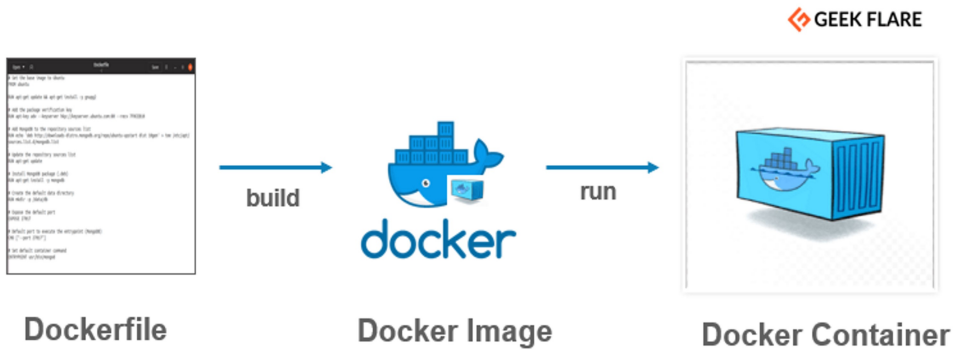
DOCKERFILE by Rahul Jadhav

15 June 2022 23:48

Docker file

Docker file basically a script that you can write and build into a image. The created image can be run to create container.



Docker file format

1) **FROM**
   Syntax : FROM < base image >
   Ex. FROM centos

   This is a instruction that inform docker about the base image that is to be used for container creation.

2) **RUN**
   Syntax : RUN < command that to be run >
   Ex. RUN apt-get update
   RUN mkdir /tmp/mydata -p
   RUN apt-get install tree -y

   This instruction used for to run specific command that you want to run in the container during its creation.
   Writing number of RUN command is not good practice as it increases layer of images. Resulted in heavier image.

3) **LABEL**
   Syntax : LABEL <key> = <value>
   Ex. LABEL name = mycentos
   LABEL email = mycentos@gmail.com

   This instruction is used to add metadata to your image.

4) **MAINTAINER**
   Syntax : MAINTAINER (name)
   Ex. MAINTAINER Rahul Jadhav

   This instruction set author field or define the author for generated image.

5) **WORKDIR**
   Syntax : WORKDIR (directory name)
   Ex: WORKDIR /tmp/project -p

   This instruction used for to define working directory of docker container at any given time. Commands like RUN, CMD. ADD, COPY all command will be executed in that specified directory. If WORKDIR has not written in docker file, it will automatically created by docker complier.

6) **COPY**

Syntax : COPY (source) (Destination)

Ex. COPY text.txt /tmp/project

This command copies file from local source location to the destination in docker container.

7) **ADD**

Syntax : ADD (source) (destination)

Ex. ADD dock.tar.gz /tmp/project

The ADD command is used to copy files/directories into a Docker image. It can copy data in three ways:

- Copy files from the local storage to a destination in the Docker image.
- Copy a tarball from the local storage and extract it automatically inside a destination in the Docker image.
- Copy files from a URL to a destination inside the Docker image

8) **CMD**

Syntax : CMD ["executable", "parameters1", "parameter2"]

Ex. CMD ["yum" , "install", "git", "-y"]

CMD ["echo", "Hello world"]

- Similar function as a RUN command, but it gets executed only after the container is instantiated.
- The CMD specifies the instruction that is to be executed when a Docker container starts.
- The main purpose of the CMD command is to launch the software required in a container.

Multiple CMD commands.

In principle, there should only be *one* CMD command in your Dockerfile. When CMD is used multiple times, only the last instance is executed.

Overriding of CMD

- A CMD command can be overridden by providing the executable and its parameters in the docker run command.
- However, ENTRYPOINT cannot be overridden by docker run. Instead, whatever is specified in docker run will be appended to ENTRYPOINT – this is not the case with CMD.

9) **ENTRYPOINT**

Syntax : ENTRYPOINT [""executable", "parameters1", "parameter2"]

Ex. : ENTRYPOINT ["yum" , "install", "git", "-y"]

ENTRYPOINT ["echo", "Hello world"]

The both CMD & ENTRYPOINT specify program that execute when container start running, but

CMD - The argument which is given while creating container overridden by CMD or replaced by CMD.

ENTRYPOINT - The argument which is given while creating container appended by ENTRYPOINT or get added.

10) **EXPOSE**

Syntax : EXPOSE (PORT) or EXPOSE (PORT)/tcp

Ex . EXPOSE 22 80 8080

The EXPOSE instruction expose a particular port with specified protocol inside a docker container.

Expose vs. Publish

The user has three options when running a Docker container:

- Neither expose nor publish
- Only expose
- Both expose and publish

Neither expose nor publish

The user restricts the communication with the container from within the container only. This approach is adopted by the user if the container at hand is running isolated from all the other containers.

Only expose

The user restricts the communication with the container from within the Docker. It helps to establish inter-container communication in Docker.

Both expose and publish

After publishing (-p) the container, the user allows communication with the container from outside the Docker.

