Dockerfile Explained with Examples of all Instructions

https://www.learnitguide.net/2018/06/dockerfile-explained-with-examples-of.html

```
# Comment
INSTRUCTION arguments
```

Instructions can be given in lowercase or uppercase letters. But to differentiate from the instructions and arguments, we use uppercase letters.

Dockerfile example:

```
FROM docker.io/centos
MAINTAINER Devops Engineer
RUN yum update && yum -y install httpd
RUN mkdir -p /data/myscript
WORKDIR /data/myscript
CMD _python app._py
```

We have listed all dockerfile instructions and dockerfile explained with examples below.

FROM
FROM instruction used to specify the valid docker image name. So specified Docker Image will be downloaded from docker hub registry if it is not exists locally.

Examples:

```
FROM docker.io/centos:latest
FROM docker.io/centos:6
```

If tag "6" is not specfied, FROM instruction will use the latest tag (version).
This is a mandatory instruction in dockerfile, rest all are optional and those can be used based on the requirement.

MAINTAINER
MAINTAINER instruction is used to specify about the author who creates this new docker image for the support.

Examples:

```
MAINTAINER Administrator
MAINTAINER admin @ learnitguide.net
MAINTAINER Devops Engineer(admin @ learnitguide.net)
```

LABEL
LABEL instruction is used to specify metadata informations to an image. A LABEL is a key-value pair.

Examples:

```
LABEL "Application_Environment"="Development"
LABEL "Application_Support"="LearnITGuide.net Group"
```

EXPOSE
EXPOSE instruction is used to inform about the network ports that the container listens on runtime. Docker uses this information to interconnect containers using links and to set up port redirection on docker host system.

Examples:

```
EXPOSE 80 443
EXPOSE 80/tcp 8080/udp
```

ADD
ADD instruction is used to copy files, directories and remote URL files to the destination (docker container) within the filesystem of the Docker Images. Add instruction also has two forms - Shell Form and Executable Form.

Examples:
Shell Form - ADD src dest

```
ADD /root/testfile /data/
```

Executable Form - ADD ["src","dest"]

```
ADD /root/testfile /data/
```

If the "src" argument is a compressed file (tar, gzip, bzip2, etc) then it will extract at the specified "dest" in the container's filesystem.

COPY
COPY instruction is used to copy files, directories and remote URL files to the destination within the filesystem of the Docker Images. COPY instruction also has two forms - Shell Form and Executable Form.

Examples:
Shell Form

```
COPY src dest
COPY /root/testfile /data/
```

Executable Form

```
COPY ["src","dest"]
COPY /root/testfile /data/
```

If the "src" argument is a compressed file (tar, gzip, bzip2, etc), then it will copy exactly as a compressed file and will not extract.

RUN
RUN instruction is used to executes any commands on top of the current image and this will create a new layer. RUN instruction has two forms - Shell Form and Executable Form.

Examples:
Shell form:

```
RUN yum update
RUN systemctl start httpd
```

Executable form:

```
RUN ["yum","update"]
RUN ["systemctl","start","httpd"]
```

CMD
CMD instruction is used to set a command to be executed when running a container. There must be only one CMD in a Dockerfile. If more than one CMD is listed, only the last CMD takes effect.
CMD instruction has two forms - Shell Form and Executable Form.

Example :
Shell form:

```
CMD ping google.com
CMD _python myapplication._py
```

Executable form:

```
CMD ["ping","google.com"]
CMD ["_python","myapplication._py"]
```

ENTRYPOINT
ENTRYPOINT instruction is used to configure and run a container as an executable.
ENTRYPOINT instruction also has two forms - Shell Form and Executable Form.

Examples:
Shell form:

```
ENTRYPOINT ping google.com
ENTRYPOINT python myapplication.py
```

Executable form:

```
ENTRYPOINT ["ping","google.com"]
ENTRYPOINT ["python","myapplication.py"]
```

If user specifies any arguments (commands) at the end of "docker run" command, the specified commands override the default in CMD instruction, But ENTRYPOINT instruction are not overwritten by the docker run command and ENTRYPOINT instruction will run as it is.

So Docker CMD and ENTRYPOINT commands are used for same purpose, but both has some different functionality, refer this link to understand the differences between Docker CMD and ENTRYPOINT Command with examples.

VOLUME
VOLUME instruction is used to create or mount a volume to the docker container from the docker host filesystem.

Examples:

```
VOLUME /data
VOLUME /appdata:/appdata
```

USER
USER instruction is used to set the username,group name, UID and GID for running subsequent commands. Else root user will be used.

Examples:

```
USER webadmin
USER webadmin:webgroup
USER 1008
USER 1008:1200
```

WORKDIR
WORKDIR instruction is used to set the working directory.

Examples:

```
WORKDIR /app/
WORKDIR /java_dst/
```

ENV
ENV instruction is used to set environment variables with key and value. Lets say, we want to set variables APP_DIR and app_version with the values /data and 2.0 respectively. These variables will be set during the image build also available after the container launched.

Examples:

```
ENV APP_DIR /data/
ENV app_version 2.0
```

ARG
ARG instruction is also used to set environment variables with key and value, but this variables will set only during the image build not on the container.

Examples:

```
ARG TMP_NAME mycustom_image
ARG TMP_VER 2.0
```

So both Docker ENV and ARG commands are used to set environment variables, but there are some differences in functionality, refer this link to understand the differences between Docker ENV and ARG Command with examples.

ONBUILD
ONBUILD instruction is used to specify a command that runs when the image in the Dockerfile is used as a base image for another image.

Examples:

```
ONBUILD ADD . /app/data
ONBUILD RUN yum install httpd
```