

Docker Command Line Interface (CLI)

Business Science

11/26/2019

Contents

Docker Command Line Interface (CLI)	1
How Docker Works	1
Definitions	2
Installation	2
Docker Commands	2

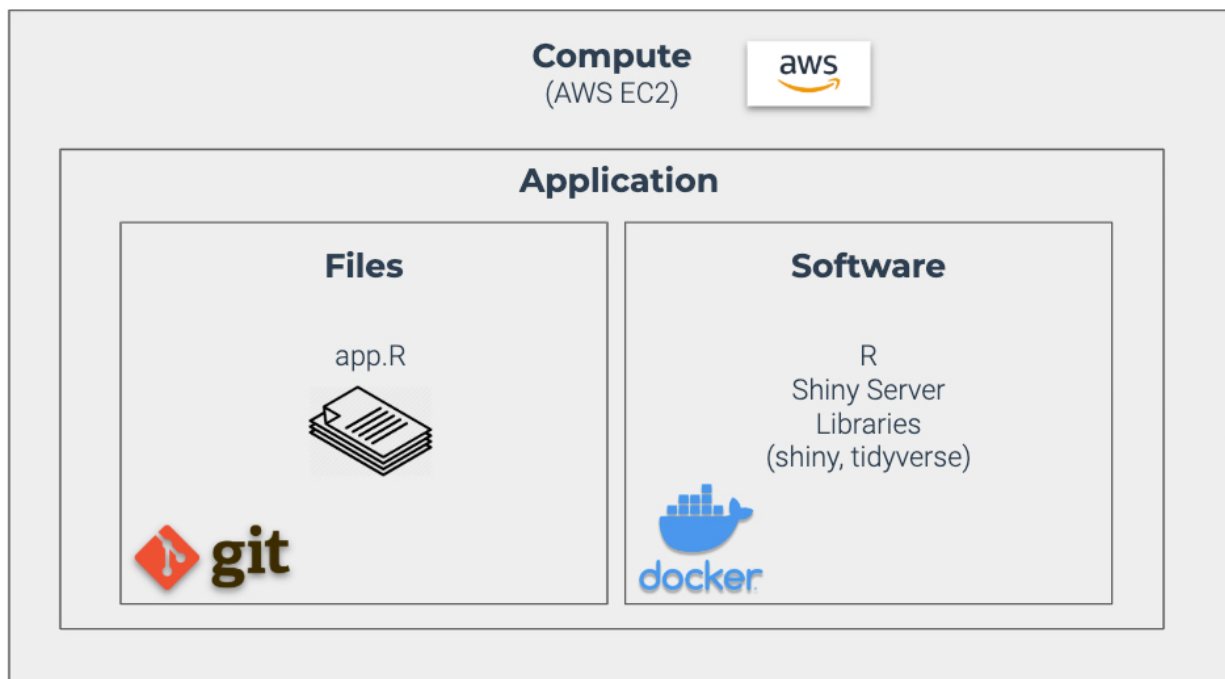
Docker Command Line Interface (CLI)

This document covers the basic `docker` commands used frequently when working with images and containers.

How Docker Works

A Shiny App run on AWS EC2 consists of:

1. **Files** - Controlled by `git` version control
2. **Software Environment** - Controlled by `docker` image



Definitions

1. **DockerHub** - An online community for storing and sharing container images. Has Public and Private repositories for image storage.
2. **Container** - A container is a virtual environment that combines a Docker Image with software (files) to run an application in a controlled environment (a reproducible software environment created virtually from the Docker Image).
3. **Image** - An image is an environment that has been built from a series of instructions called a DockerFile. Images can be prebuilt and hosted on DockerHub (similar to how GitHub hosts version controlled software files). The image is needed to run a Docker Container.

