BATCH	:	B107 AWS-DevOps
LESSON	:	**Docker**
DATE	:	12.04.2023
SUBJECT	:	**Images**

ZOOM GİRİŞLERİNİZİ LÜTFEN **LMS** SİSTEMİ ÜZERİNDEN YAPINIZ

Tips

- Containers are stateless, they do not store your data inside.
- Each container gets an IP address at creation.

- namespaces: running isolated processes
- cgroup: assign resources to namespaces
- container: running processes with dedicated resources

- Docker runs on Linux, on platforms like MacOS, Windows, it uses a tiny Linux environment
- Containers are used for a single application. They are the basic of microservices.

- Docker is made up of
  - a CLI
  - a background daemon (service)
  - REST API

Docker Storage

- Bind Mount

- Used at Development stage
- May lead to sensitive local data/system data
- Risky
- You manage

- Volumes

- Docker recommends for Production stage
- Used for data sharing between containers
- Easy to backup
- Docker manages

- tmpfs

- Used when data is not needed to be stored physically
- Uses RAM

Docker Volume Mapping

host

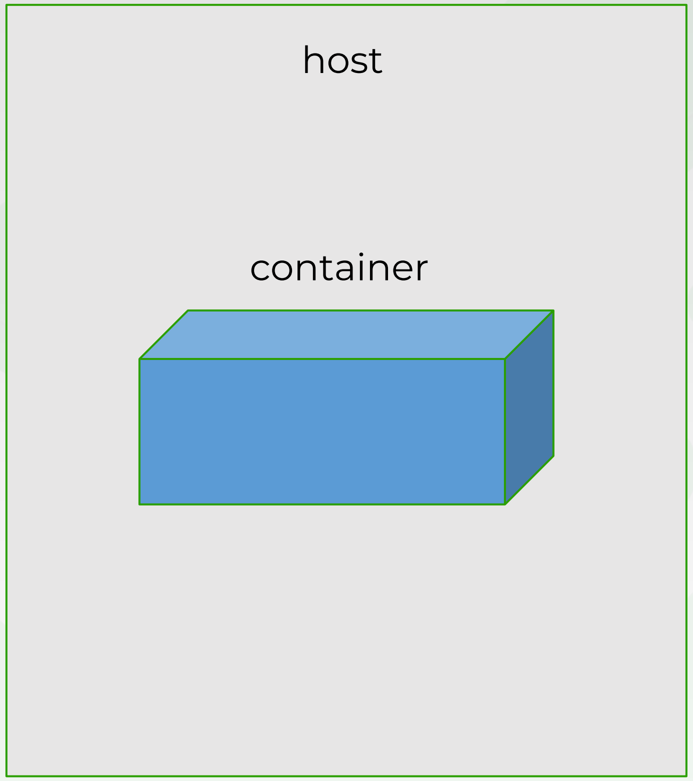
```
docker run
    -v /home/mount/data:/var/lib/mysql/data
```

anonymous

```
docker run
    -v  /var/lib/mysql/data
```

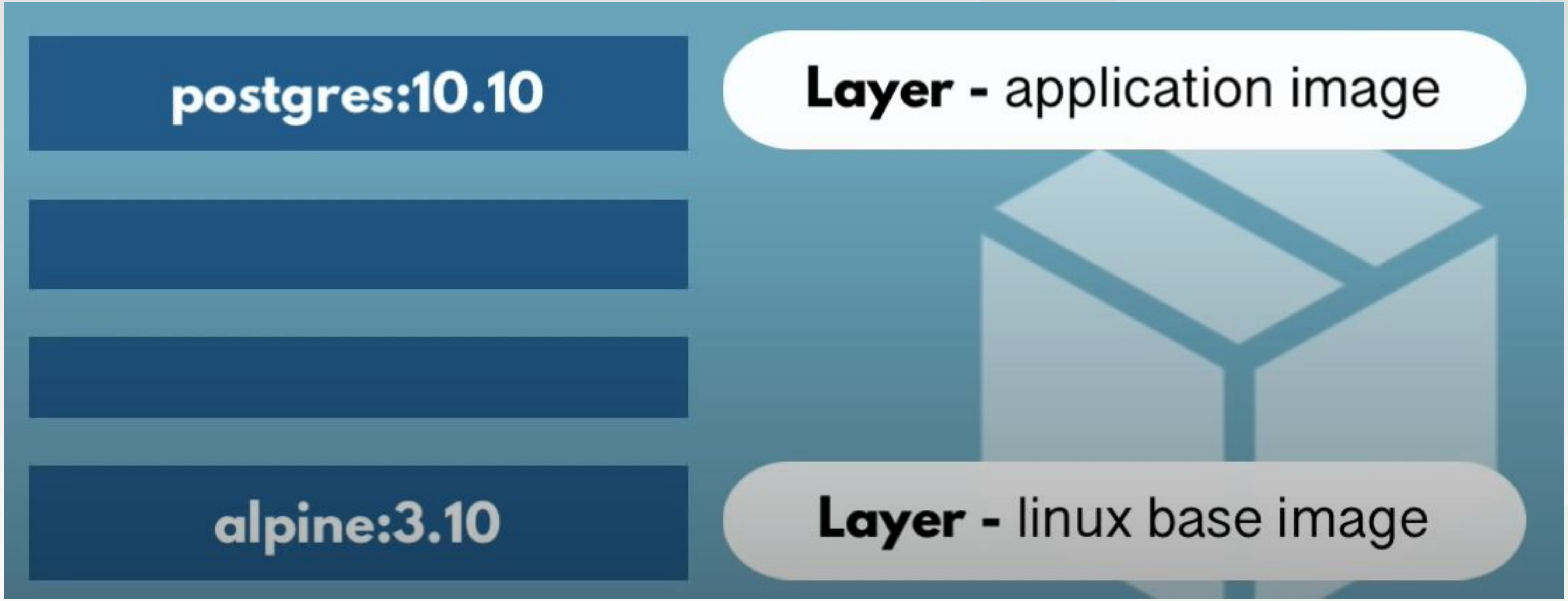
named

```
docker run
    -v  name:/var/lib/mysql/data
```



