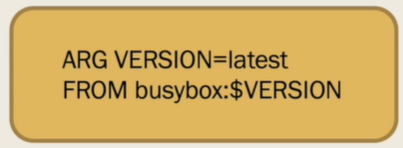
A Dockerfile is a text document that contains instructions (Commands+arguments) on how to build your own images.

**FROM, ARG,LABEL, ENV, RUN,CMD, ENTRYPOINT, VOLUME,EXPOSE,COPY,ADD,WORKDIR**

The [**FROM**](https://docs.docker.com/reference/builder/#from)instruction used to set the base image or parent image

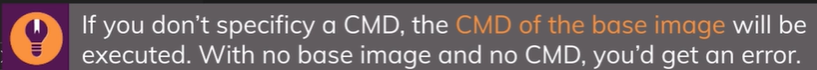
**ARG** instruction is used to pass variables to docker at build time.This is the one which can precede **FROM**



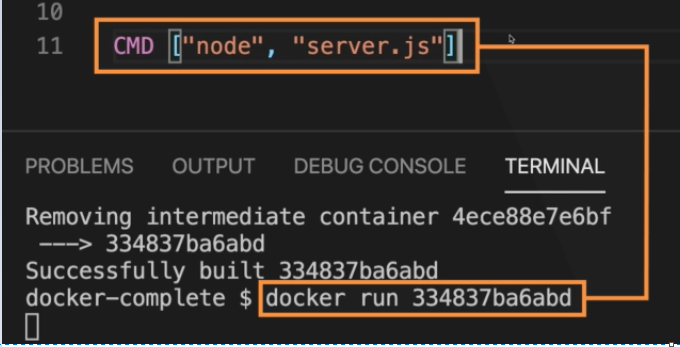
**LABEL** this instruction is used to add metadata to an image like version,description,maintainer information ,application information etc

**ENV** instruction is used to set environmental variables

**CMD** -- This instruction is used to execute a command at runtime when the container is executed. while launching the container, you can override the default CMD by providing it at the command line ( Always it’ll be the last instruction and will be executed when container is created)



Life of the container depends on the CMD (For ex :: in the below snippet when we create a container ,the container will keep on running because CMD starts the node server which is on-going process )



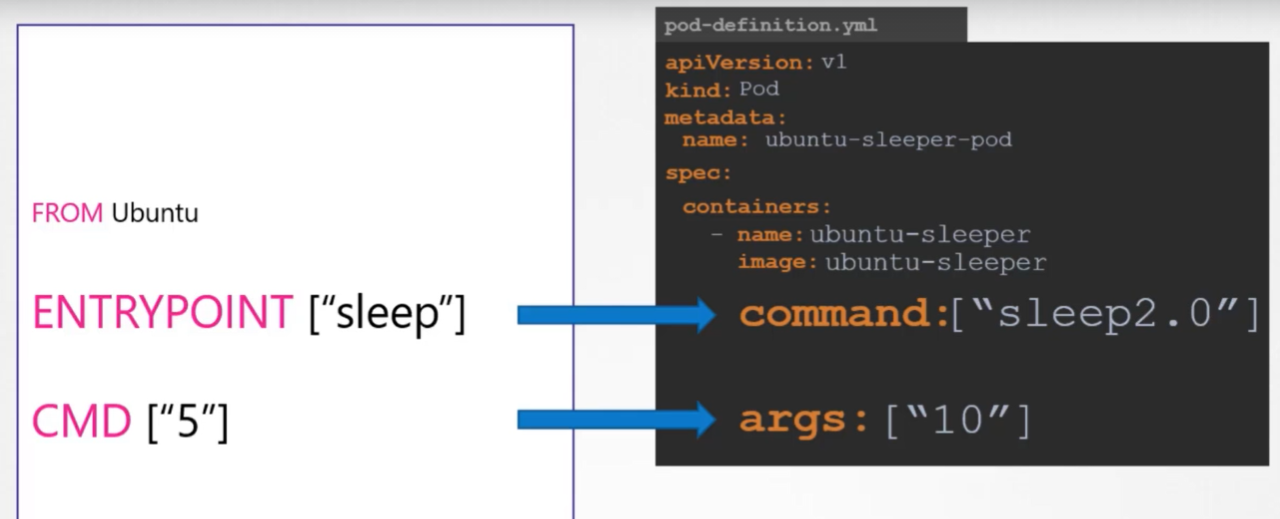
If a Dockerfile has multiple CMDs, it only applies the instructions from the last one.

