us nutrients as we learn our key Docker commands.

Overview

Recall that a Docker image is made of a Dockerfile + any necessary dependencies. Also recall that a Docker container is a Docker image brought to life. To work with Docker commands, you first need to know whether you're dealing with an image or a container.

- **A Docker image either exists or it doesn't.**
- **A Docker container either exists or it doesn't.**
- **A Docker container that exists is either running or it isn't.**

Once you know what you're working with you can find the right command for the job.

Commmand Commonalities

Here are a few things to know about Docker commands:

- Docker CLI management commands start with `docker`, then a space, then the management category, then a space, and then the command. For example, `docker container stop` stops a container.
- A command referring to a specific container or image requires the name or id of that container or image.

For example, `docker container run my_app` is the command to build and run the container named *my_app*. I'll use the name `my_container` to refer to a generic container throughout the examples. Same goes for `my_image`, `my_tag`, etc.

I'll provide the command alone and then with common flags, if applicable. A flag with two dashes in front is the full name of the flag. A flag with one dash is a shortcut for the full flag name. For example, `-p` is short for the `--port` flag.

Flags provide options to commands

The goal is to help these commands and flags stick in your memory and for this guide to serve as a reference. This guide is current for Linux and Docker Engine Version 18.09.1 and API version 1.39.

First, we'll look at commands for containers and then we'll look at commands for images. Volumes will be covered in the next article. Here's the list of 15 commands to know — plus 3 bonus commands!

Containers

Use `docker container my_command`

`create` — Create a container from an image.
`start` — Start an existing container.
`run` — Create a new container and start it.
`ls` — List running containers.
`inspect` — See lots of info about a container.
`logs` — Print logs.
`stop` — Gracefully stop running container.
`kill` —Stop main process in container abruptly.
`rm` — Delete a stopped container.

Images

Use `docker image my_command`

`build` — Build an image.

`push` — Push an image to a remote registry.

`ls` — List images.

`history` — See intermediate image info.

`inspect` — See lots of info about an image, including the layers.

`rm` — Delete an image.

Misc

`docker version` — List info about your Docker Client and Server versions.

`docker login` — Log in to a Docker registry.

`docker system prune` — Delete all unused containers, unused networks, and dangling images.

Containers

Container Beginnings

The terms create, start, and run all have similar semantics in everyday life. But each is a separate Docker command that creates and/or starts a container. Let's look at creating a container first.

`docker container create my_repo/my_image:my_tag` — Create a container from an image.

I'll shorten `my_repo/my_image:my_tag` to `my_image` for the rest of the article.

There are a lot of possible flags you could pass to `create`.

```
docker container create -a STDIN my_image
```

`-a` is short for `--attach`. Attach the container to STDIN, STDOUT or STDERR.

Now that we've created a container let's start it.

`docker container start my_container` — Start an existing container.

Note that the container can be referred to by either the container's ID or the container's name.

```
docker container start my_container
```

Start

Now that you know how to create and start a container, let's turn to what's probably the most common Docker command. It combines both `create` and `start` into one command: `run`.

`docker container run my_image` — Create a new container and start it. It also has a lot of options. Let's look at a few.

```
docker container run -i -t -p 1000:8000 --rm my_image
```

`-i` is short for `--interactive`. Keep STDIN open even if unattached.

`-t` is short for `--tty`. Allocates a pseudo terminal that connects your terminal with the container's STDIN and STDOUT.

You need to specify both `-i` and `-t` to then interact with the container through your terminal shell.

`-p` is short for `--port`. The port is the interface with the outside world. `1000:8000` maps the Docker port 8000 to port 1000 on your machine. If you had an app that output something to the browser you could then navigate your browser to `localhost:1000` and see it.

`--rm` Automatically delete the container when it stops running.

Let's look at some more examples of `run`.

```
docker container run -it my_image my_command
```

`sh` is a command you could specify at run time. `sh` will start a shell session inside your container that you can interact with through your terminal. `sh` is preferable to `bash` for Alpine images because Alpine images don't come with `bash` installed. Type `exit` to end the interactive shell session.

Notice that we combined `-i` and `-t` into `-it`.

```
docker container run -d my_image
```

`-d` is short for `--detach`. Run the container in the background. Allows you to use the terminal for other commands while your container runs.

Checking Container Status

If you have running Docker containers and want to find out which one to interact with, then you need to list them.

`docker container ls` — List running containers. Also provides useful information about the containers.

```
docker container ls -a -s
```

`-a` is short for `-all`. List all containers (not just running ones).

`-s` is short for `--size`. List the size for each container.

