Docker Images

SkillAssure

What's In an Image (And what isn't)

- App binaries and dependencies.
- Metadata about the image data and how to run the image
- Not a complete OS. No kernel, Kernel modules (e.g. drivers)

- **Official Definition**: "A Docker image is a file, comprised of multiple layers, that is used to execute code in a Docker container. An image is essentially built from the instructions for a complete and executable version of an application, which relies on the host OS kernel."

Why do you need to create your own image?

- If you want to dockerize your application for ease of shipping and development.
- If you cannot find a component or a service that you want to use as part of your application on docker hub

How to create my own image?

- Let's create an image for a simple web application that is built using the python flask framework.
- First, we need to understand
  - For what application are we creating an image?
  - How the application is built?
- What all we need to deploy this application manually?
  - Install Operating System (e.g. Ubuntu)
  - Update Operating System
  - Install Python
  - Install Pythons Libraries & Dependencies
  - Copy source code
  - Run the application

SkillAssure

How to create my own image?

Dockerfile

```
FROM Ubuntu

RUN apt-get update
RUN apt-get -y 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
```

```
docker build –t prabhavagrawal/my-custom-app .
```

```
docker login --username=prabhavagrawal
```

```
docker push prabhavagrawal/my-custom-app
```

1. OS - Ubuntu
2. Update apt repo
3. Install dependencies using apt
4. Install Python dependencies using pip
5. Copy source code to /opt folder
6. Run the web server using flask command

Dockerfile

Dockerfile

```
INSTRUCTION      ARGUMENT
```

Dockerfile

```
FROM Ubuntu

RUN apt-get update
RUN apt-get -y 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
```

- Everything on the left is an instruction
- Everything on the right is an argument to those instructions

SkillAssure

Dockerfile

Dockerfile

```
INSTRUCTION      ARGUMENT
```

Dockerfile

```
FROM Ubuntu

RUN apt-get update
RUN apt-get -y 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
```

Start from a base OS or another image

- You can find official releases of all operating systems on Docker Hub.
- It's important to note that all Docker files must start with a from instruction

SkillAssure

Dockerfile

Dockerfile

```
INSTRUCTION        ARGUMENT
```

Dockerfile

```
FROM Ubuntu

RUN apt-get update
RUN apt-get -y 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
```

Start from a base OS or another image

Install all dependencies

Copy Source Code

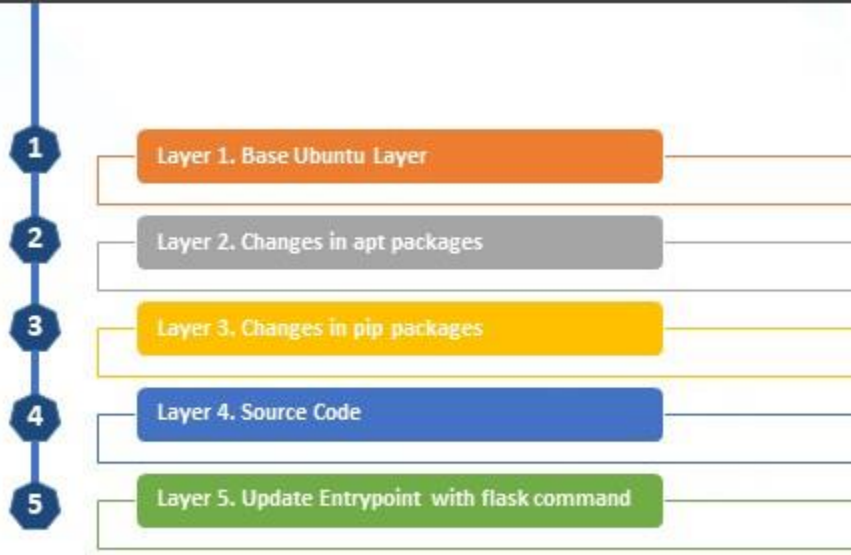
Specify Entrypoint

Layered Architecture

Dockerfile

```
FROM Ubuntu

RUN apt-get update && apt-get -y install python

RUN pip install flask flask-mysql

COPY . /opt/source-code

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