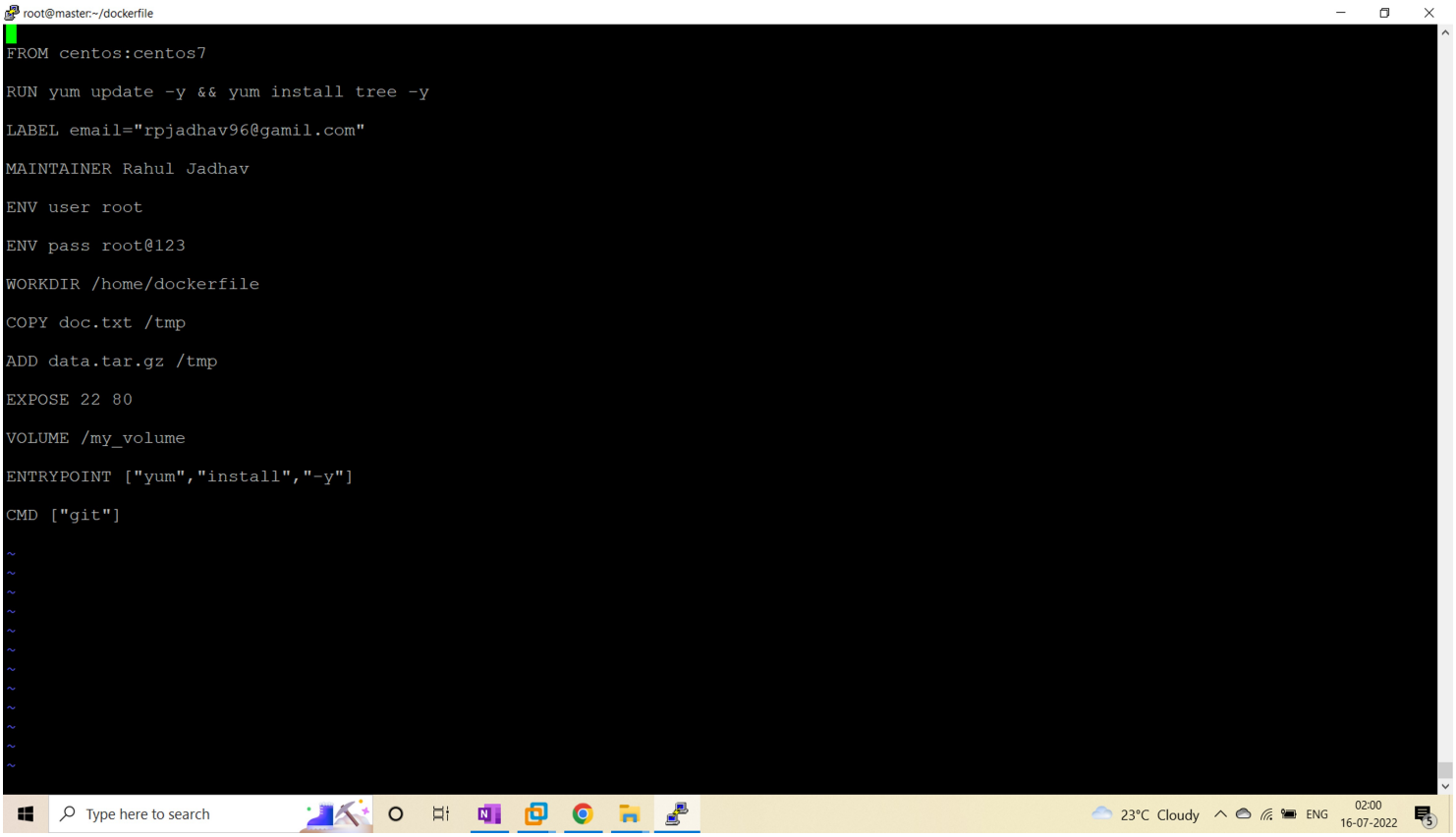
To publish a container, **use the -p flag with the run command.**

11) VOLUME

Syntax : VOLUME [" /path"]
Ex. VOLUME ["/data"]
VOLUME["/backup"]

Volumes in Docker is a mechanism which enables us to generate data into host machine directory.
Volumes are managed mainly by docker daemon, on the other hand, bind mounts is created on the user managed directory on the host machine.

Note :
You can write anything before creating volumes. Creating volumes at the last will enable you to do what you want. So please keep in mind that the changes done after the volume creation will be discarded.



root@master:~/dockerfile

```
[root@master dockerfile]# vi dockerfile
[root@master dockerfile]# docker image build -t centos:9 .
Sending build context to Docker daemon  18.43kB
Step 1/13 : FROM centos:centos7
 ---> eeb6ee3f44bd
Step 2/13 : RUN yum update -y && yum install tree -y
 ---> Using cache
 ---> 14b2ae7ee5f7
Step 3/13 : LABEL email="rpjadhav96@gamil.com"
 ---> Using cache
 ---> a5e0b8db2136
Step 4/13 : MAINTAINER Rahul Jadhav
 ---> Using cache
 ---> b2211fd73a93
Step 5/13 : ENV user root
 ---> Using cache
 ---> 5ea086f90f0c
Step 6/13 : ENV pass root@123
 ---> Using cache
 ---> 1762d011c4dd
Step 7/13 : WORKDIR /home/dockerfile
 ---> Using cache
 ---> 028baa0bb47d
Step 8/13 : COPY doc.txt /tmp
 ---> Using cache
 ---> 61a8be0400db
Step 9/13 : ADD data.tar.gz /tmp
 ---> Using cache
 ---> cbd74cdde82c
Step 10/13 : EXPOSE 22 80
 ---> Using cache
 ---> 1acbce8ef9c9
Step 11/13 : VOLUME /my_volume
 ---> Using cache
 ---> 4edcc5da1f50
Step 12/13 : ENTRYPOINT ["yum","install","-y"]
 ---> Using cache
 ---> d8dd562b5d92
Step 13/13 : CMD ["git"]
 ---> Using cache
 ---> 717b89832fcc
Successfully built 717b89832fcc
Successfully tagged centos:9
[root@master dockerfile]#
```