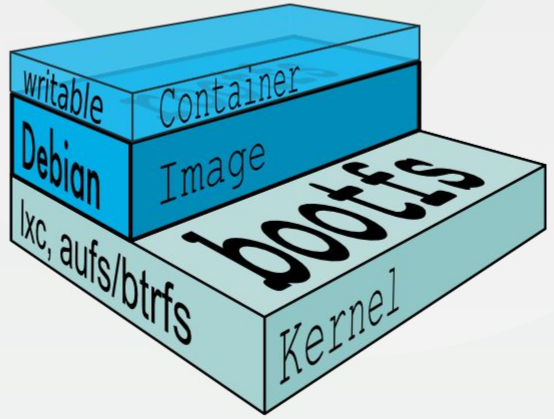
Docker Image

Docker Image



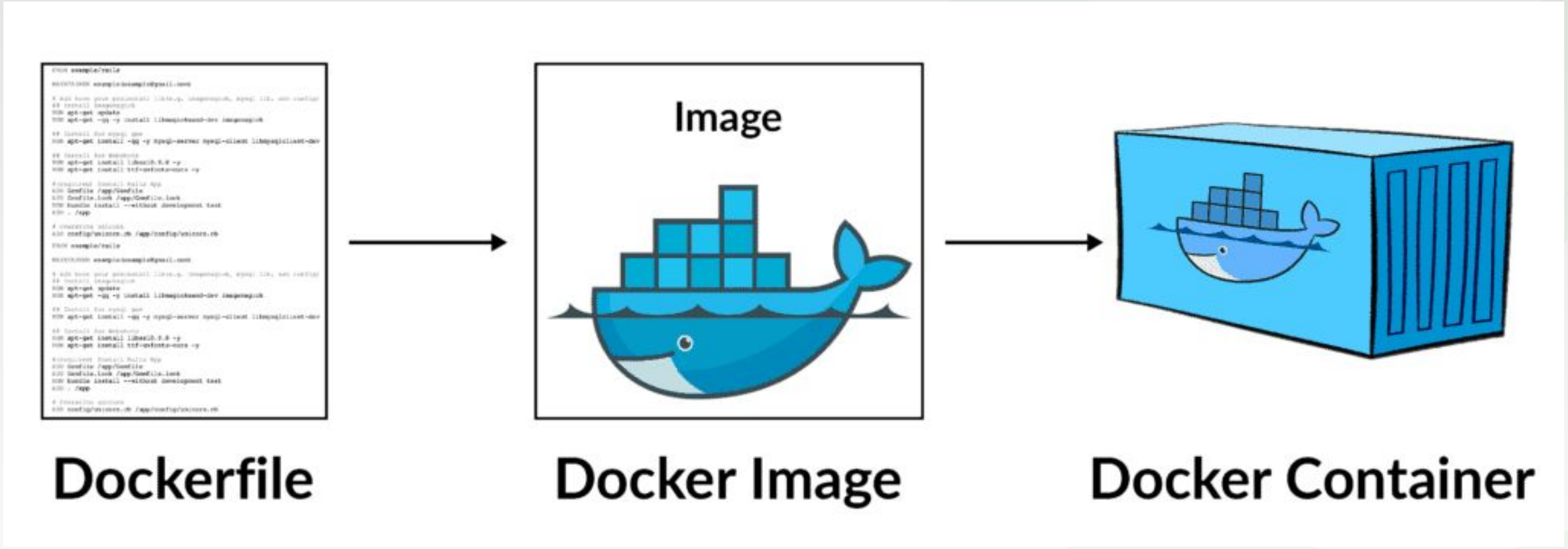
Docker Image

- An image is a collection of files and some metadata
- Images are comprised of multiple layers 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