```
docker build Dockerfile –t prabhav/my-custom-app .
```


1
Layer 1. Base Ubuntu Layer
2
Layer 2. Changes in apt packages
3
Layer 3. Changes in pip packages
4
Layer 4. Source Code
5
Layer 5. Update Entrypoint with flask command

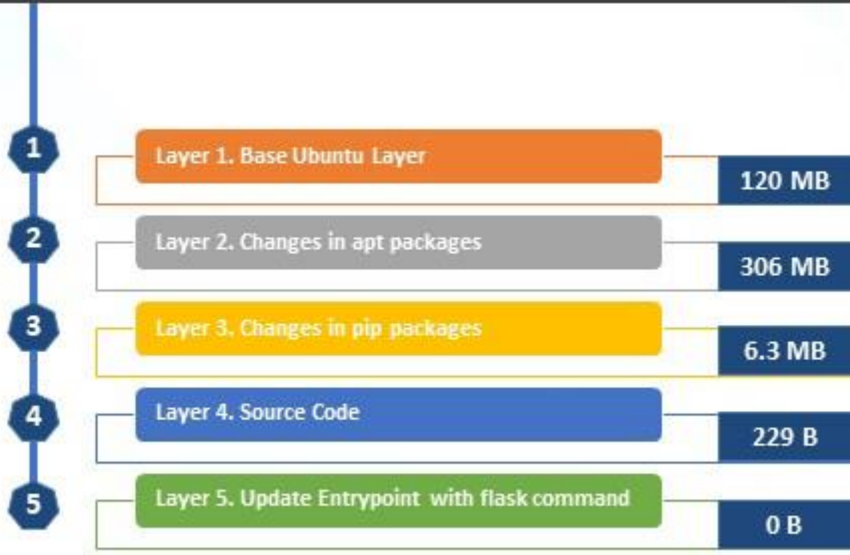
- Docker builds the images in a layered architecture
- Each line of instruction creates a new layer in the docker image with just the changes from the previous layer.

Layered Architecture

Dockerfile

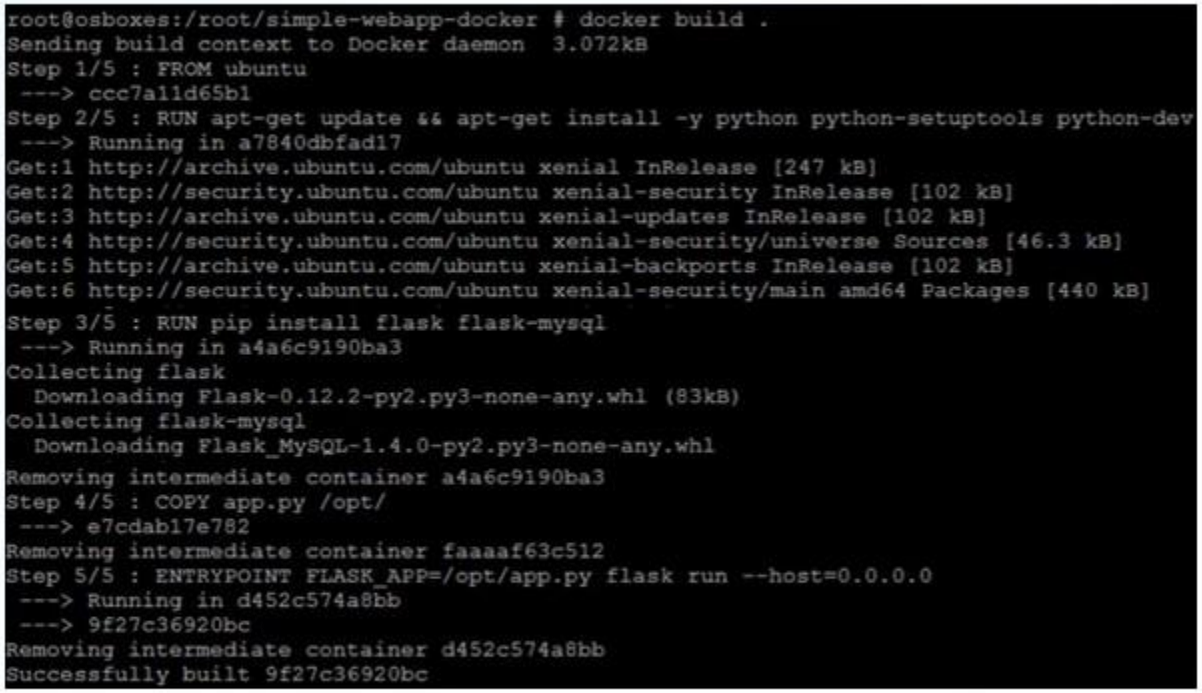
```
FROM Ubuntu

