

Building Images

`docker build` Build a new image from the source code at PATH

`docker build -t myimg`

`-q` Suppress the output generated by containers

`--rm` Remove intermediate containers after a successful build

`-t` Repository name (and optionally a tag) for the image

Examples:

Simplest possible build instruction:

```
docker build .
```

Name image and tag as v1.5:

```
docker build -t myorg/myimg:1.5 .
```

Misc useful commands

(This section is currently blank in the original image)

Misc useful commands (cont)

`top` Lookup the running processes of a container

`-wait` Block until a container stops, then print its exit code

Run `docker COMMAND --help` for help on the command.

Running Docker

`docker run` starts a process with its own file system, its own networking, and its own isolated process tree

`docker run -itP image`

`--name` name the container

`docker run --name =some name o rg`

`-t` terminal interface

`-i` interactive session

`-d` daemon Mode

`-P` publish all exposed ports

`-p` expose specific port

`-p ip:hostport:containerport`

`--rm` remove intermediate images

`-v` bind mount a volume

`-v /host:/container`

Examples:

run an interactive session:

```
sudo docker run -P --name =some name image
```

run container in background, on port 80:

```
sudo docker run --d -v /etc/app data :/data -p 0.0.0.0 :80:8080 --rm nginx
```

Docker Compose

TBC

Dockerfiles (for creating images)

Dockerfiles (for creating images) (cont)

`EXPOSE` listen on the network ports

`EXPOSE <port> [<port>`

`ENV` set the environment variables

`ENV <key> <value>`

`ENV <key>= <value> <value>`

`ADD /COPY??` copies new files from <src> to them to the container at <dest>

`ADD <src>... <dest>`

`ENTRYPOINT` configure container to run as executable

`ENTRYPOINT ["executable", "`

`VOLUME` create externally mounted volumes

`VOLUME ["/data"]`

`USER` sets the user name or UID to running the image

`USER<user>`

`WORKDIR` sets working dir for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions

`WORKDIR /path/ to/ work`

`ONBUILD` adds a trigger instruction for execution when image is used as base image

`ONBUILD [INSTRUCTION]`

Check out the Manual Page for more detail.

attach	Attach to a running container
cp	Copy files/folders from a container's filesystem to the host path
create	Create a new container
exec	Run a command in a running container
images	List images
export	Stream the contents of a container as a tar archive
import	Create a new filesystem image from the contents of a tarball
inspect	Return low-level information on a container or image
kill	Kill a running container
load	Load an image from a tar archive
logs	Fetch the logs of a container
port	Lookup the public-facing port that is NAT-ed to PRIVATE_PORT
ps	List containers
pull	Pull an image or a repository from a Docker registry server
push	Push an image or a repository to a Docker registry server
rm	Remove one or more containers
rmi	Remove one or more images
save	Save an image to a tar archive
search	Search for an image on the Docker Hub
start	Start a stopped container
stop	Stop a running container

FROM	set base image for this image FROM <image>:<tag>
MAINTAINER	The author of the image MAINTAINER <name>
RUN	execute commands in new layer RUN <command> RUN ["executable", "param1", ...]
CMD	provide default for an executing container CMD ["executable", "param1", ...]
LABEL	add metadata to image LABEL <key>=<value> <key>=<value>



By **Andrew Matthews** (aabs)
cheatography.com/aabs/
aabs.wordpress.com

Published 15th July, 2015.
 Last updated 12th May, 2016.
 Page 1 of 2.

Sponsored by **CrosswordCheats.com**
 Learn to solve cryptic crosswords!
<http://crosswordcheats.com>