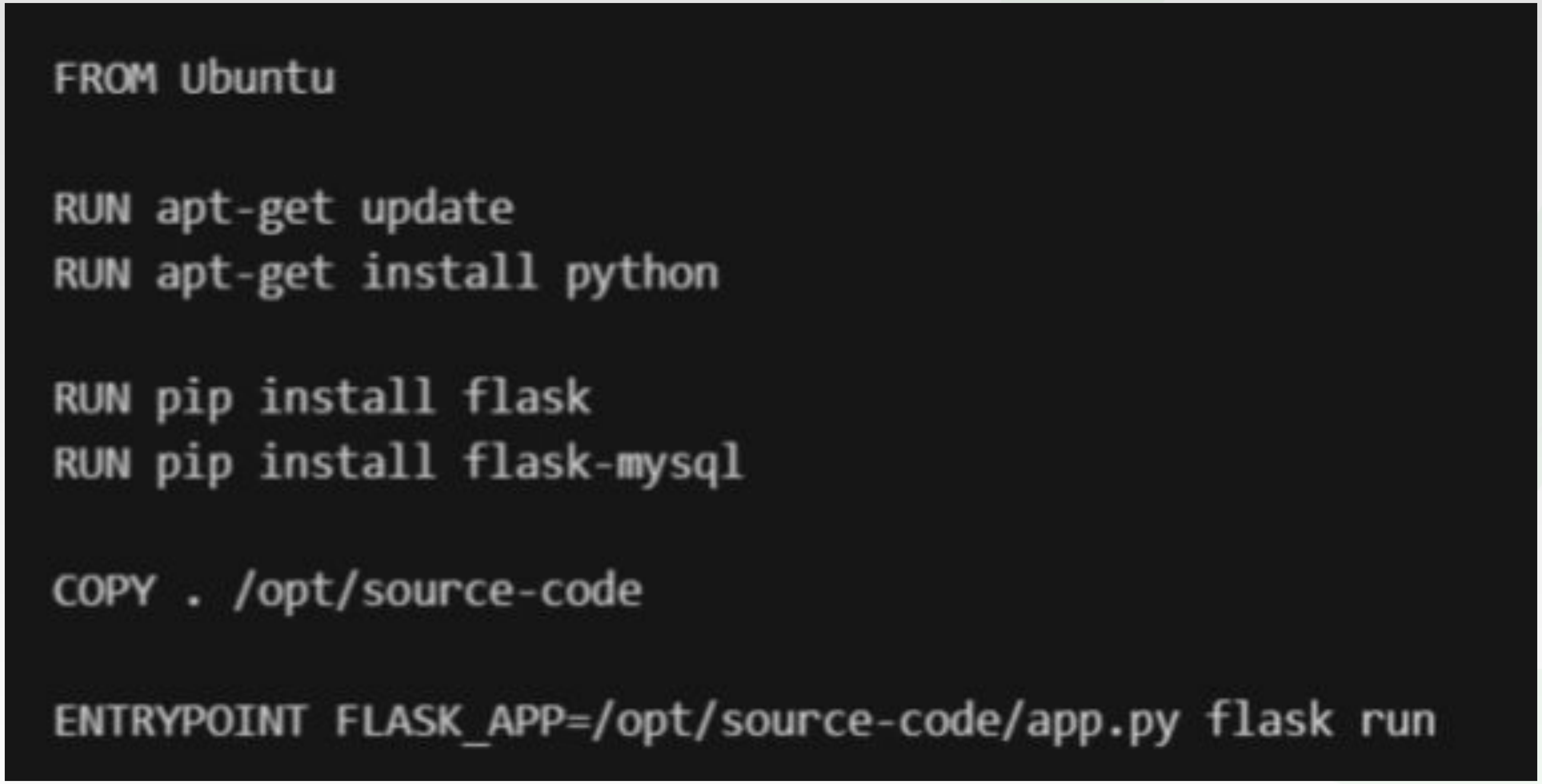


```
FROM example/rails

MAINTAINER example example@gmail.com

# Add here your preinstall lib(e.g. imagemagick, mysql lib, sed config)
## Install ImageMagick
RUN apt-get update
RUN apt-get -qq -y install libmagickwand-dev imagemagick

## Install for mysql gem
RUN apt-get install -qq -y mysql-server mysql-client libmysqlclient-dev

## Install for [illegible]
RUN apt-get install libssl0.9.8 -y
RUN apt-get install ttf-unfonts-core -y

#(required) Install Rails App
ADD Gemfile /app/Gemfile
ADD Gemfile.lock /app/Gemfile.lock
RUN bundle install --without development test
ADD . /app

# Overwrite unicorn
ADD config/unicorn.rb /app/config/unicorn.rb
FROM example/rails

MAINTAINER example example@gmail.com

# Add here your preinstall lib(e.g. imagemagick, mysql lib, sed config)
## Install ImageMagick
RUN apt-get update
RUN apt-get -qq -y install libmagickwand-dev imagemagick

## Install for mysql gem
RUN apt-get install -qq -y mysql-server mysql-client libmysqlclient-dev

## Install for [illegible]
RUN apt-get install libssl0.9.8 -y
RUN apt-get install ttf-unfonts-core -y

#(required) Install Rails App
ADD Gemfile /app/Gemfile
ADD Gemfile.lock /app/Gemfile.lock
RUN bundle install --without development test
ADD . /app

# Overwrite unicorn
ADD config/unicorn.rb /app/config/unicorn.rb
```

