# DockerFile

## Introduction

### What are dockerfiles?

Dockerfile is a small program with easy to understand syntax to create a Docker image. It includes all the required dependencies and components to run an application.

The command to build the dockerfile is:

**docker build –t *image-name* . 🡐 The dot specifies the current location**

The -t is used for tagging the image. The dot specifies the current location. If the dockerfile is in a different location, the path of the dockerfile has to be mentioned.

**The Docker Workflow** resolves “works on my machine” issue by setting up the development environment across all other environments without any hassle.

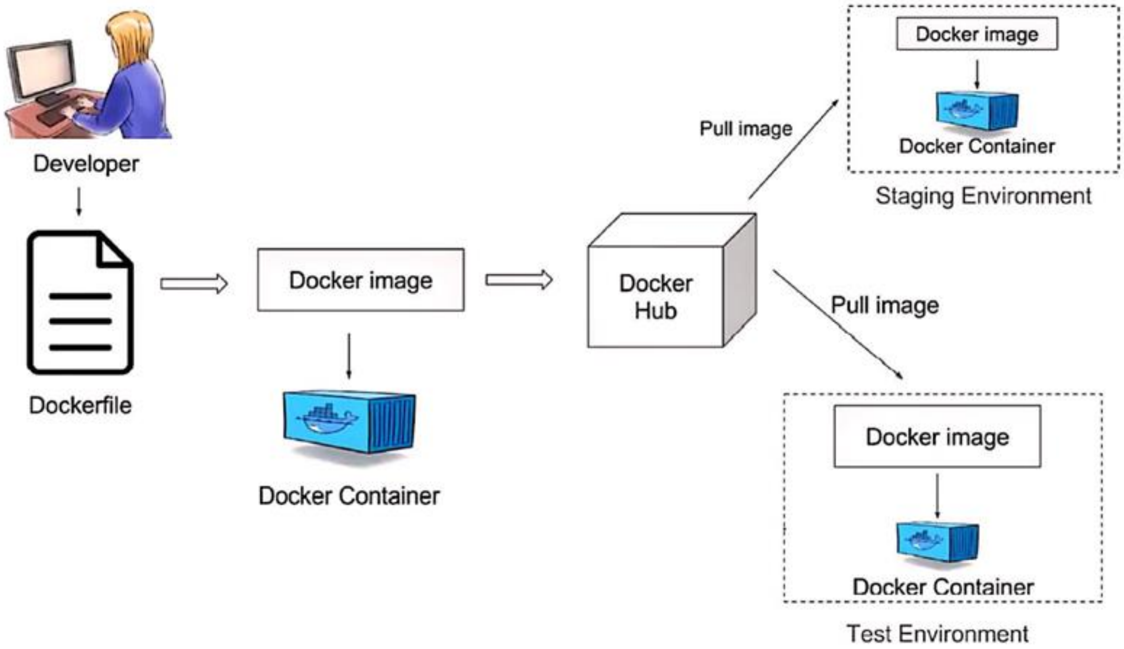


*Figure 1: The Docker Workflow [1]*

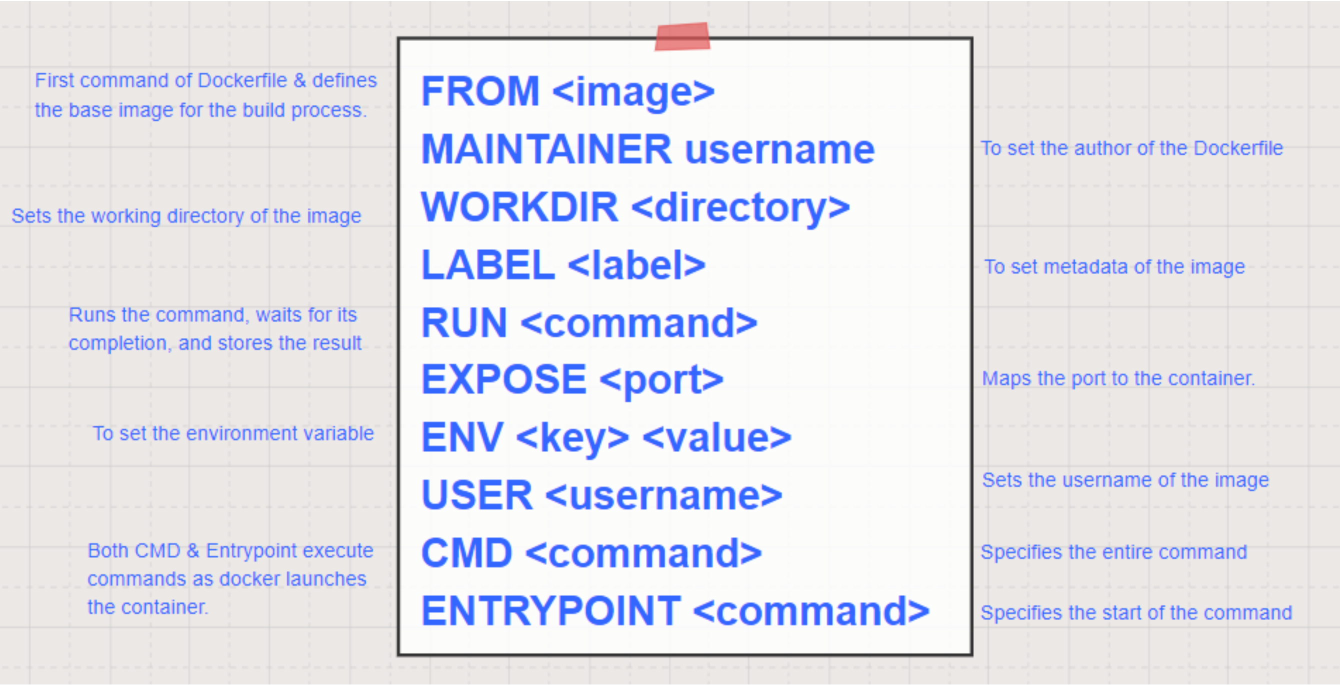
**Producing the next image with each step -** Every line in a dockerfile creates a new image. It takes the image from the previous line, run containers from it, executes the required tasks in the containers, and build new images from the container. The previous image or state of the previous line is unchanged.

**Caching -** To speed up the build process, Docker implements caching. This saves a lot of time in building the images. The parts that are frequently changed are added to the end of the dockerfile.

**Not a Shell Script -** Dockerfiles are different from shell scripts. The process on each line will not be executed on the next line.

## Dockerfile Format

A dockerfile must start with FROM statement. It must include either CMD or ENTRYPOINT. The other statements are optional and depends on the application.



```
FROM <image>
MAINTAINER username
WORKDIR <directory>
LABEL <label>
RUN <command>
EXPOSE <port>
ENV <key> <value>
USER <username>
CMD <command>
ENTRYPOINT <command>
```

First command of Dockerfile & defines the base image for the build process.

To set the author of the Dockerfile

Sets the working directory of the image

To set metadata of the image

Runs the command, waits for its completion, and stores the result

Maps the port to the container.

To set the environment variable

Sets the username of the image

Both CMD & Entrypoint execute commands as docker launches the container.