`docker container inspect my_container` — See lots of info about a container.

`docker container logs my_container` — Print a container's logs.

Logs. Not sure how virtual logs are related. Maybe via reams of paper?

Container Endings

Sometimes you need to stop a running container.

`docker container stop my_container` — Stop one or more running containers gracefully. Gives a default of 10 seconds before container shutdown to finish any processes.

Or if you are impatient:

`docker container kill my_container` — Stop one or more running containers abruptly. It's like pulling the plug on the TV. Prefer `stop` in most situations.

`docker container kill $(docker ps -q)` — Kill all running containers.

docker kill cockroach

Then you delete the container with:

`docker container rm my_container` — Delete one or more containers.

`docker container rm $(docker ps -a -q)` — Delete all containers that are not running.

Those are the eight essential commands for Docker containers.

To recap, you first create a container. Then, you start the container. Or combine those steps with `docker run my_container`. Then, your app runs. Yippee!

Then, you stop a container with `docker stop my_container`. Eventually you delete the container with `docker rm my_container`.

Now, let's turn to the magical container-producing molds called images.

Images

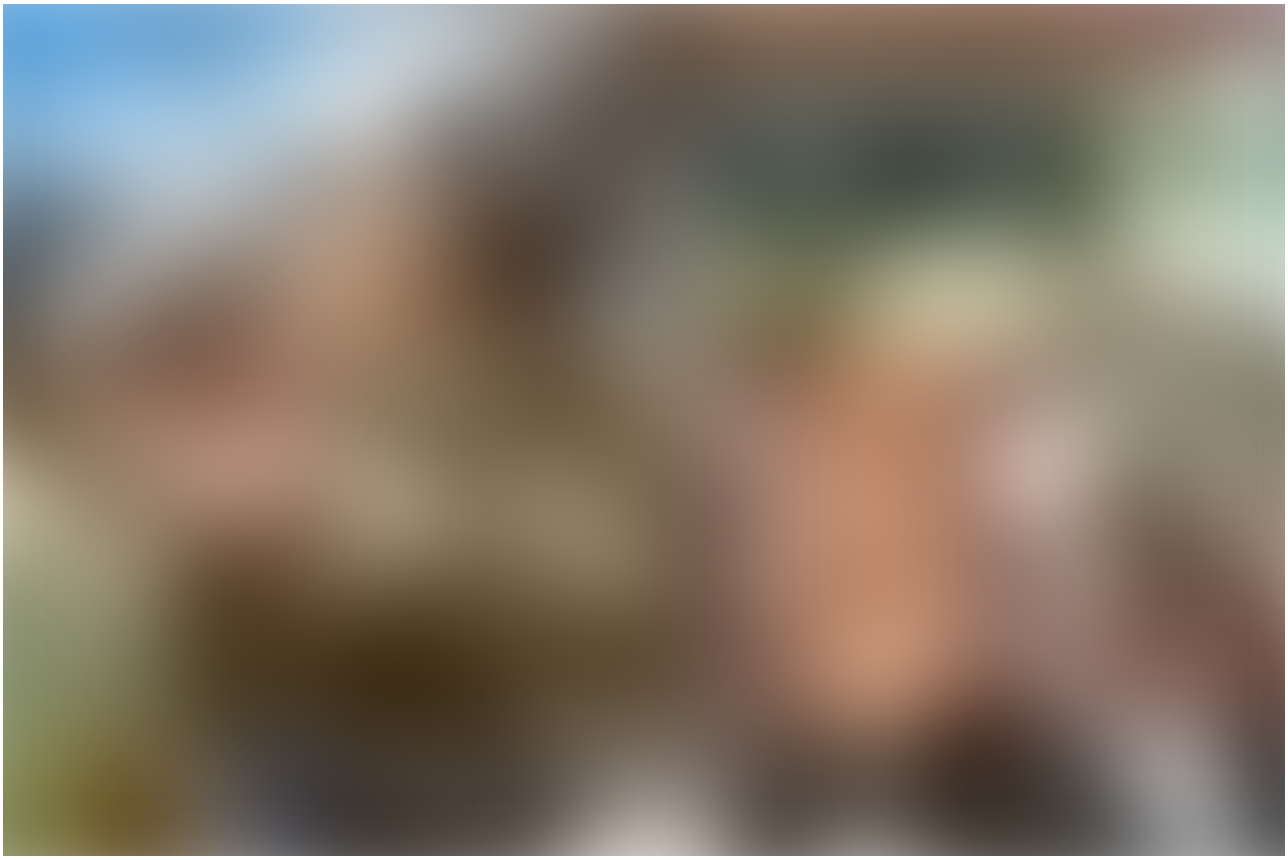
Here are seven commands for working with Docker images.