RUN apt-get update && apt-get -y install python

RUN pip install flask flask-mysql

COPY . /opt/source-code

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

```
docker build Dockerfile –t prabhav/my-custom-app .
```


1
Layer 1. Base Ubuntu Layer
120 MB
2
Layer 2. Changes in apt packages
306 MB
3
Layer 3. Changes in pip packages
6.3 MB
4
Layer 4. Source Code
229 B
5
Layer 5. Update Entrypoint with flask command
0 B

- Since each layer only stores the changes from the previous layer, it is reflected in the size as well.
- See the image size information
- **docker history prabhav/simple-webapp**

Docker Build Output



```
root@osboxes:/root/simple-webapp-docker # docker build .
Sending build context to Docker daemon  3.072kB
Step 1/5 : FROM ubuntu
 ---> ccc7a11d65b1
Step 2/5 : RUN apt-get update && apt-get install -y python python-setuptools python-dev
 ---> Running in a7840dbfad17
Get:1 http://archive.ubuntu.com/ubuntu xenial InRelease [247 kB]
Get:2 http://security.ubuntu.com/ubuntu xenial-security InRelease [102 kB]
Get:3 http://archive.ubuntu.com/ubuntu xenial-updates InRelease [102 kB]
Get:4 http://security.ubuntu.com/ubuntu xenial-security/universe Sources [46.3 kB]
Get:5 http://archive.ubuntu.com/ubuntu xenial-backports InRelease [102 kB]
Get:6 http://security.ubuntu.com/ubuntu xenial-security/main amd64 Packages [440 kB]
Step 3/5 : RUN pip install flask flask-mysql
 ---> Running in a4a6c9190ba3
Collecting flask
  Downloading Flask-0.12.2-py2.py3-none-any.whl (83kB)
Collecting flask-mysql
  Downloading Flask_MySQL-1.4.0-py2.py3-none-any.whl
Removing intermediate container a4a6c9190ba3
Step 4/5 : COPY app.py /opt/
 ---> e7cdab17e782
Removing intermediate container faaaaf63c512
Step 5/5 : ENTRYPOINT FLASK_APP=/opt/app.py flask run --host=0.0.0.0
 ---> Running in d452c574a8bb
 ---> 9f27c36920bc
Removing intermediate container d452c574a8bb
Successfully built 9f27c36920bc
```

- When you run the docker build command, you will see the various steps involved and the result of each task.
- All the layers build are cached by Docker.

SkillAssure

Docker Images - Demo

SkillAssure

Exercise

- Create a Dockerfile for the following Applications
- Create a Docker Hub Account
- Push the docker image to your Docker Hub Account

Exercise 1 – Python (Simple-Webapp-Color)