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

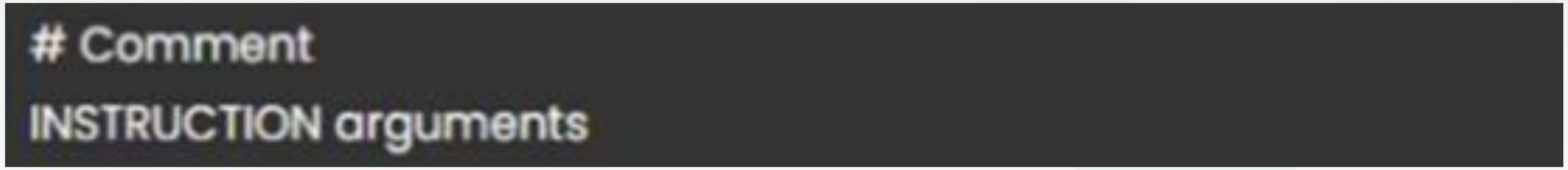
Dockerfile



```
FROM 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

- Instructions can be given in lowercase or uppercase letters.
- We use uppercase letters, in order to differentiate instructions and arguments.



```
# Comment
INSTRUCTION arguments
```

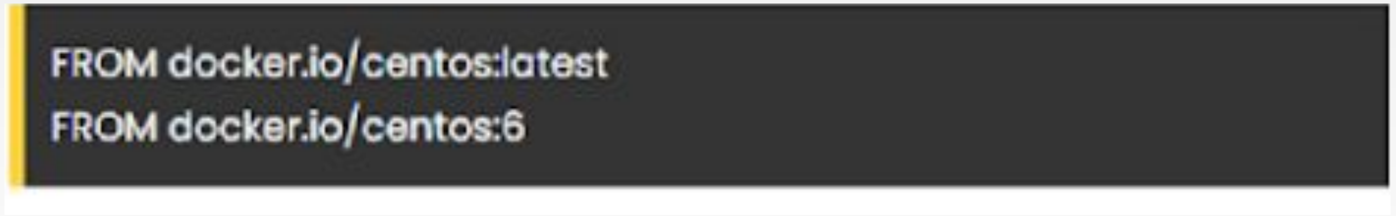
Dockerfile Instructions

Command	Purpose
FROM	To specify the base image which we want to use.
WORKDIR	To define the working directory for any commands that follow in the Dockerfile.
RUN	To install a package or any application.
COPY	To copy over files or directories from a specific location
ADD	Same as COPY, but we can also use a URL instead of a local file / directory and we can extract a tar file from the source directly into the destination.
ENTRYPOINT	Command that will always be executed when the container starts. If not specified, the default is /bin/sh -c
CMD	To define a default command to run when your container starts.
EXPOSE	To define which port through which to access your container application.
LABEL	To add metadata to the image.

Dockerfile Commands

FROM

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

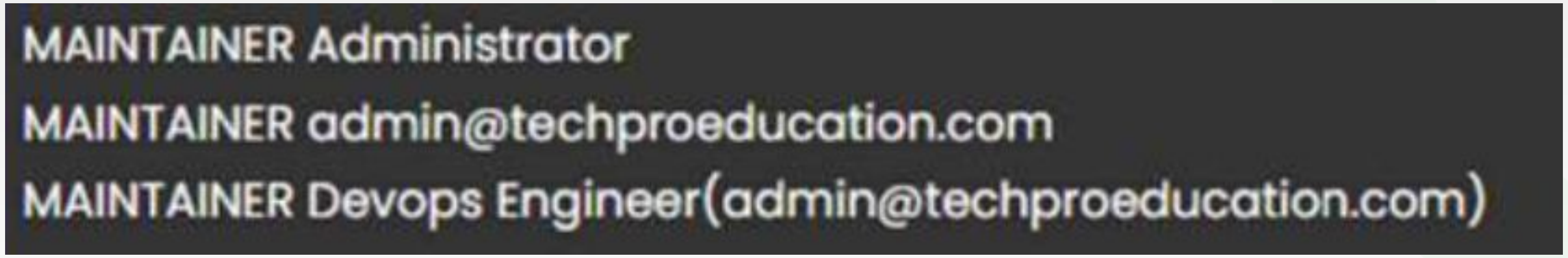


```
FROM docker.io/centos:latest
FROM docker.io/centos:6
```

Dockerfile Commands

MAINTAINER

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

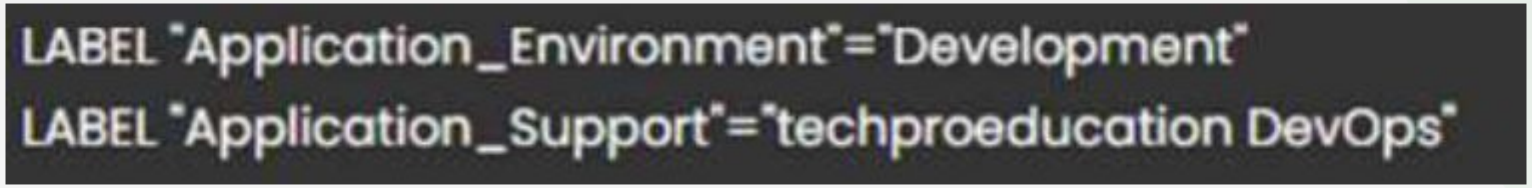


