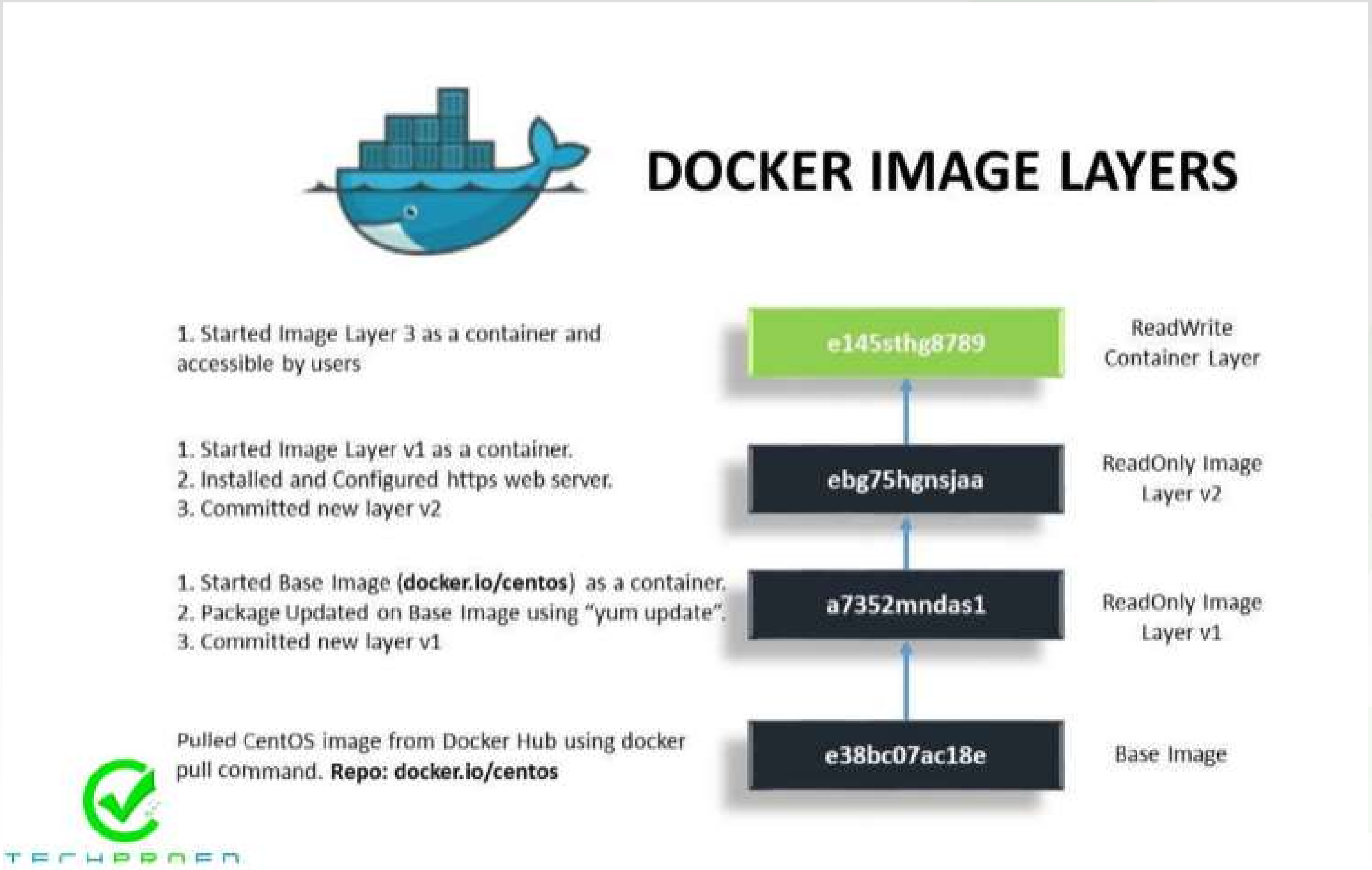
BATCH	BATCH 85
LESSON	Docker
DATE	28.09.2022
SUBJECT	Docker Image

Docker Image



DOCKER IMAGE LAYERS

1. Started Image Layer 3 as a container and accessible by users

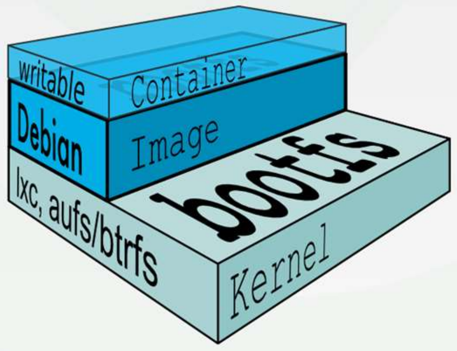
1. Started Image Layer v1 as a container.
2. Installed and Configured https web server.
3. Committed new layer v2

1. Started Base Image (**docker.io/centos**) as a container.
2. Package Updated on Base Image using "yum update".
3. Committed new layer v1

Pulled CentOS image from Docker Hub using docker pull command. **Repo: docker.io/centos**