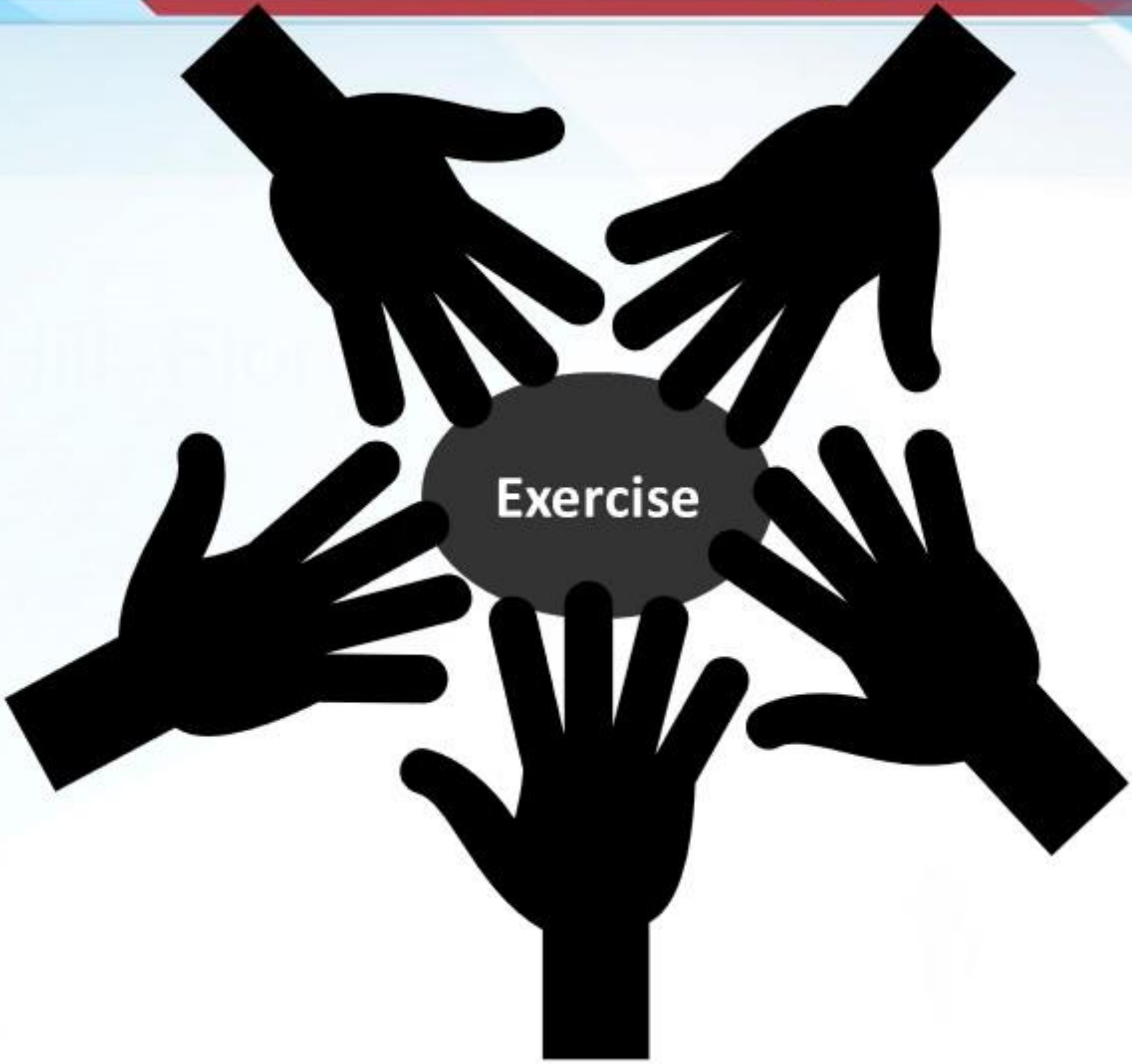
Repository URL

https://github.com/prabhavagrawal/simple-webapp-color-docker

Exercise 2 – JAVA (Time Tracker)

Repository URL

https://github.com/prabhavagrawal/timetracker


Exercise

Docker Command vs Entrypoint

Command vs Entrypoint

- Run a docker container from an ubuntu image

  ```
  docker run ubuntu
  ```

  - It runs an instance of ubuntu image and exits immediately
- List the running containers

  ```
  docker ps
  ```