Developing Images

`docker image build -t my_repo/my_image:my_tag .` — Build a Docker image named *my_image* from the Dockerfile located at the specified path or URL.

`-t` is short for tag. Tells docker to tag the image with the provided tag. In this case *my_tag* .

The `.` (period) at the end of the command tells Docker to build the image according to the Dockerfile in the current working directory.



Build it

Once you have an image built you want to `push` it to a remote registry so it can be shared and pulled down as needed. Assuming you want to use Docker Hub, go there in your browser and create an account. It's free. 😄

This next command isn't an image command, but it's useful to see here, so I'll mention it.

`docker login` — Log in to a Docker registry. Enter your username and password when prompted.

Push

`docker image push my_repo/my_image:my_tag` — Push an image to a registry.

Once you have some images you might want to inspect them.

Inspecting Images

Inspection time

`docker image ls` — List your images. Shows you the size of each image, too.

`docker image history my_image` — Display an image's intermediate images with sizes and how they were created.

`docker image inspect my_image` — Show lots of details about your image, including the layers that make up the image.

Sometimes you'll need to clean up your images.

Removing Images

`docker image rm my_image` — Delete the specified image. If the image is stored in a remote repository, the image will still be available there.

`docker image rm $(docker images -a -q)` — Delete all images. Careful with this one! Note that images that have been pushed to a remote registry will be preserved — that's one of the benefits of registries. 😃

Now you know most essential Docker image-related commands. We'll cover data-related commands in the next article.

Commands are like fruit — nutritious and delicious. Err. Yeah.

Misc

`docker version` — List info about your Docker Client and Server versions.

`docker login` — Log in to a Docker registry. Enter your username and password when prompted.

`docker system prune` makes an appearance in the next article. Readers on Twitter and Reddit suggested that it would be good to add to this list. I agree, so I'm adding it.

`docker system prune` —Delete all unused containers, unused networks, and dangling images.

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

`-a` is short for `--all`. Delete unused images, not just dangling ones.

`--volumes` Remove unused volumes. We'll talk more about volumes in the next article.

UPDATE Feb. 7, 2019: Management Commands

In CLI 1.13 Docker introduced management command names that are logically grouped and consistently named. The old commands still work, but the new ones make it easier to get started with Docker. The original version of this article listed the old names. I've updated the article to use the management command names based on reader suggestions. Note that this change only introduces two command name changes — in most cases it just means adding `container` or `image` to the command. A mapping of the commands is here.

Wrap

If you are just getting started with Docker, these are the three most important commands:

`docker container run my_image` — Create a new container and start it. You'll probably want some flags here.

`docker image build -t my_repo/my_image:my_tag .` — Build an image.

`docker image push my_repo/my_image:my_tag` — Push an image to a remote registry.

Here's the larger list of essential Docker commands:

Containers

Use `docker container my_command`

`create` — Create a container from an image.

`start` — Start an existing container.

`run` — Create a new container and start it.

`ls` — List running containers.

`inspect` — See lots of info about a container.

`logs` — Print logs.

`stop` — Gracefully stop running container.

`kill` —Stop main process in container abruptly.

`rm` — Delete a stopped container.

Images

Use `docker image my_command`

`build` — Build an image.

`push` — Push an image to a remote registry.

`ls` — List images.

`history` — See intermediate image info.

`inspect` — See lots of info about an image, including the layers.

`rm` — Delete an image.

Misc

`docker version` — List info about your Docker Client and Server versions.

`docker login` — Log in to a Docker registry.

`docker system prune` — Delete all unused containers, unused networks, and dangling images.

To view the CLI reference when using Docker just enter the command `docker` in the command line. You can see the Docker docs here.

Now you can really build things with Docker! As my daughter might say in emoji: 🍒🥝🍊🍋🍉🍏🍎🍇. Which I think translates to “Cool!” So go forth and play with Docker!

If you missed the earlier articles in this series, check them out. Here’s the first one:

Learn Enough Docker to be Useful

Part 1: The Conceptual Landscape

towardsdatascience.com

In the final article in this series we'll spice things up with a discussion of data in Docker. Follow me to make sure you don't miss it!

I hope you found this article helpful. If you did, please give it some love on your favorite social media channels.

I write about data science, cloud computing, and other tech stuff. Follow me and read more here.

Docker on! 👋

Thanks to Kathleen Hale.

Docker Software Development Programming Technology Towards Data Science