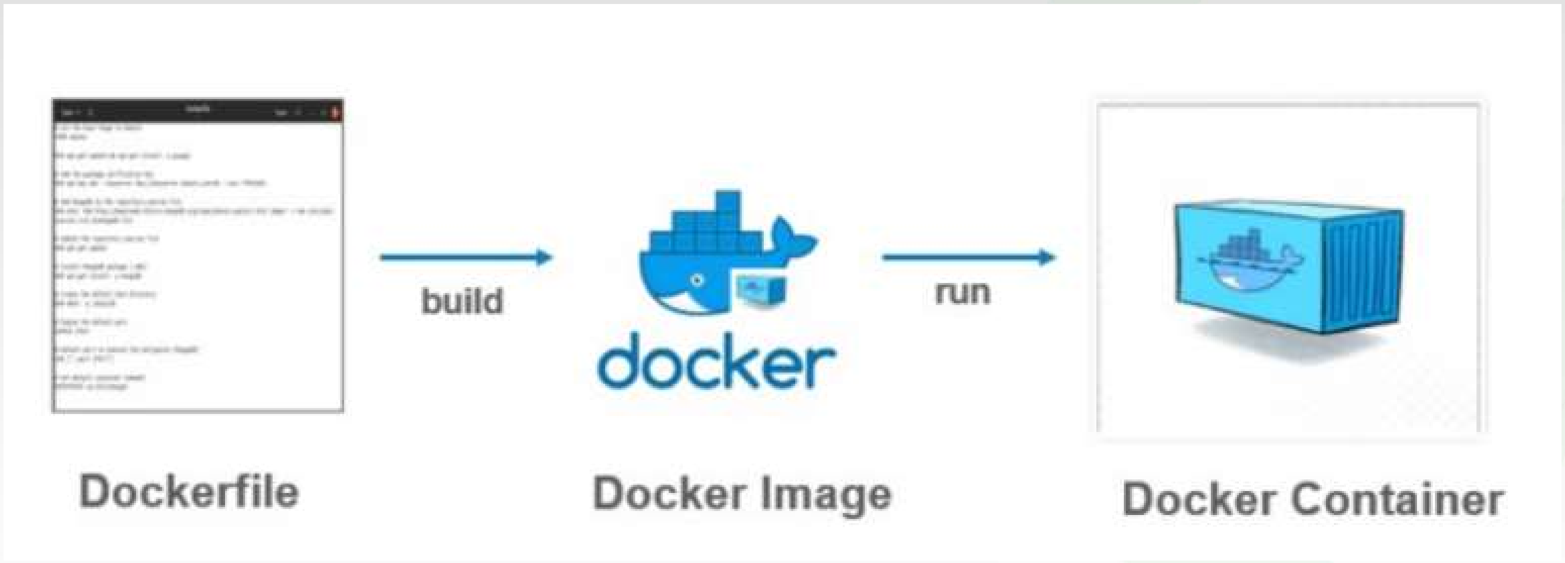
Docker Image

- An image is a collection of files and some metadata
- Images are comprised of multiple layers that referencing another image
- Each image contains source code or software that you want to run
- Every image starts from a base image
- Layers are immutable or read only



Dockerfile

Dockerfile


build
run
docker
Dockerfile
Docker Image
Docker Container

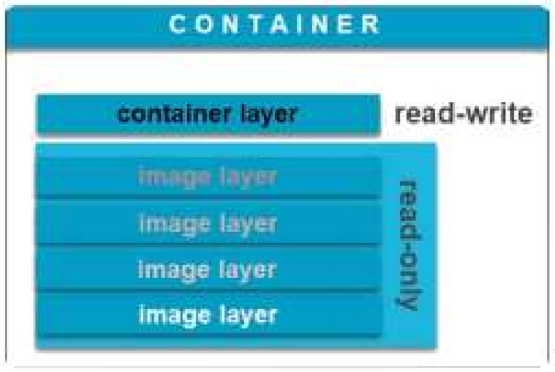
Dockerfile

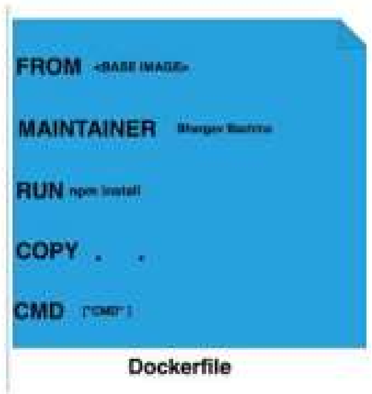
- A Dockerfile is a simple text document as a template that defines the steps of the image creation.