  - No container running
- List all containers

  ```
  docker ps -a
  ```

  - New container which we ran is in an exited state

SkillAssure

Command vs Entrypoint

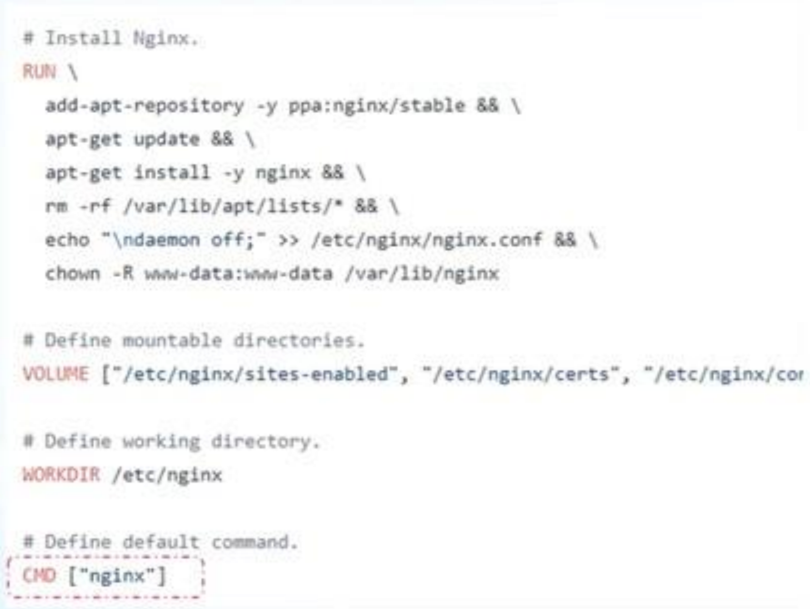
- Unlike virtual machines, containers are not meant to host an operating system.
- Containers are meant to run a specific task or process
  - host an instance of a web server or application server or a database or simply to carry out some kind of computation or analysis.
- Once the task is complete the container exits
- The container only lives as long as the process inside it is alive.
- If the web service inside the container is stopped or crashes the container exits.

- So who defines what process is run within the container?

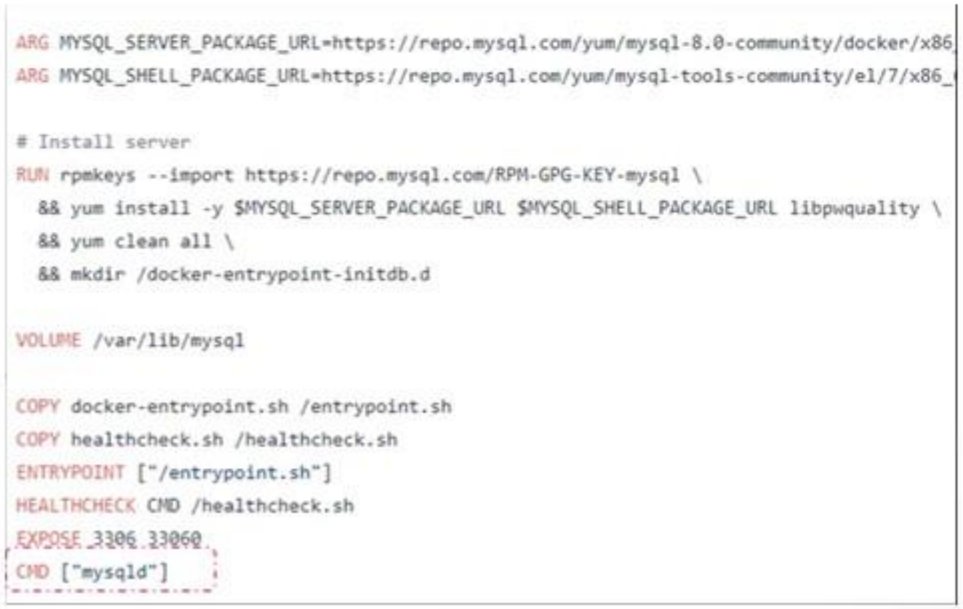
- Let’s look at Dockerfile on Nginx

Command vs Entrypoint

- We see an instruction called CMD
- CMD defines the program that will be run within the container when it starts.
- For the NGINX image, it is the nginx command
- For the MySQL image, it is the mysqld command.



```
# Install Nginx.
RUN \
  add-apt-repository -y ppa:nginx/stable && \
  apt-get update && \
  apt-get install -y nginx && \
  rm -rf /var/lib/apt/lists/* && \
  echo "\ndaemon off;" >> /etc/nginx/nginx.conf && \
  chown -R www-data:www-data /var/lib/nginx

# Define mountable directories.
VOLUME ["/etc/nginx/sites-enabled", "/etc/nginx/certs", "/etc/nginx/cor

# Define working directory.
WORKDIR /etc/nginx

# Define default command.
CMD ["nginx"]
```

