Docker Documentation

Kyle Chapman

February 2020

Contents

1 Structure

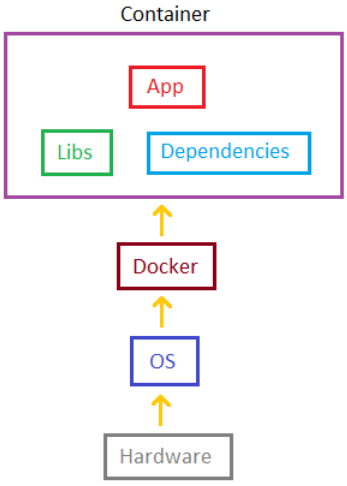


Figure 1: Structure of Docker

A `Dockerfile` is used to create images, and an image is used to deploy an application. A container only lives as long as the process inside it is alive.

2 Dockerfile Format

Given the following `Dockerfile`:

```
1.  FROM ubuntu
2.
3.  RUN apt-get update
4.  RUN apt-get install python
5.
6.  RUN pip install flask
7.
8.  COPY . ~/Desktop/Project/docker-files
9.
10. ENTRYPOINT FLASK_APP=~/Desktop/Project/docker-files/app.py flask run
```

Note:

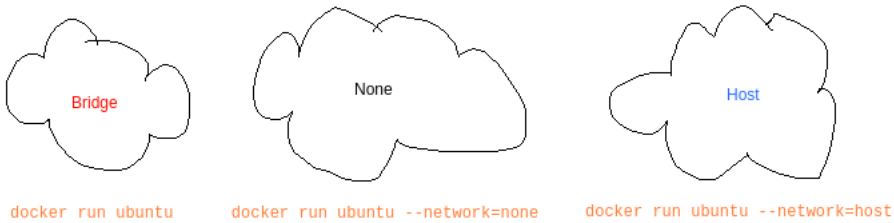
- All words in `CAPS` are **instructions** and all other words represent the **arguments**.
- Every docker image must be based on another image and the `Dockerfile` must start with `FROM`.
- `COPY` copies files from the local storage onto the Docker container.
- `ENTRYPOINT` specifies a command that must be run when starting the image in the Docker container.
- All **arguments** are stored in cache so that if you re-run the `docker build` command, you don‘t need to re-download the necessary files.

Given the following `Dockerfile`:

```
1.  FROM ubuntu
2.
3.  ENTRYPOINT ["sleep"]
4.
5.  CMD ["5"]
```

When running `docker run <appName> <sleepMilli>` with no value given for `sleepMilli`, a default value of `5` will be used as specified by `CMD`. Otherwise, any value entered for `sleepMilli` will replace the `5`.

3 Networking



When using `--network=host`, the port used to access the docker container is also the same port used to access the website from outside the container, without needing to map any ports. However, you may not have more than one of the same image on the same port.

When using `--network=none`, the container is isolated from the network and cannot be accessed from outside the docker container.

4 Commands

Prefix all the following commands with `docker`:

4.1 ps

Lists all the running containers.

Examples:

- `ps` ⇒ List all the running containers.
- `ps -a` ⇒ List all containers that are currently running or have exited.

4.2 history

Shows all the instructions that were executed as a result of building the image.

Example:

- `history <hubUsername>/<appName>` ⇒ Shows the layers executed by building the image `appName`.

4.3 build

Builds a docker image using settings from the `Dockerfile` configuration file.

Example:

- `build <Dockerfile> -t <hubUsername>/<appName>` ⇒ Builds a docker image called `appName` using the configuration found in the `Dockerfile`.

4.4 run

Starts a docker container with the given image, downloading the image if not already present.

Examples:

- `run <imageName>` ⇒ Runs a specific image in a docker container.
- `run <imageName>:<versionNumber>` ⇒ Runs a specific image with a specific version number. If no version number is provided, defaults to the latest version.
- `run -d <imageName>` ⇒ Runs a specific image in a detached state, i.e. no console output. See 4.11 to see console output.
- `run -p <externalPort>:<internalPort> <imageName>` ⇒ Runs a specific image and binds an external port to an internal port, so that users outside of the docker host can access the app using the `externalPort`, which would redirect to the `internalPort` inside the docker host.
- `run -v <dir>:<dockerDir> <imageName>` ⇒ Runs a specific image and persists all changes to the `dockerDir` directory to the `dir` directory.
  - If `dir` is not a complete path, i.e. `data-dir`, then docker will create a volume called `data-dir` in the `/var/lib/docker/volumes` directory, if not already present. Known as '**volume mounting**'.
  - If `dir` is a complete path, i.e. `/mnt/Desktop/data`, then docker will copy the contents of that directory into the `dockerDir` directory. Known as '**bind mounting**'
- `run -it <imageName>` ⇒ Runs a specific image in interactive mode (`-i`) with a pseudo-terminal (`-t`), allowing for user input.
- `run --mount type=<mountType>,source=<sourceDir>,target=<targetDir> <imageName>` ⇒ Preferred over using `-v` option as it specifies the type of mounting and is easier to read.

4.5 stop

Stops the running container that matches the container ID.

Example:

- `stop <containerID>` ⇒ Stops the container that matches the container ID.

4.6 rm

Removes a container that matches the container ID.

Example:

- `rm <containerID>` ⇒ Removes the container that matches the container ID from the local file system.

4.7 rmi

Removes an image that matches the image name.

Example:

- `rmi <imageName>` ⇒ Remove a specific image that has been downloaded.

4.8 images

Shows all of the downloaded docker images.

4.9 pull

Download a specific docker image from docker hub.

Example:

- `pull <imageName>` ⇒ Download the image with the matching image name.

4.10 exec

Executes a command on a docker container.

Example:

- `exec <containerID> <command>` ⇒ Executes a specific command on the matching container ID, where `<command>` can be `ls -la` as an example.

4.11 attach

Attaches the console to the container ID specified.

Example:

- `attach <containerID>` ⇒ Attach a console to the container ID specified.

4.12 inspect

Shows detailed information about the container specified.

Example:

- `inspect <containerID>` ⇒ Show detailed information about the container with the matching ID.

4.13 logs

Shows the logs of the container that has been specified.

Example:

- `logs <containerID>` ⇒ Show logs for container with matching ID.

4.14 push

Pushes a generated Docker image to the Docker hub.

Example:

- `push <hubUsername>/<appName>` ⇒ Pushes the generated image to Docker hub with the name `appName`.

4.15 network

Used to manage the docker internal network architecture.

Examples:

- `network ls` ⇒ Shows all the current networks.
- `network create --driver <driverType> --subnet <subnetMask>/<numSubnets> <networkName>` ⇒ Creates a new network of type `driverType` that uses internal IP addresses generated from the `subnetMask` and named `networkName`.

4.16 volume

Manages folders inside the docker container at `/var/lib/docker/volumes/`.

Example:

- `volume create <directoryName>` ⇒ Creates a folder named `directoryName` inside the docker container.