Specifies the entire command

Specifies the start of the command

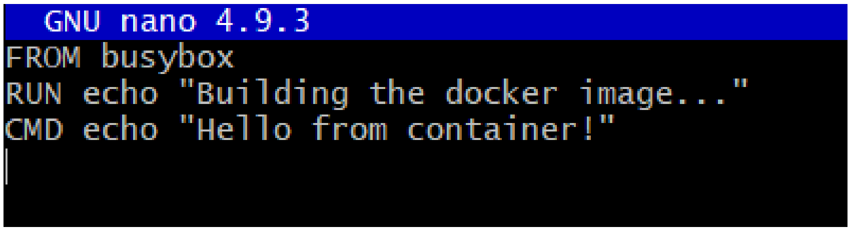
*Figure 2: Dockerfile Sample Format*

## Implementation

### a) Download Busybox

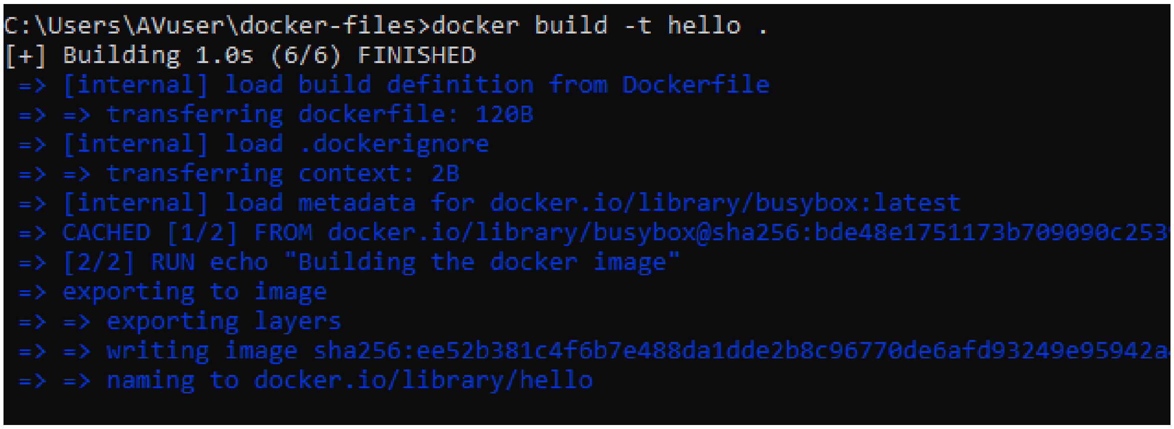
1. Start by creating a file named Dockerfile (Note: Dockerfile starts with uppercase D).

2. Type the content as shown below to create an image. Busybox is a simple image with command line utility. The RUN command will be executed while building the image. The CMD statement will be executed after launching the container.



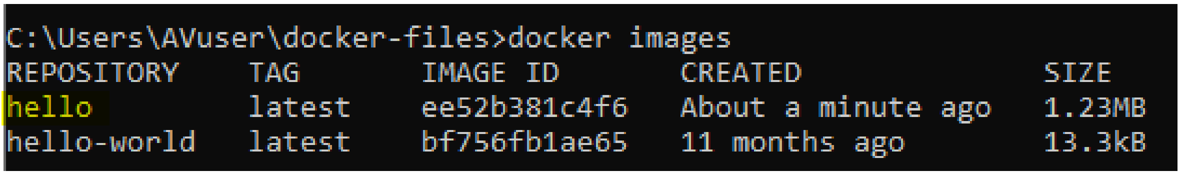
```
  GNU nano 4.9.3
FROM busybox
RUN echo "Building the docker image..."
CMD echo "Hello from container!"
```

3. To run the dockerfile, type the command “docker build –t image-name .”.



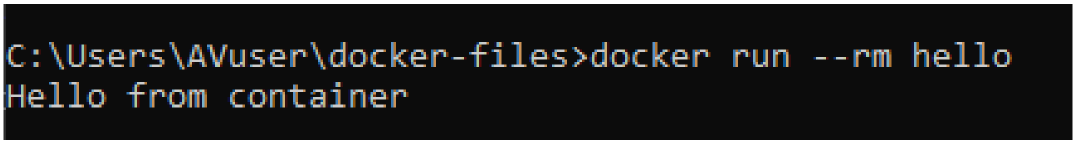
```
C:\Users\AVuser\docker-files>docker build -t hello .
[+] Building 1.0s (6/6) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 120B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/busybox:latest
 => CACHED [1/2] FROM docker.io/library/busybox@sha256:bde48e1751173b709090c253
 => [2/2] RUN echo "Building the docker image"
 => exporting to image
 => => exporting layers
 => => writing image sha256:ee52b381c4f6b7e488da1dde2b8c96770de6afd93249e95942a
 => => naming to docker.io/library/hello
```