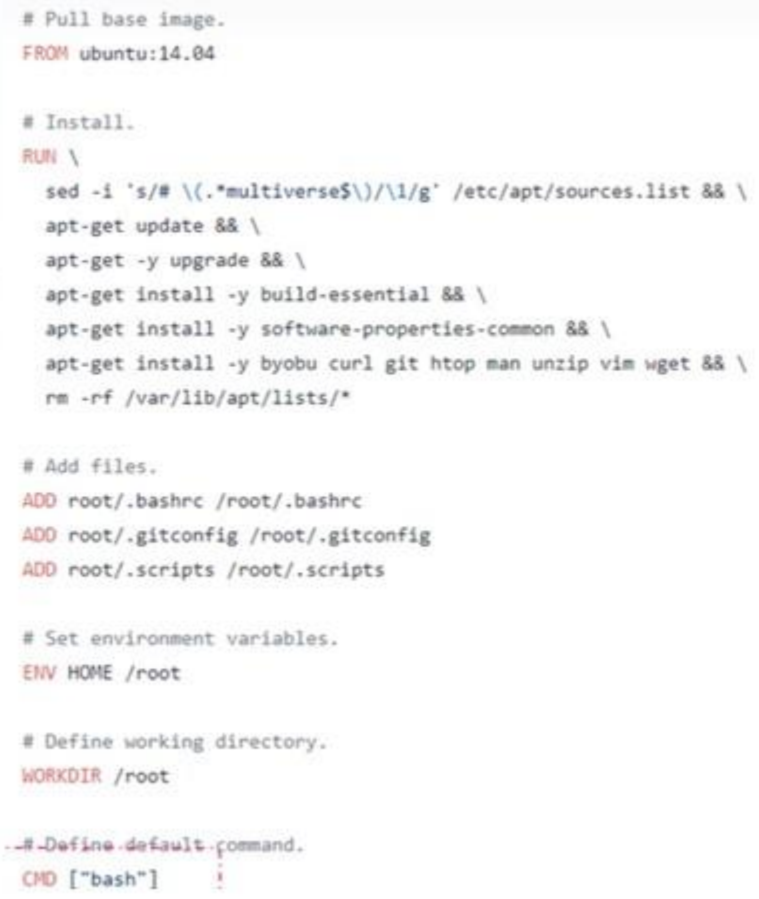


```
ARG MYSQL_SERVER_PACKAGE_URL=https://repo.mysql.com/yum/mysql-8.0-community/docker/x86
ARG MYSQL_SHELL_PACKAGE_URL=https://repo.mysql.com/yum/mysql-tools-community/el/7/x86_

# Install server
RUN rpmkeys --import https://repo.mysql.com/RPM-GPG-KEY-mysql \
  && yum install -y $MYSQL_SERVER_PACKAGE_URL $MYSQL_SHELL_PACKAGE_URL libpwquality \
  && yum clean all \
  && mkdir /docker-entrypoint-initdb.d

VOLUME /var/lib/mysql

COPY docker-entrypoint.sh /entrypoint.sh
COPY healthcheck.sh /healthcheck.sh
ENTRYPOINT ["/entrypoint.sh"]
HEALTHCHECK CMD /healthcheck.sh
EXPOSE 3306 33060
CMD ["mysqld"]
```

Command vs Entrypoint

- We ran a container with a plain Ubuntu operating system.
- Let us look at the docker file for this image
- It uses bash as the default command.
- Bash is not really a process like the web server or database server.
- It is a shell that listens for inputs from a terminal.
- If it cannot find a terminal it exits.

