

Chapter-6 (Dockerfile)

01 August 2022 02:58

Dockerfile is a script, composed of various commands (instructions) and arguments listed successively to automatically perform an action on a base image in order to create (or form) a new one.

Dockerfile comments:

```
# Dockerfile Starts
FROM openjdk:17-alpine
# Add class file to working directory
ADD HelloWorld.class HelloWorld.class
# Execute class file by firing
CMD ["java","HelloWorld"]
# Dockerfile Starts
```

Syntax to write instruction and its argument within a Dockerfile is:

INSTRUCTION arguments

- Instruction can be given in lowercase or uppercase letters. But to differentiate from the instructions and arguments we use uppercase letters.
- Docker runs instructions in a Dockerfile in order. A Dockerfile must start (ARG can exist) with a **FROM** instruction. **The FROM instruction specifies the Base Image from which you are building.**
- A Docker image consists of read-only layers each of which represents a Dockerfile instruction. The layers are stacked and each one is a delta of the changes from the previous layer.
- When we run an image and generate a container, we add a new writable layer (the “container layer”) on top of the underlying layers. All changes made to the running container, such as writing new files, modifying existing files, and deleting files, are written to this thin writable container layer.

Understand Instructions :

- **FROM** : It defines the base image to use to start the build process. This instruction is used to set the base image for subsequent instructions. It is mandatory to set this in a Dockerfile. You can use it any number of times though.

```
FROM openjdk:17-alpine
ADD HelloWorld.class HelloWorld.class
CMD ["java","HelloWorld"]
```

- **MAINTAINER**: This is a non-executable instruction used to indicate the author of the Dockerfile. It should come nonetheless after FROM

```
FROM openjdk:17-alpine
MAINTAINER CodeHop
ADD HelloWorld.class HelloWorld.class
CMD ["java","HelloWorld"]
```

- **ADD**: This instruction is used to add files from local as well from remote to current image.

```
FROM openjdk:17-alpine
MAINTAINER CodeHop
ADD HelloWorld.class HelloWorld.class
CMD ["java","HelloWorld"]
```

■ → Local directory
■ → Container directory

- **RUN**: This instruction allows us to execute a command on top of an existing layer and create a new layer with the result of command execution. This is what runs within the container at build time.
 - ◆ Use case let us consider we are writing into a file using java.
 - ◆ Java writes the file in log directory
 - ◆ But log directory do not exist inside the docker container when created
 - ◆ We can achieve this using RUN command

- Sample Java code :

```
import java.io.*;
class FileWriterLineByLine{
    public static void main(String[] args) throws Exception {
        File fout = new File("log/out.txt");
        FileOutputStream fos = new FileOutputStream(fout);
        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(fos));
        for (int i = 1; i <=10; i++) {
            bw.write("Printing line "+i);
            Thread.sleep(10000);
            bw.newLine();
        }
        bw.close();
    }
}
```

FileWriterLineByLine.java

- Sample Dockerfile :

```
FROM openjdk:17-alpine
MAINTAINER CodeHop
RUN mkdir log
ADD FileWriterLineByLine.class FileWriterLineByLine.class
CMD ["java","FileWriterLineByLine"]
```

- To run a container in detached mode (run a container without waiting it to get up) we use **-d** as input to run command

```
docker run -d -name filewriter filewriter:v1
```

- To run a container in interactive mode (run a container and see the output inside container until container stops) we use **-it** as input to run command

```
docker run -it -name filewriter filewriter:v1
```

- To run a command inside a container while is container is up & running in detached mode we use **exec** (Execute) command

```
docker exec -it container_id sh
```

- We can also use **RUN** command of Dockerfile to install or update a binary which is not present inside a docker container
 - ◆ For case lets say we want to update the Linux distribution before starting out build inside container for the above application
- Sample Dockerfile :

```
FROM openjdk:17-alpine
MAINTAINER CodeHop
RUN apk update && mkdir log
ADD FileWriterLineByLine.class FileWriterLineByLine.class
CMD ["java","FileWriterLineByLine"]
```