4. To view the created image, type the command “docker images”. The created image “hello” is listed in the image list.



```
C:\Users\AVuser\docker-files>docker images
REPOSITORY     TAG       IMAGE ID       CREATED              SIZE
hello          latest    ee52b381c4f6   About a minute ago   1.23MB
hello-world    latest    bf756fb1ae65   11 months ago        13.3kB
```

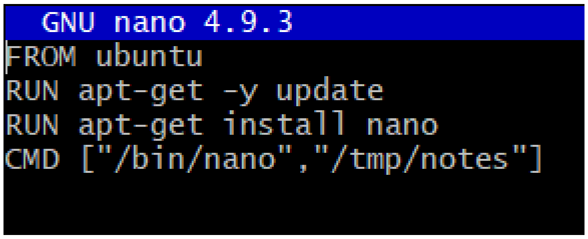
5. To build container from the image, type “docker run --rm image-name”. The --rm tag is used to clean up the container after its usage.



```
C:\Users\AVuser\docker-files>docker run --rm hello
Hello from container
```

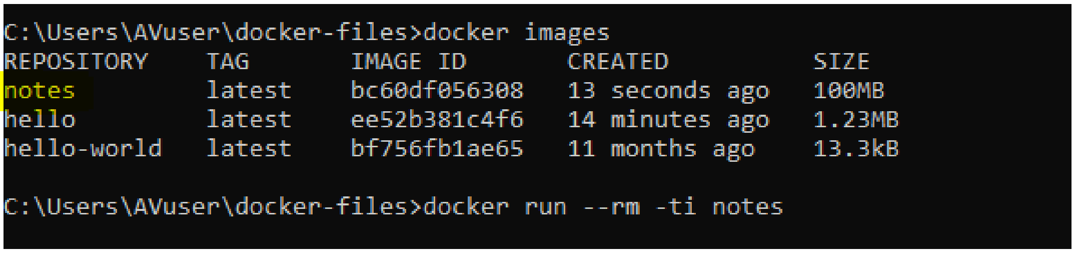
### b) Download Ubuntu image

1. Start by creating a file named Dockerfile. Type the contents as shown below. The FROM statement downloads Ubuntu as the base OS for the container. The RUN statement updates the packages of the image and install nano. The CMD command opens the notes file using the nano command. The CMD command is represented in EXEC form.



```
  GNU nano 4.9.3
FROM ubuntu
RUN apt-get -y update
RUN apt-get install nano
CMD ["/bin/nano","/tmp/notes"]
```

2. After building the image, the created image “notes” will be displayed in the image list. To run the image, type the command “docker run --rm –ti image-name”. The tag –ti is used for the interactive terminal. By running the image, the container is created and the notes file (/tmp/notes) is opened.



```
C:\Users\AVuser\docker-files>docker images
REPOSITORY     TAG       IMAGE ID       CREATED          SIZE
notes          latest    bc60df056308   13 seconds ago   100MB
hello          latest    ee52b381c4f6   14 minutes ago   1.23MB
hello-world    latest    bf756fb1ae65   11 months ago    13.3kB

C:\Users\AVuser\docker-files>docker run --rm -ti notes
```

## References:

[1]"Docker explained in simple terms!!", *Medium*, 2020. [Online]. Available: https://medium.com/dev-genius/docker-explained-in-simple-terms-178748e28e99#:~:text=A%20Dockerfile%20is%20a%20text,on%20that%20in%20a%20moment). [Accessed: 22- Dec- 2020].