- When we run the Ubuntu container, docker launches the bash program.
- By default Docker does not attach a terminal to a container
- So the bash program does not find the terminal and it exits



```
# Pull base image.
FROM ubuntu:14.04

# Install.
RUN \
  sed -i 's/# \(.*multiverse$\)/\1/g' /etc/apt/sources.list && \
  apt-get update && \
  apt-get -y upgrade && \
  apt-get install -y build-essential && \
  apt-get install -y software-properties-common && \
  apt-get install -y byobu curl git htop man unzip vim wget && \
  rm -rf /var/lib/apt/lists/*

# Add files.
ADD root/.bashrc /root/.bashrc
ADD root/.gitconfig /root/.gitconfig
ADD root/.scripts /root/.scripts

# Set environment variables.
ENV HOME /root

# Define working directory.
WORKDIR /root

# Define default command.
CMD ["bash"]
```

Command vs Entrypoint

- Specify a different command to start the container

```
docker run ubuntu [COMMAND]
docker run ubuntu sleep 5
```

  - It overrides the default command specified within the image
  - When the container starts it runs the sleep program waits for 5 seconds and then exits

- How do we make this change permanent?
- Let's say you want the image to always run the sleep command when container starts

- We need to create our own image. Let's see how

SkillAssure

Command vs Entrypoint

Dockerfile

```
FROM Ubuntu

CMD sleep 5
```

Shell

```
CMD command param1                CMD sleep 5
```

JSON Array

```
CMD ["command", "param1"]         CMD ["sleep", "5"]
```

- Build image

  `docker build -t ubuntu-sleeper .`
- Build image

  `docker run ubuntu-sleeper`
  - It always sleeps for 5 seconds and exits

- Now when the container runs it will sleep for 5 sec
- There are different ways of specifying the command
  - Shell
  - JSON Array
- Remember when you specify in a JSON array format, the first element in the array should be the executable.

SkillAssure

Command vs Entrypoint

- How do I change the number of seconds it sleeps?
- It is hard coded to 5 seconds
- We want to pass in the number of seconds the container should sleep, and sleep command should be invoked automatically

- **Using Entrypoint**

SkillAssure

Command vs Entrypoint

Dockerfile

```
FROM Ubuntu

ENTRYPOINT ["sleep"]
```

```
docker run ubuntu-sleeper 10
```

Command at Startup: sleep 10

- The entry point instruction is like the command instruction
- You can specify the program that will be run when the container starts
- Whatever you specify on the command line will get appended to the entrypoint

- **CMD -** The command line parameters passed will get replaced entirely
- **ENTRYPOINT -** The command line parameters will get appended

SkillAssure

Command vs Entrypoint

Dockerfile

```
FROM Ubuntu

ENTRYPOINT [“sleep”]
```

```
docker run ubuntu-sleeper
```

Command at Startup: sleep

```
sleep: missing operand
Try 'sleep --help' for more information.
```

- What if I do not specify the number of seconds?
- Then the command at startup will be just sleep
- You will get an error that the operand is missing.

SkillAssure

Command vs Entrypoint

Dockerfile

```
FROM Ubuntu

ENTRYPOINT [“sleep”]

CMD [“5”]
```

```
docker run ubuntu-sleeper
```

Command at Startup: sleep 5

- How do we configure a default value?
- We will make use of both CMD & ENTRYPOINT.
- The CMD instruction will be appended to the ENTRYPOINT instruction.
- The command at startup would be sleep 5.

SkillAssure

Command vs Entrypoint

Dockerfile

```
FROM Ubuntu

ENTRYPOINT [“sleep”]

CMD [“5”]
```

```
docker run ubuntu-sleeper 10
```

Command at Startup: sleep 10

- If we specify any parameters in the command line, then that will override the command instruction.