- Each command in the Dockerfile creates a layer in the image

- Dockerfile is featured property of Docker when compared to other technologies ie. VMs

Dockerfile



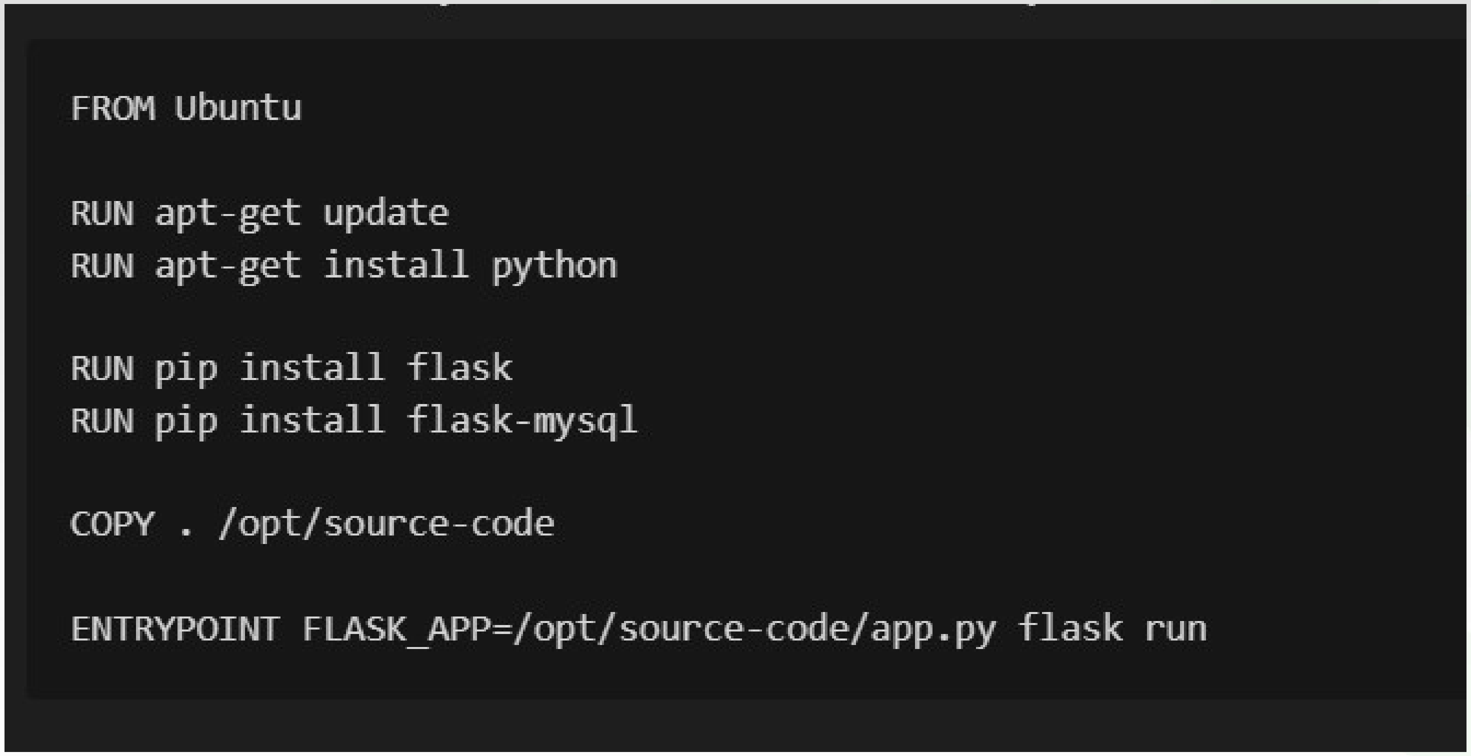


```
FROM <BASE IMAGE>
MAINTAINER Bhargav Bachina
RUN npm install
COPY . .
CMD ["CMD"]
```

Dockerfile

- Use a base Image
- Get the dependencies
- Copy the source code
- Enter a CMD

Dockerfile

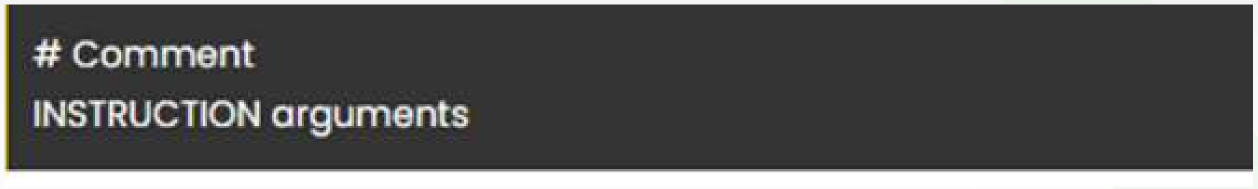


```
FROM Ubuntu

RUN apt-get update
RUN apt-get install python

RUN pip install flask
RUN pip install flask-mysql

COPY . /opt/source-code

ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
```

Dockerfile Commands

Syntax to write instruction and its arguments within a dockerfile is;

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



```
# Comment
INSTRUCTION arguments
```

Dockerfile Instructions

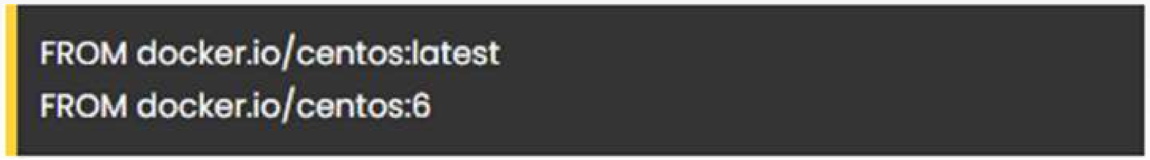
- comments
- FROM
- CMD
- ENTRYPOINT
- WORKDIR
- ENV
- COPY
- LABEL
- RUN
- ADD
- ARG
- EXPOSE
- USER
- VOLUME
- .dockerignore

ECHPROED

Dockerfile Commands

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.