```
MAINTAINER Administrator
MAINTAINER admin@techproeducation.com
MAINTAINER Devops Engineer(admin@techproeducation.com)
```

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 at runtime. Docker uses this information to interconnect containers using links and to set up port redirection on docker host system.
- Does not publish, it is used for documentation purpose.

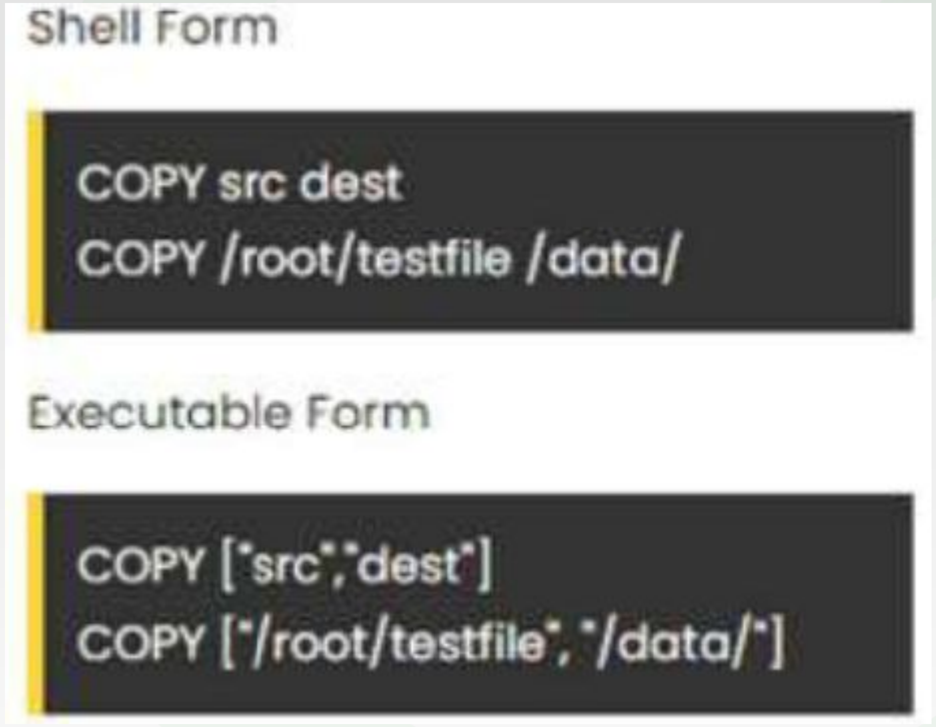
```
EXPOSE 80 443
EXPOSE 80/tcp 8080/udp
```

Dockerfile Commands

COPY

- COPY instruction is used to copy files, directories 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

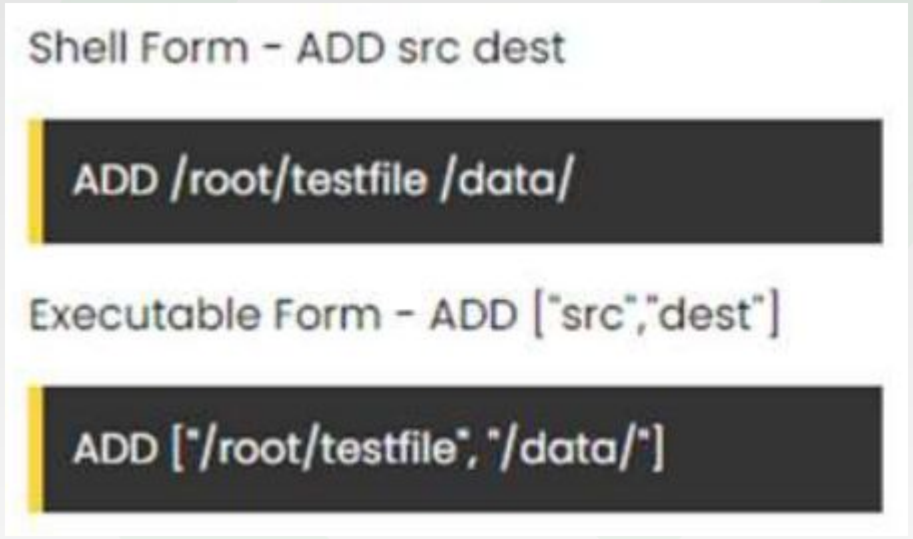
```
COPY ["src","dest"]
COPY ["/root/testfile","/data/"]
```

```
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.
- Auto extracts .tar files
- 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