Installation

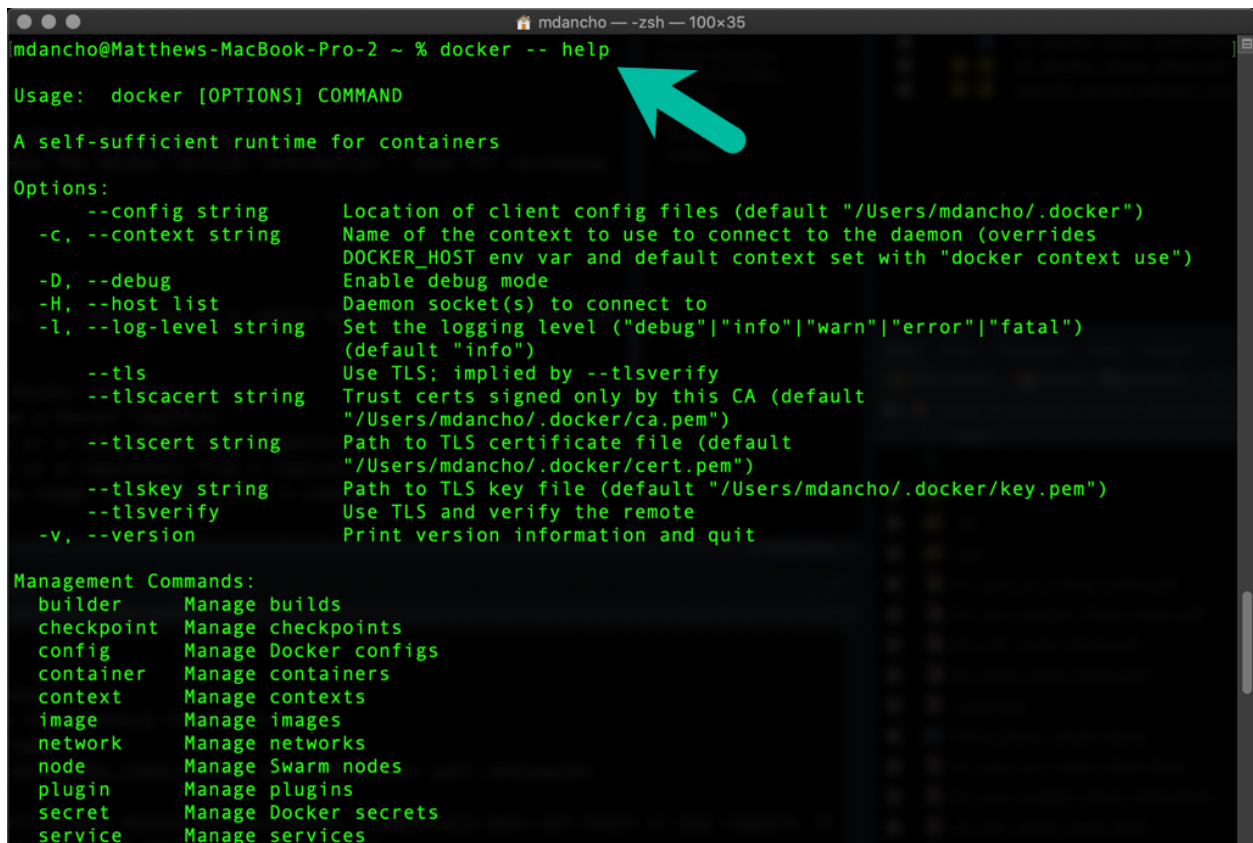
Installing on Ubuntu using **snap** ubuntu package manager.

```
sudo snap install docker
```

Docker Commands

General

- `docker --help` - Show all docker commands
- `docker --version` - Show the Docker version information - Good for verifying that docker is installed



```
mdancho@Matthews-MacBook-Pro-2 ~ % docker -- help
Usage:  docker [OPTIONS] COMMAND

A self-sufficient runtime for containers

Options:
  --config string      Location of client config files (default "/Users/mdancho/.docker")
  -C, --context string  Name of the context to use to connect to the daemon (overrides
                        DOCKER_HOST env var and default context set with "docker context use")
  -D, --debug           Enable debug mode
  -H, --host list       Daemon socket(s) to connect to
  -l, --log-level string Set the logging level ("debug"|"info"|"warn"|"error"|"fatal")
                        (default "info")
  --tls                Use TLS; implied by --tlsverify
  --tlscacert string    Trust certs signed only by this CA (default
                        "/Users/mdancho/.docker/ca.pem")
  --tlscert string       Path to TLS certificate file (default
                        "/Users/mdancho/.docker/cert.pem")
  --tlskey string        Path to TLS key file (default "/Users/mdancho/.docker/key.pem")
  --tlsverify           Use TLS and verify the remote
  -v, --version         Print version information and quit

Management Commands:
  builder      Manage builds
  checkpoint   Manage checkpoints
  config        Manage Docker configs
  container     Manage containers
  context       Manage contexts
  image         Manage images
  network       Manage networks
  node          Manage Swarm nodes
  plugin        Manage plugins
  secret        Manage Docker secrets
  service       Manage services
```

DockerHub

DockerHub (<https://hub.docker.com/>) contains many pre-built images that other users and organizations have created.

These commands are useful for connecting to a remote Docker registry to download docker images.

- **login** - Log in to a Docker registry
- **logout** - Log out from a Docker registry
- **search** - Search the Docker Hub for images
- **push** - Push an image or a repository to a registry
- **pull** - Pull an image or a repository from a registry
- **commit** - Create a new image from a container's changes

Containers

Use `docker container --help` to list all “container management” commands. These are the common commands.

- **run** - Run a command in a new container
- **ps** - List running containers (I use `docker container ls`, which does the same thing)
- **stop** - Stop one or more running containers
- **start** - Start one or more stopped containers
- **rm** - Remove one or more containers (I use `docker container rm`, which does the same thing)

Images

Use `docker image --help` to list “image management” commands.

- **build** - Build an image from a Dockerfile
- **tag** - Create a tag `TARGET_IMAGE` that refers to `SOURCE_IMAGE`
- **images** - List images (I use `docker image ls`, which does the same thing)
- **rmi** - Remove one or more images (I use `docker image rm`, which does the same thing)