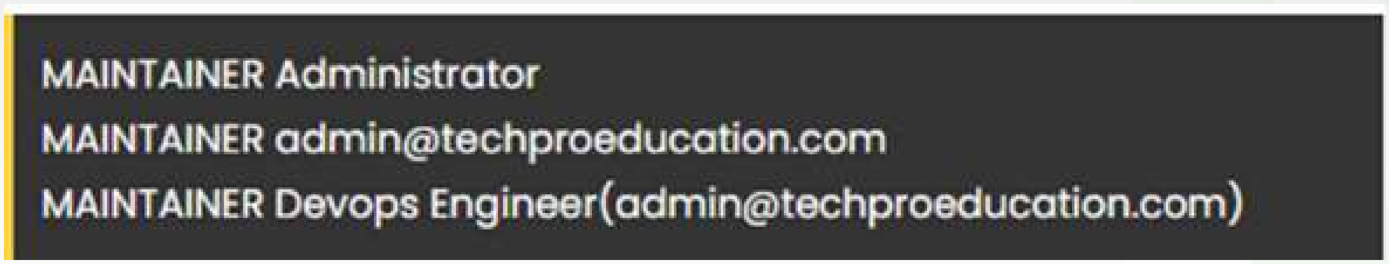


```
FROM docker.io/centos:latest
FROM docker.io/centos:6
```

Dockerfile Commands

MAINTAINER

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

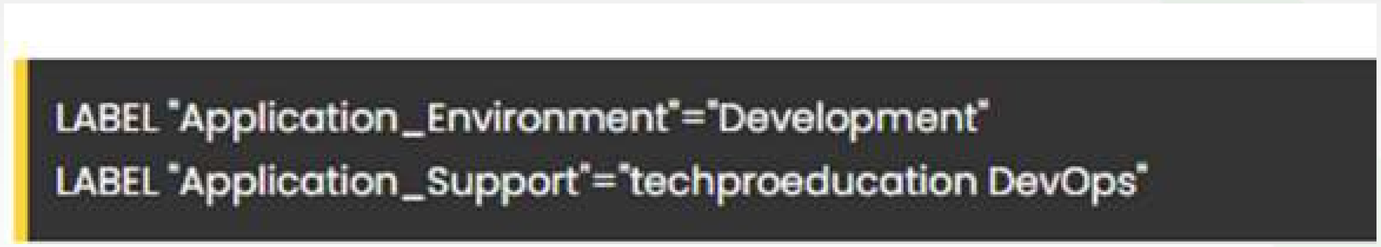


```
MAINTAINER Administrator
MAINTAINER admin@techproeducation.com
MAINTAINER Devops Engineer(admin@techproeducation.com)
```

Dockerfile Commands

LABEL

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

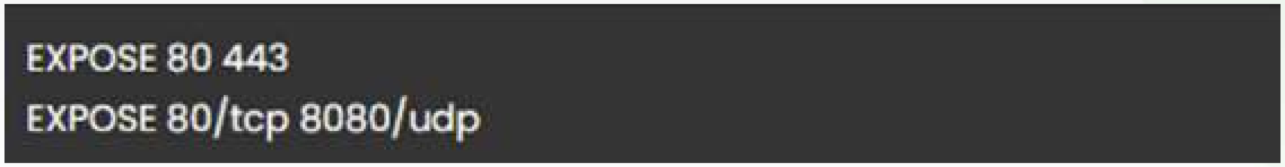


```
LABEL "Application_Environment"="Development"
LABEL "Application_Support"="techproeducation DevOps"
```

Dockerfile Commands

EXPOSE

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

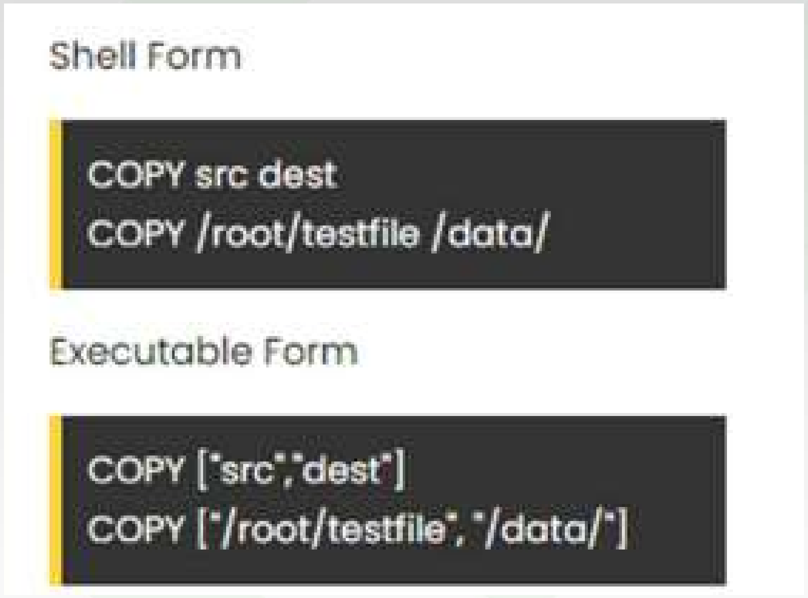


```
EXPOSE 80 443
EXPOSE 80/tcp 8080/udp
```

