Dockerfile best practices

Let’s build a web application

About me

The application

We will build a web application in python using flask, a popular web framework for python.

We will also use virtualenv to build our application. Virtualemv enables portability and consistency. ---> **no more “It works on my computer”**

First steps

Best practice: Create ephemeral containers

- A linux base image. Let’s use ubuntu 19.04
- Our files inside
- Executing app.py

Build and run the container:

```
$ docker build -t <imagename:tag> .
$ docker run <imagename:tag>
```

How to run another command in a container?

```
$ docker run -it <image> <command>
```

What’s wrong?

So, we need to install python:

```
$ apt-get update
```

and then

```
$ apt-get install python.
```

Concatenate RUN command and sort alphabetically.

We should add also the pip package so we could install python packages.
But it still doesn’t work ----> requirements.txt

```
$ pip freeze --local > requirements.txt
```

To install packages

```
$ pip install -r requirements.txt
```

Isn’t it a little big?

- The image became too big. Can we start with a lighter base image?
- Install only the necessities.
  - Smaller image
  - Less layers
  - Use COPY, not ADD
  - Build context - use .dockerignore

To summarize

- Create ephemeral containers
- Minimize installed packages
- Decouple applications
- Don’t run under root user
- Minimize the number of layers
- Leverage build cache
- COPY is preferred
- Concatenate and sort RUN
- Use multi build if possible

Thanks!

https://github.com/aviyam/docker-image-demo