Dockerfile Cheat Sheet

(for version 1.12.0)

Usage

The Docker daemon runs the instructions in the Dockerfile one-by-one, committing the result of each instruction to a new intermediate image, before finally displaying the ID of your new image.

`docker build .`	Builds an image from a Dockerfile and a context.
`docker build -f /path/to/a/Dockerfile .`	Use -f flag with docker build to point to a Dockerfile anywhere in your file system
`docker build -t shykes/myapp .`	Use -t flag specify a repository and tag at which to save the new image if the build succeeds

Format

`INSTRUCTION arguments`	Not case-sensitive. However, convention is for them to be UPPERCASE to distinguish them from arguments more easily.

FROM

`FROM <repo>`
`FROM <repo>:<tag>`
`FROM <registry>/<repo>:<tag>`
	Sets the Base Image for subsequent instructions. As such, a valid Dockerfile must have FROM as its first instruction. The image can be any valid repo.

MAINTAINER

`MAINTAINER <name>`	Allows you to set the Author field of the generated images.

RUN

`RUN <command>`	Shell form, the command is run in a shell, which by default is /bin/sh -c on Linux or cmd /S /C on Windows
`RUN ["executable", "param1", "param2"]`	(exec form) To use a different shell, other than '/bin/sh', use the exec form passing in the desired shell.

Example

```
RUN ["/bin/bash", "-c", "echo hello"]
```

CMD

Program to run after container is launched. There can only be one CMD instruction in a Dockerfile. If you list more than one CMD then only the last CMD will take effect.

`CMD ["executable","param1","param2"]` exec form (preferred form)

Example

```
CMD ["/usr/bin/wc","--help"]
```

`CMD command param1 param2`	(shell form) using the shell form of the CMD, then the <command> will execute in /bin/sh -c

Example

```
CMD echo "This is a test." | wc -
```

LABEL

Adds labels/metadata to an image.

Example

```
LABEL "Vendor="school_of_devops" Version="1.0"
```

EXPOSE

Informs Docker that the container listens on the specified network ports at runtime. EXPOSE by default does not make the ports of the container accessible to the host. To do that, you must use either the -p flag to publish a range of ports or the -P flag to publish all of the exposed ports.

`EXPOSE <port> [<port>...]` Can have multiple values

Example

```
EXPOSE 80 8989
```

ENV

`ENV <key> <value>`	Sets the environment variable <key> to the value <value>

Example

```
ENV myName John Doe
ENV myDog Rex The Dog
```

`ENV <key>=<value> ...`	This allows for multiple variables to be set at one time

Example

```
ENV myName="John Doe" myDog=Rex\ The\ Dog \
```

ADD

Copies new files, directories or remote file URLs from <src> and adds them to the filesystem of the container at the path <dest>. More advanced than copy. Extracts archives automatically.

`ADD <src>... <dest>`	paths containing without whitespace
`ADD ["<src>",... "<dest>"]`	this form is required for paths containing whitespace

Copy

Copies new files or directories from <src> and adds them to the filesystem of the container at the path <dest>.

`COPY <src>... <dest>`

`COPY ["<src>",... "<dest>"]`	This form is required for paths containing whitespace

ENTRYPOINT

This run after launching the container, before running CMD. Typically used to perform initializitation before launching the application. e.g. Initializing databases, Creating `users` in case of mysql server.

`ENTRYPOINT ["executable", "param1", "param2"]` exec form, preferred

Example

```
ENTRYPOINT ["/usr/sbin/apache2ctl", "-D", "FOREGROUND"]
ENTRYPOINT ["top", "-b"]
```

`ENTRYPOINT command param1 param2` shell form

Example

```
ENTRYPOINT top -b
```

VOLUME

`VOLUME ["/data"]`	Create and mount a volume to hold persistent data. This bypasses container's union file system and is created on the docker host. Volumes are excluded if a image is created with runing container.

Example

```
VOLUME /tmp/dockerdata
```

USER

Sets the user to run commands. Applies to RUN, CMD and ENTRYPOINT instruction that follows after this.

`USER daemon` Setting the user named 'daemon'

WORKDIR

sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instruction that follows

`WORKDIR /path/to/workdir`

ARG

Defines a variable that users can pass at build-time to the builder with the docker build command using the --build-arg <varname>=<value> flag.
If a user specifies a build argument that was not defined in the Dockerfile, the build outputs an error.

`ARG <name>[=<default value>]`

Example

Dockerfile author may optionally specify default value for an ARG

```
FROM busybox
ARG user1=someuser
ARG buildno=1
```

A user builds this file by calling:

```
$ docker build --build-arg user=what_user Dockerfile
```

```
1 FROM busybox
2 USER ${user:-some_user}
3 ARG user
4 USER $user
...
```

The USER at **line 2** evaluates to **some_user** as the user variable is defined on the subsequent **line 3.**
The USER at **line 4** evaluates to **what_user** as user is defined and the **what_user** value was passed on the command line.
Prior to its definition by an ARG instruction, any use of a variable results in an empty string.

SHELL

The SHELL instruction allows the default shell used for the shell form of commands to be overridden.

`SHELL ["executable", "parameters"]`	SHELL instruction can appear multiple times. Each SHELL instruction overrides all previous SHELL instructions, and affects all subsequent instructions.