Dockerfile Commands

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



Shell Form

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

Executable Form

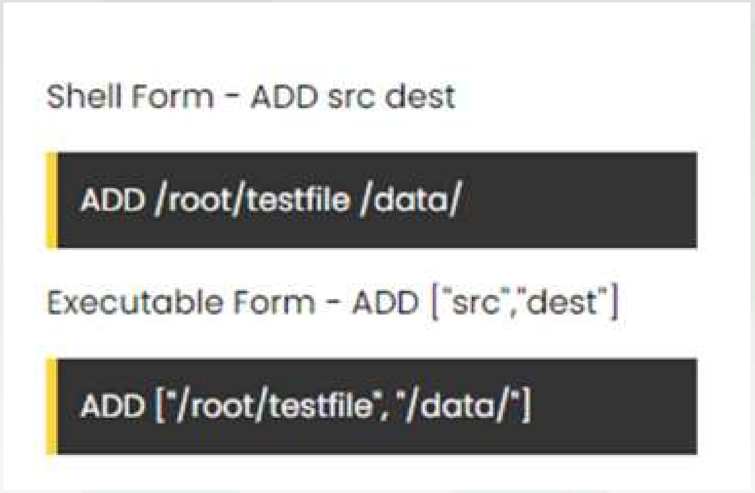
```
COPY ["src","dest"]
COPY ["/root/testfile", "/data/"]
```

```
COPY . /opt/source-code
```

Dockerfile Commands

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 Exec Form



Shell Form - ADD src dest

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

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

```
ADD ["/root/testfile", "/data/"]
```

Dockerfile Commands

RUN

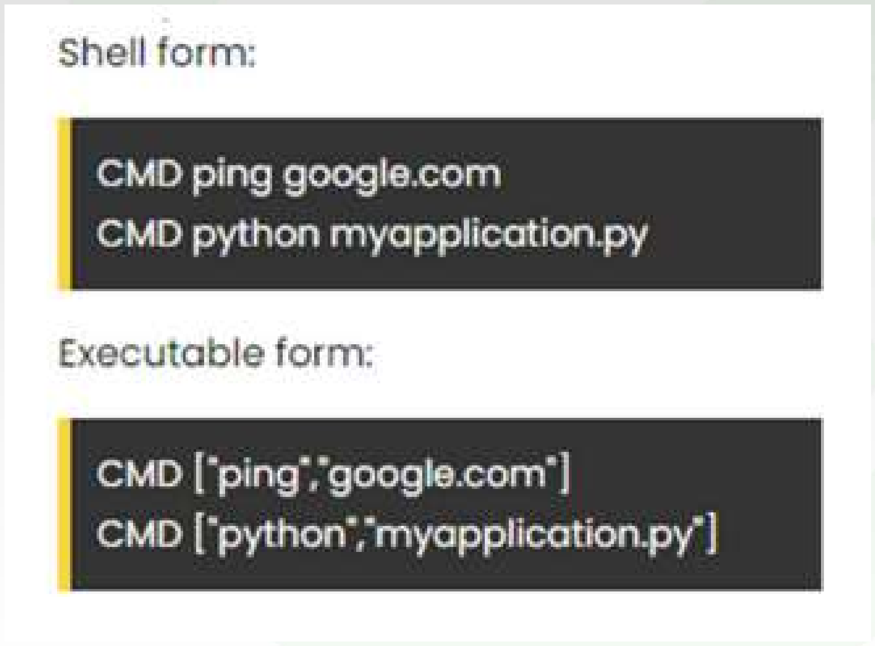
- RUN instruction is used to execute any commands on top of the current image and this will create a new layer.

```
RUN apt-get update
RUN apt-get install python
```

Dockerfile Commands

CMD

- CMD instruction is used to set a command to be executed when running a container. It doesn't execute while build stage.
- There must be only one CMD in a Dockerfile. If more than one CMD is listed, only the last CMD takes effect.


Shell form:

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

Executable form:

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

Dockerfile Commands

ENTRYPOINT

- ENTRYPOINT instruction is used to configure and run a container as an executable.

