Docker -- Dockerfile

VISHWANATH M S

VISHWACLOUDLAB.COM

DockerFile Basics

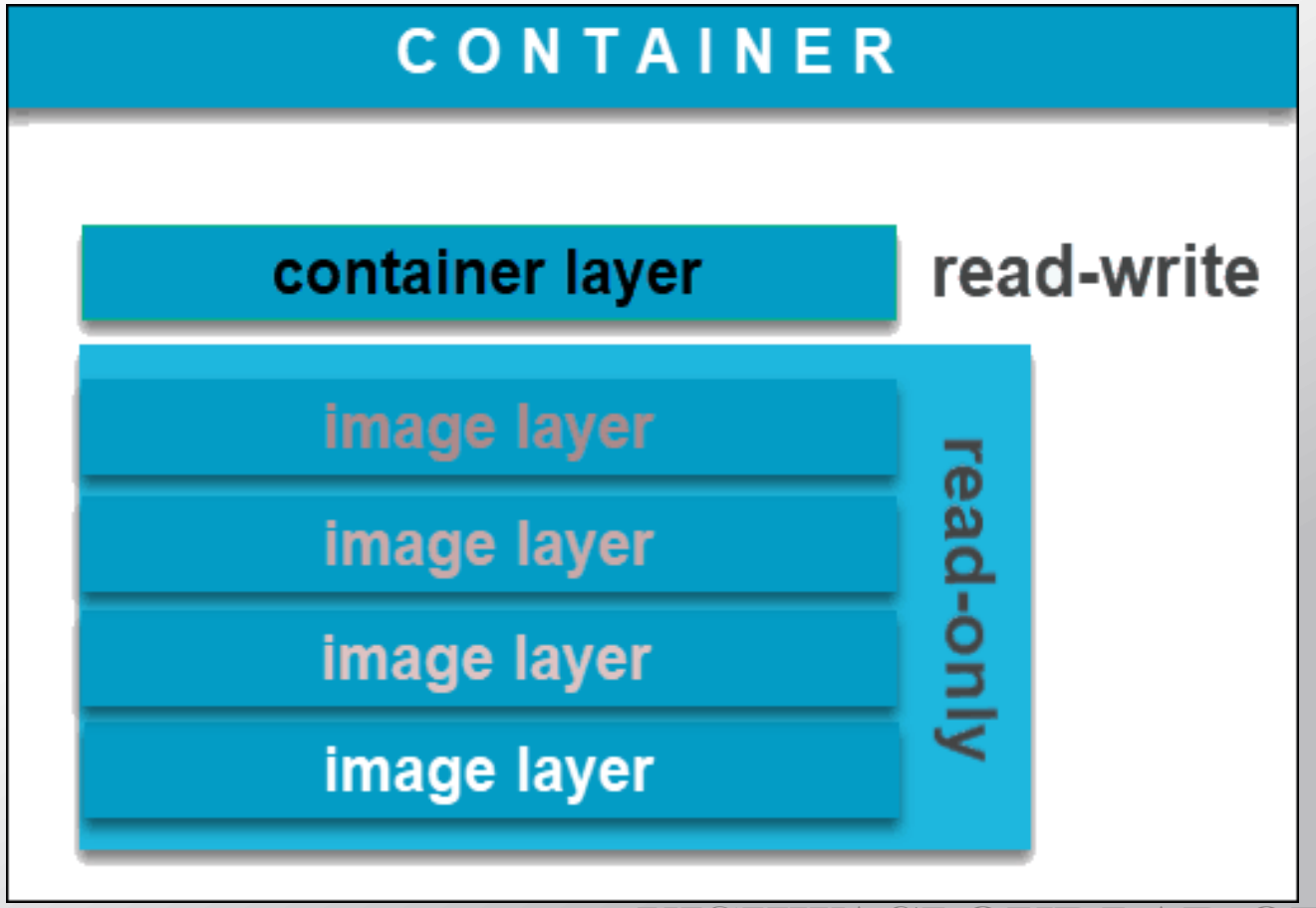
- Docker images are built from a base image.
- Base images are built up using simple instructions such as
  - **Run** a command.
  - **Add** a file or directory
  - Create an **Env** Variable
  - What process to **run when launching a container** from this image.

Docker Images

- Docker images are read only templates

  - Each ***Image*** consists of a series of layers using the union file system.

  - When ever an image is changed a ***new layer*** is created.



Building an Image

- **docker build . -t vishwacloudlab/tomcat-cust01:v1**

- Dockerfile contains set of instructions, to inform Docker how to build the image.

- Docker daemon does actual build process.
- We can reduce build overhead by using ***.dockerignore*** files

Dockerfile → FROM

- **FROM →**
- This sets the Base Image for subsequent instructions.
- Every Dockerfile must have FROM, to start from the Base Image.
- **Eg: -- FROM tomcat:latest**
- This would check if the image is available in the local machine, if not it would download from the central docker hub either from Private or Public.

Dockerfile → ENV & EXPOSE

- **ENV →**
- This is an additional option for providing required environment variables.

- **Eg: -- ENV TOMCAT_VERSION 8.1.0**

- **EXPOSE →**
- This informs Docker that the container will listen on the specified network ports at runtime, to interconnect containers using ***links.***
- **Eg: -- EXPOSE 8080**

Dockerfile → RUN

- **RUN →**
- This 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.
- **Eg: -- RUN yum update && yum install –y httpd wget**
- This would update and install “httpd” & “wget” on the Base image on the **new layer.**

Dockerfile → ADD & COPY

Details	ADD	COPY
Can copy file from URL to the DST container folder	Yes	No
Copies and extract compressed files	Yes	No
Duplicate file/Dir in a specified location in their format	Yes	Yes

- **COPY →** **Eg: -- COPY /source/file/path /dst/path**

Dockerfile → WORKDIR

- **WORKDIR →**
- This sets the working directory for any RUN, CMD, ENTRYPOINT, & COPY that follow it.
- We could use it multiple times in the same Dockerfile.
- The relative path provided would be relative to the path of the previous WORKDIR instruction.
- **Eg: -- WORKDIR $MY_HOME**
- **WORKDIR /app1 → absolute path**
- **WORKDIR fold1 → relative path**

Dockerfile → CMD

- **CMD →**

- CMD is used to identify what should be executed when the container comes up.
- If a Dockerfile has multiple CMDs, it only applies the instructions from the last one.

- **Eg: --**
- **CMD [“script.sh”, “run”]**
- **CMD [“python”, “app1.py”]**

Dockerfile → ENTRYPOINT

- **ENTRYPOINT →**

- ENTRYPOINT is used to identify what should be executed when the container comes up.
- However, the user has the option to override either of these values at run time.
- **Eg: --**
- **docker run --entrypoint ping localhost**

Dockerfile → ENTRYPOINT vs CMD

- **CMD →**
- When we want the user of the image to have the flexibility to run whichever executable they choose when starting the container

- **ENTRYPOINT →**
- It Should be used in scenarios where we want the container to behave exclusively as if it were the executable it's wrapping.
- When we don't want or expect the user to override the executable

Custom image on the central repo

- As best practice we will need to upload the image to central repo.
  - Internal docker registry for private images.
  - Docker hub for Public and private images but stored in the cloud.

Docker push <<docker hub id>>/cust-01:v1

Dockerfile example

```
FROM httpd:latest
ADD . /code
WORKDIR /code
RUN echo “hello world”
ENV NAME World
EXPOSE 80
CMD [“script1.sh", “run"]
```