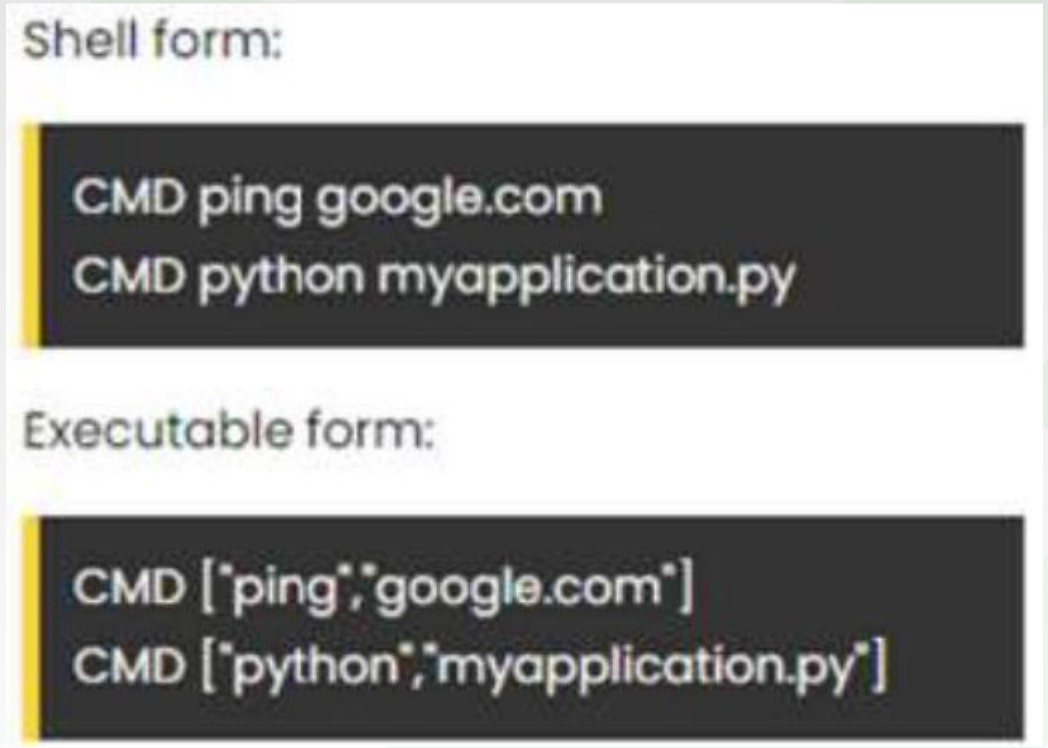
- RUN instruction is used to execute any commands on top of the current image and this will create a new layer.

```
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.

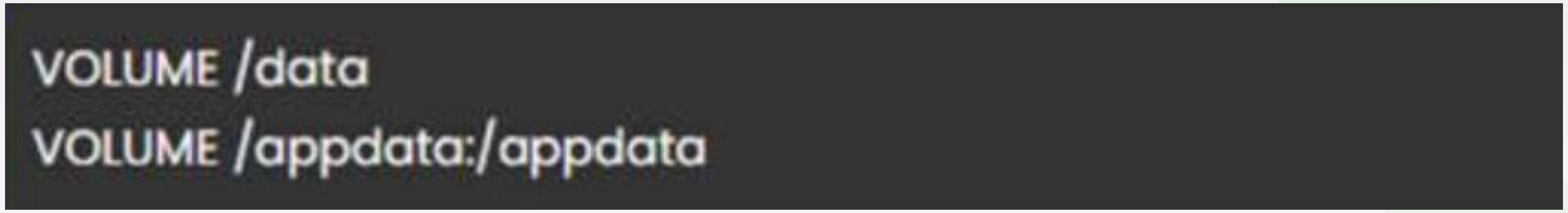
```
ENTRYPOINT ["executable", "param1", "param2"]
ENTRYPOINT command param1 param2
```

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

Dockerfile Commands

VOLUME

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

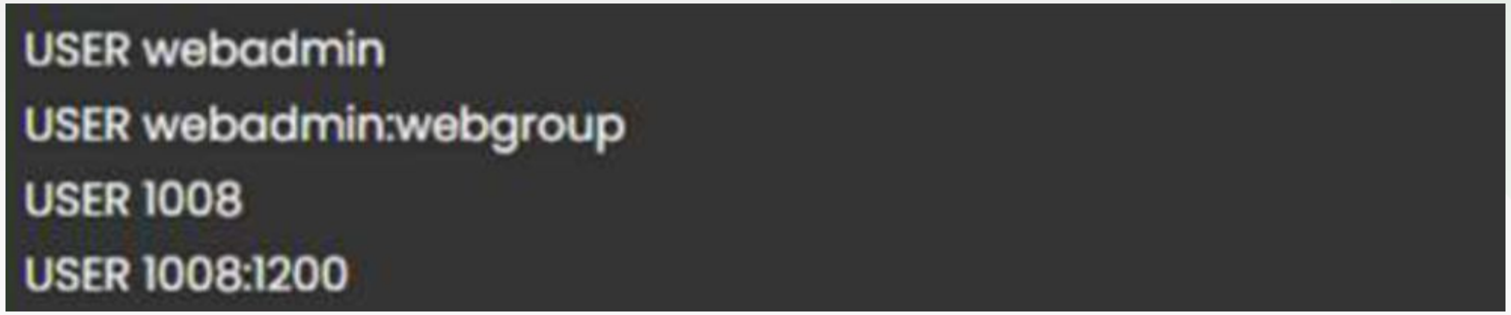