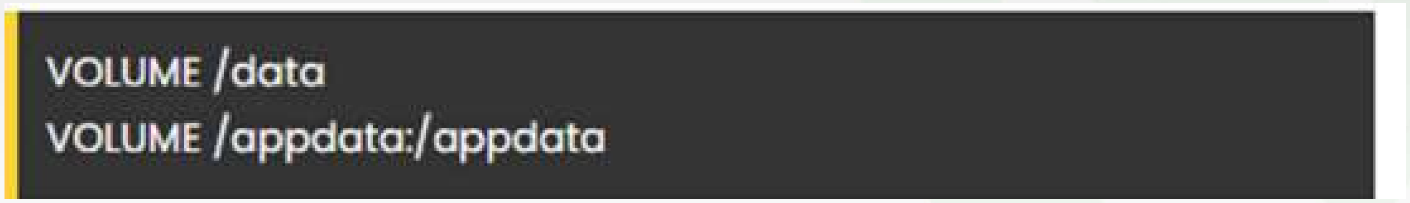
```
ENTRYPOINT FLASK_APP=/opt/source-code/app.py flask run
# Updates Endpoint
```

```
RUN
CMD
ENTRYPOINT
```

Dockerfile Commands

VOLUME

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

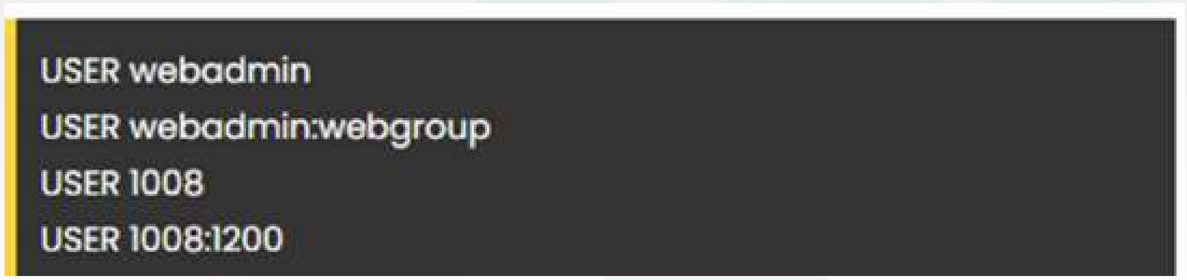


```
VOLUME /data
VOLUME /appdata:/appdata
```

Dockerfile Commands

USER

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

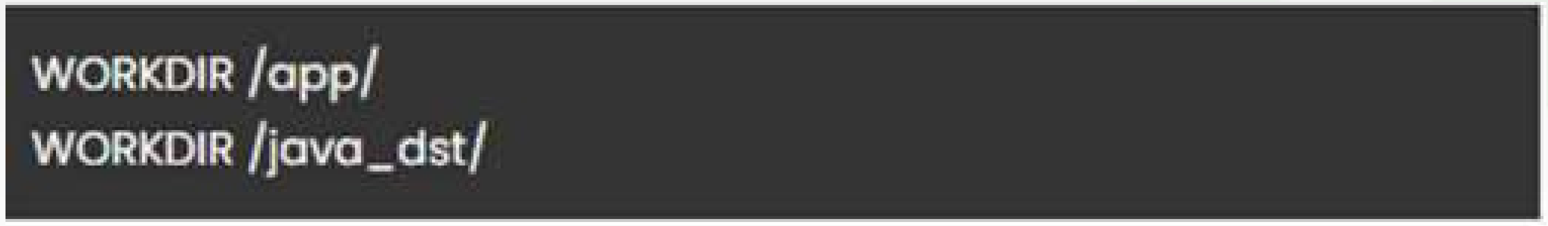


```
USER webadmin
USER webadmin:webgroup
USER 1008
USER 1008:1200
```

Dockerfile Commands

WORKDIR

- WORKDIR instruction is used to set the working directory.

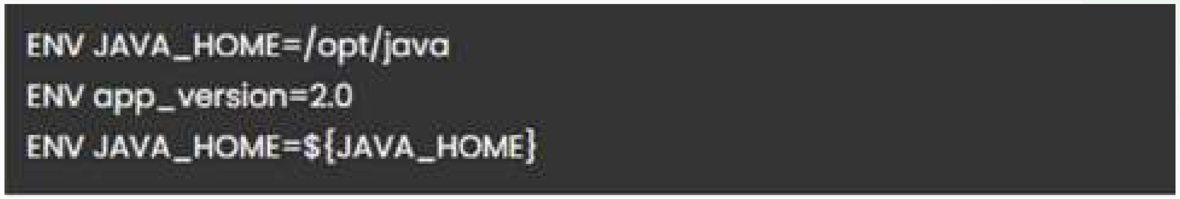


```
WORKDIR /app/
WORKDIR /java_dst/
```

Dockerfile Commands

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 or permanent after the container launched.

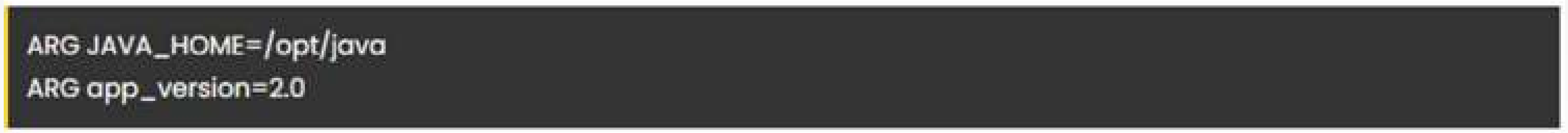


```
ENV JAVA_HOME=/opt/java
ENV app_version=2.0
ENV JAVA_HOME=${JAVA_HOME}
```

Dockerfile Commands

ARG

- ARG instruction is also used to set environment variables with key and value, but this variables will set only during the image build or temporary on the container.