The **ENTRYPOINT** instruction can also be used to execute commands at runtime for the container. But we can be more flexible with the ENTRYPOINT command. And it Is also used specify the default app that you want to run.

You cannot [override an ENTRYPOINT](https://phoenixnap.com/kb/docker-run-override-entrypoint) while starting a container unless you add the **--entrypoint** flag.

Note :: There are many situations in which combining CMD and ENTRYPOINT would be the best solution for your Docker container. In such cases, the **executable is defined with ENTRYPOINT**, while **CMD specifies the default parameter**.

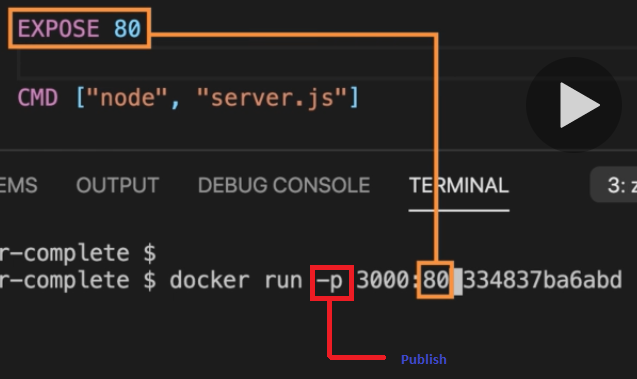
* **ENTRYPOINT ["echo”]**
* **CMD ["Hello World"]**



 A [**RUN**](https://docs.docker.com/reference/builder/#run)  --This instruction takes commands as its arguments and run ;it basically instructs docker to run particular cmds on the base image .

Also this is used to execute any command in the new layer on top your current image.

**EXPOSE** Instruction indicates on which port the container will be listening on

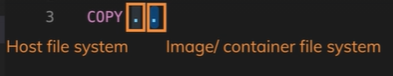


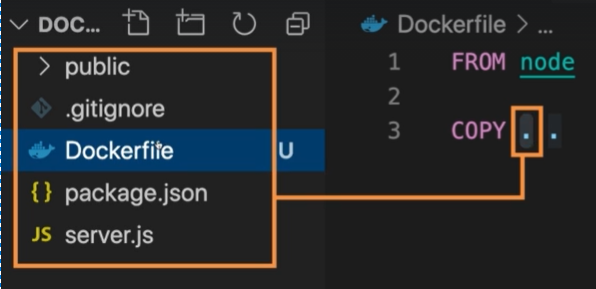
**USER** --The USER instruction sets the user name or UID to use when running the image.

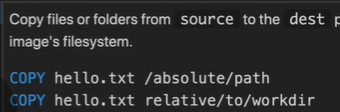
**COPY and ADD**  instructions both  are used for similar purposes. They let you copy files/directories from a specific location(Host) into a Docker image.

**COPY** takes in a *src* and *destination*. It lets you copy file or directory from your host into the Docker image itself.

**ADD** instruction is similar to COPY instruction but it has more features like it can copy files/directories from remote url and also it can also copy a tar file inside your container and untar it in one shot.







**WORKDIR instruction ---**sets the working directory whatever the commands after this will be executed in this directory

**VOLUME instruction** :: Used to attach the volume to a container to preserve the data,as contaiers are ephemeral ,the data vanishes ones the container dies.

**ONBUILD:**

**Dockerfile to install nginx:**

FROM ubuntu

LABEL Maitainer=Lokesh

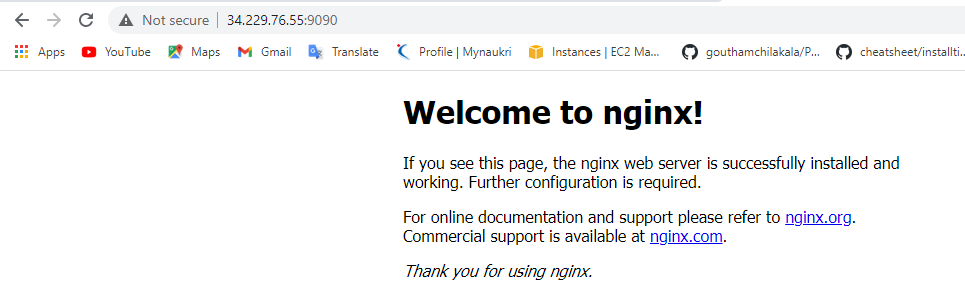
RUN apt-get update &&\

apt-get install nginx -y

COPY index.html /usr/share/nginx/html/

ENTRYPOINT ["/usr/sbin/nginx","-g","daemon off;"]