```
VOLUME /data
VOLUME /appdata:/appdata
```

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

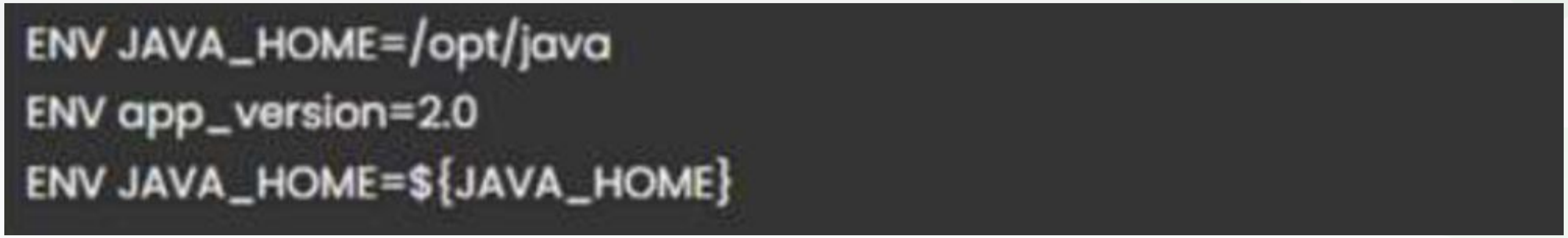
- WORKDIR instruction is used to set the working directory.

```
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.

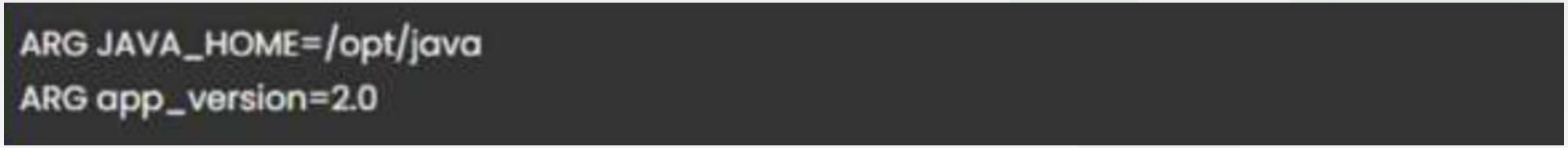


```
ENV JAVA_HOME=/opt/java
ENV app_version=2.0
ENV JAVA_HOME=${JAVA_HOME}
```

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.

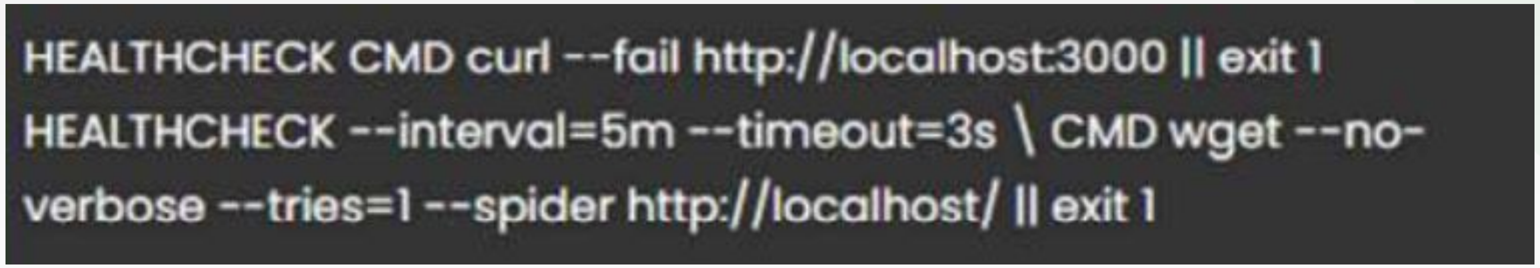


```
ARG JAVA_HOME=/opt/java
ARG app_version=2.0
```

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.

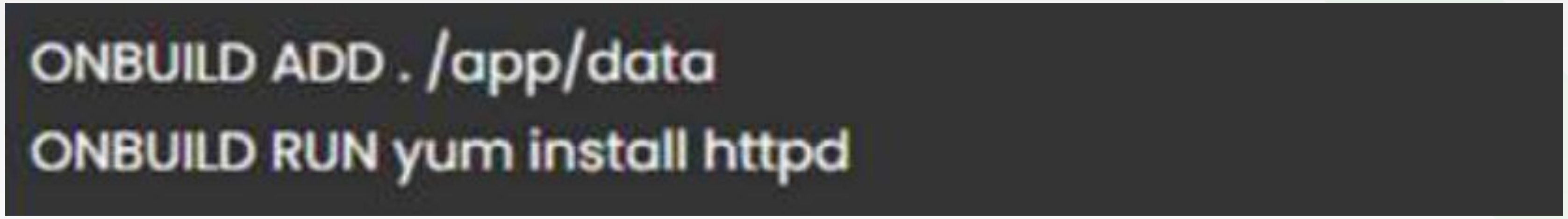


```
HEALTHCHECK CMD curl --fail http://localhost:3000 || exit 1
HEALTHCHECK --interval=5m --timeout=3s \ CMD wget --no-
verbose --tries=1 --spider http://localhost/ || exit 1
```

Dockerfile Commands

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.