- **Remember – This will only work if you specify the ENTRYPOINT & CMD instructions in a JSON format**

SkillAssure

Command vs Entrypoint

Dockerfile

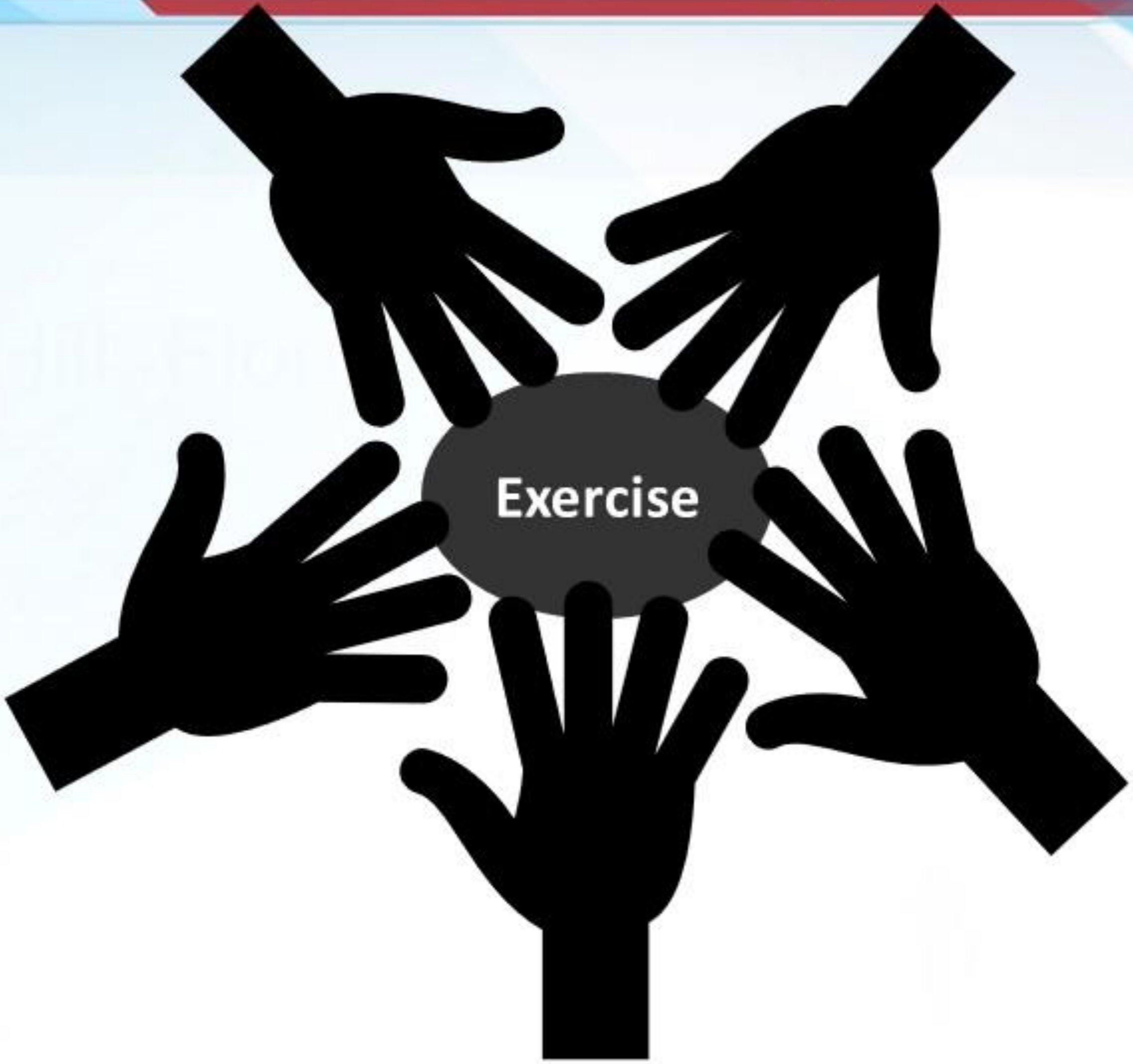
```
FROM Ubuntu

ENTRYPOINT ["sleep"]

CMD ["5"]
```

- If we want to modify the ENTRYPOINT during runtime
- Let's say change sleep to an imaginary sleep2.0 command.

- We can override it by using the entrypoint option in the docker run command.

```
docker run --entrypoint sleep2.0 ubuntu-sleeper 10
```

Command at Startup: sleep2.0 10


Exercise