Docker Image

CLARUSWAY©
WAY TO REINVENT YOURSELF

Table of Contents

CLARUSWAY©
WAY TO REINVENT YOURSELF

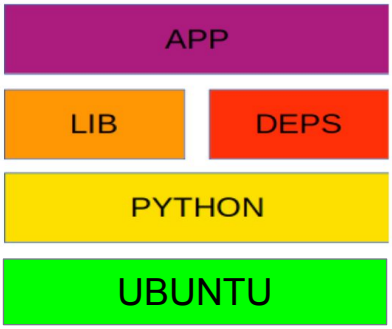
1 What is a Docker image?

Images and Containers

- An image is a read-only template with instructions for creating a Docker container.

What is a Docker image?

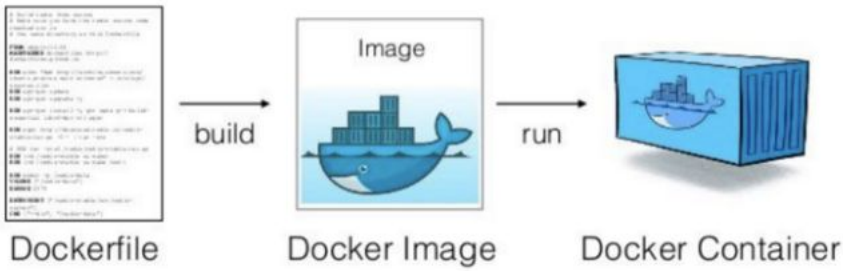
- An image is based on another image, with some additional customization.
- For example, we may build an image which is based on the ubuntu image, but installs the python and our application, as well as the configuration details needed to make our application run.



2 Dockerfile

Dockerfile

- A **Dockerfile** is a text document that define the steps needed to create the image and run it.
- We might create our own images with a **Dockerfile**.
- Each instruction in a Dockerfile creates a layer in the image. When we change the Dockerfile and rebuild the image, only those layers which have changed are rebuilt.
- This is part of what makes images so lightweight, small, and fast, when compared to other virtualization technologies.



Dockerfile

```
FROM ubuntu
ARG VERSION=2.3.3
ENV FLASK_APP=app.py
RUN apt-get update -y && apt-get install python3 -y /
        apt-get install python3-pip -y /
        pip3 install flask=${VERSION}
WORKDIR /myapp
COPY  .  app.py
CMD flask run
```

1. Start from a base OS or another image
2. Install all dependencies
3. Copy source code
4. Specify command

3 Dockerfile Instructions

Dockerfile Instructions

Instruction	Description
FROM	The FROM instruction initializes a new build stage and sets the *Base Image* for subsequent instructions. As such, a valid Dockerfile must start with a FROM instruction.
RUN	The RUN instruction will execute any commands in a new layer on top of the current image and commit the results. The resulting committed image will be used for the next step in the Dockerfile.
CMD	The main purpose of a CMD is to provide defaults for an executing container. There can only be one CMD instruction in a Dockerfile. If you list more than one CMD then only the last CMD will take effect.
EXPOSE	The EXPOSE instruction informs Docker that the container listens on the specified network ports at runtime. You can specify whether the port listens on TCP or UDP, and the default is TCP if the protocol is not specified.

Dockerfile Instructions

Instruction	Description
ENV	The ENV instruction sets the environment variable <key> to the value <value>.
ADD	The ADD instruction copies new files, directories or remote file URLs from <src> and adds them to the filesystem of the image at the path <dest>.
COPY	The COPY instruction copies new files or directories from <src> and adds them to the filesystem of the container at the path <dest>.
ENTRYPOINT	An ENTRYPOINT allows you to configure a container that will run as an executable.
VOLUME	The VOLUME instruction creates a mount point with the specified name and marks it as holding externally mounted volumes from native host or other containers.

Dockerfile Instructions

Instruction	Description
WORKDIR	The WORKDIR instruction sets the working directory for any RUN, CMD, ENTRYPOINT, COPY and ADD instructions that follow it in the Dockerfile. If the WORKDIR doesn't exist, it will be created even if it's not used in any subsequent Dockerfile instruction.
ARG	The ARG instruction defines a variable that users can pass at build-time to the builder with the docker build command using the --build-arg <varname>=<value> flag. If a user specifies a build argument that was not defined in the Dockerfile, the build outputs a warning.
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 an infinite loop and unable to handle new connections, even though the server process is still running.

4 Docker Image Naming

Docker Image Naming

docker.io/ubuntu:latest → Official Images

Registry URL — Repository — Tag

docker.io/clarusway/flask-app:1.0

Registry URL — Repository — Tag

Docker Image Naming

docker.io/clarusway/flask-app:1.0

Registry URL — Repository — Tag

docker.io/clarusway/flask-app:2.0

Registry URL — Repository — Tag

THANKS!

Any questions?

Dockerfile Best Practices

- **Keep Containers Single-Purpose**:
  Aim to keep each container focused on a single purpose or responsibility. This promotes modularity, maintainability, and easier troubleshooting.
- **Use Official Base Images such as the Docker Hub official repositories**.
  Official images are well-maintained, regularly updated, and usually have a smaller size compared to custom-built base images.
- **Minimize the Number of Layers**:
  Each instruction in a Dockerfile creates a new layer in the image, and too many layers can increase the image size and impact performance. Consolidate multiple commands into a single RUN instruction, if possible, to reduce the number of layers.
- **Leverage Caching**:
  Docker caches intermediate layers during the build process. Place frequently changing instructions towards the end of your Dockerfile.
- **Use a base image that is as small as possible to reduce the size of your image.**