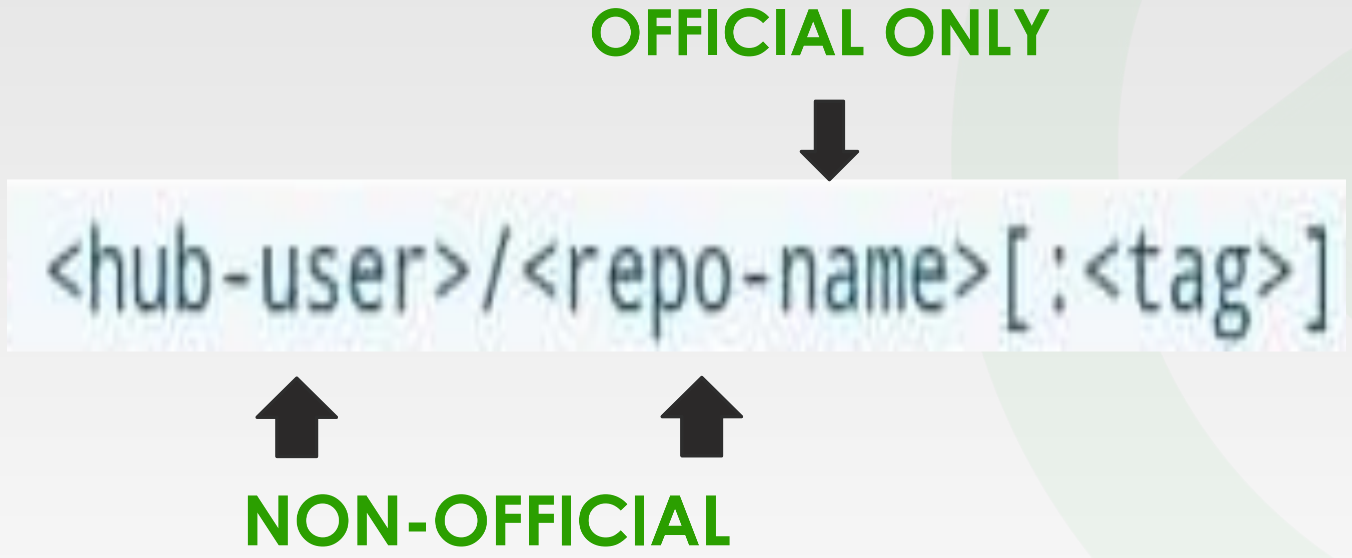
```
ONBUILD ADD . /app/data
ONBUILD RUN yum install httpd
```

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.

Docker Image Naming Convention



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

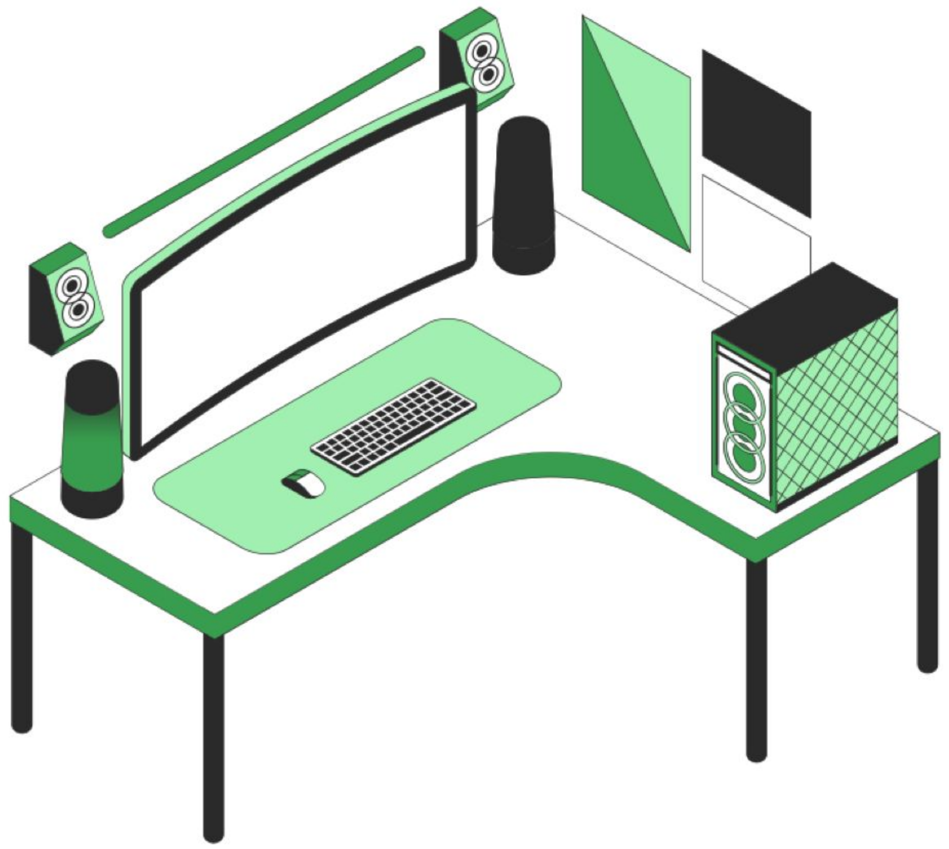
Docker Image Commands

To build image

- docker build -t myimage:tag .

To build another version of the image

- docker commit modifiedContainer newimage



Do you have any questions?

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