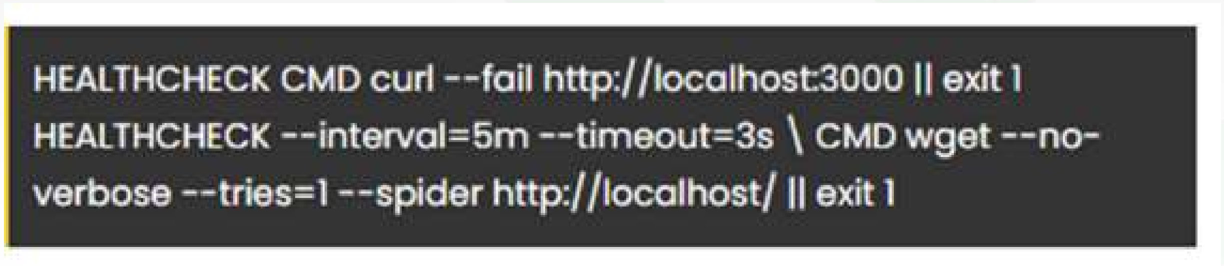


```
ARG JAVA_HOME=/opt/java
ARG app_version=2.0
```

Dockerfile Commands

HEALTHCHECK

- The HEALTHCHECK instruction tells Docker how to test a container to check that it is still working. This can detect cases such as a web server that is stuck in a infinite loop and unable to handle new connections, even though the server process is still running.

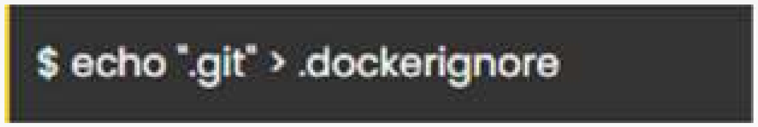


```
HEALTHCHECK CMD curl --fail http://localhost:3000 || exit 1
HEALTHCHECK --interval=5m --timeout=3s \ CMD wget --no-
verbose --tries=1 --spider http://localhost/ || exit 1
```

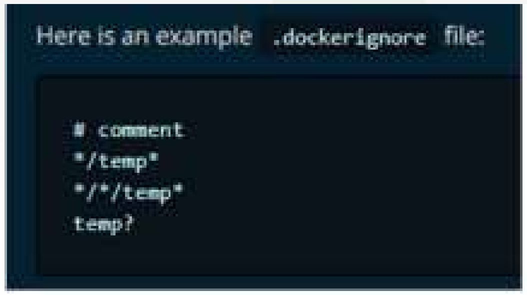
Dockerfile Commands

.dockerignore file

- Before the docker CLI sends the context to the docker deamon, it looks for a file named .dockerignore in the root directory of the context. If this file exists, the CLI modifies the context to exclude files and directories that match patterns in it.



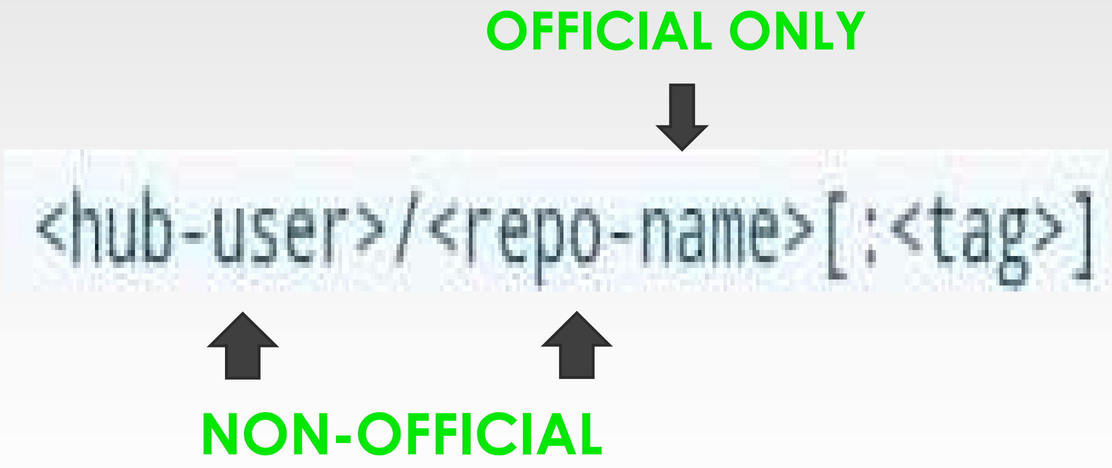
```
$ echo ".git" > .dockerignore
```



Here is an example `.dockerignore` file:

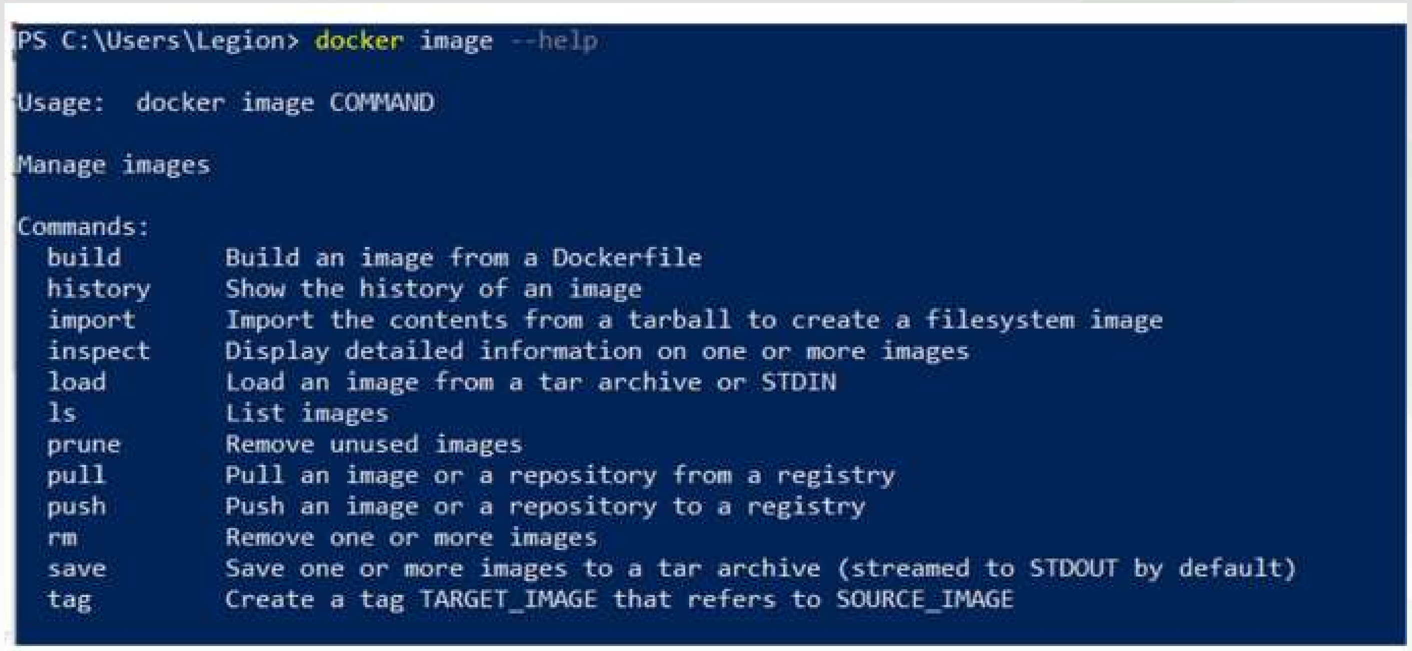
```
# comment
*/temp*
*/*/temp*
temp?
```

Docker Image Naming Convention

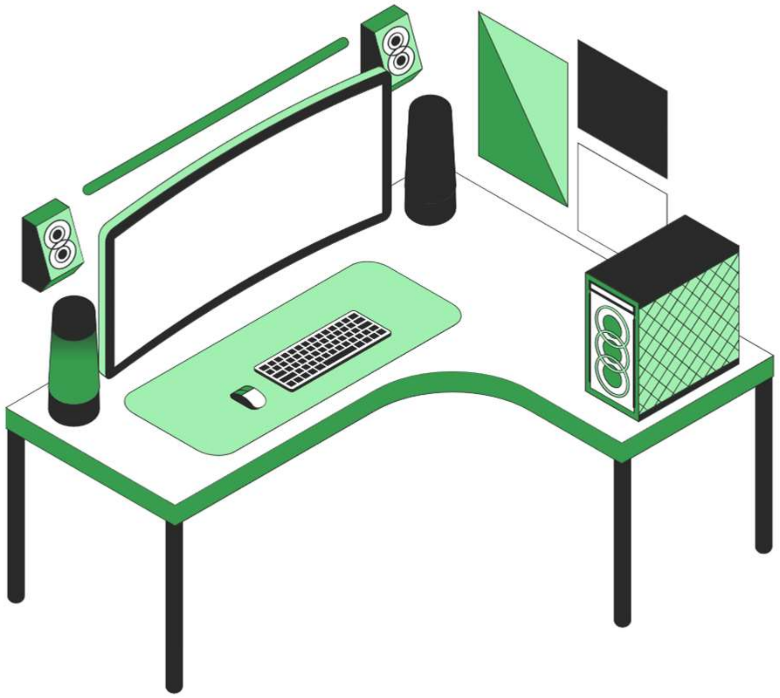


```
<hub-user>/<repo-name>[:<tag>]
```

Docker Image Commands



```
PS C:\Users\Legion> docker image --help

Usage:  docker image COMMAND

Manage images

Commands:
  build       Build an image from a Dockerfile
  history     Show the history of an image
  import      Import the contents from a tarball to create a filesystem image
  inspect     Display detailed information on one or more images
  load        Load an image from a tar archive or STDIN
  ls          List images
  prune       Remove unused images
  pull        Pull an image or a repository from a registry
  push        Push an image or a repository to a registry
  rm          Remove one or more images
  save        Save one or more images to a tar archive (streamed to STDOUT by default)
  tag         Create a tag TARGET_IMAGE that refers to SOURCE_IMAGE
```



Do you have any questions?

Send it to us! We hope you learned something new.

TECHPROED