EXPOSE 80



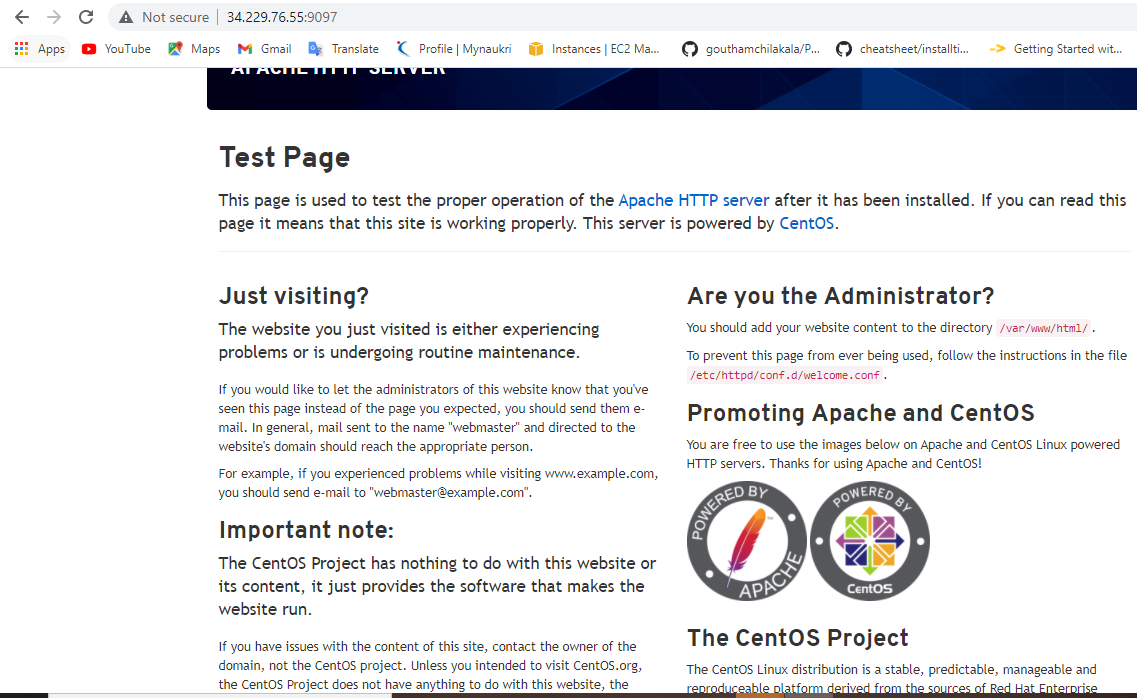
**Dockerfile to install Apache:**

FROM centos

RUN yum install httpd -y

CMD ["/usr/sbin/httpd","-D","FOREGROUND"]

EXPOSE 80



FROM centos

RUN yum install httpd -y

COPY index.html /var/www/html/

CMD ["/usr/sbin/httpd","-D","FOREGROUND"]

EXPOSE 80

**Index.html**

<!DOCTYPE html>

<html>

<head>

<title>Example</title>

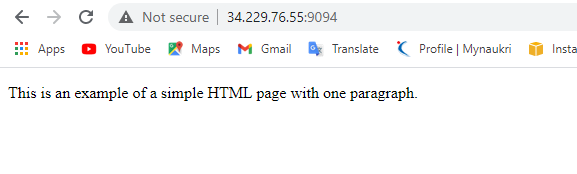
</head>

<body>

<p>This is an example of a simple HTML page with one paragraph.</p>

</body>

</html>



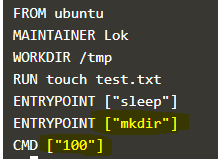
FROM tomcat:8.0-alpine

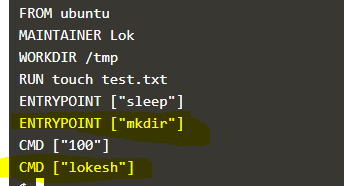
LABEL maintainer=”deepak@softwareyoga.com”

ADD sample.war /usr/local/tomcat/webapps/

EXPOSE 8080

CMD [“catalina.sh”, “run”]





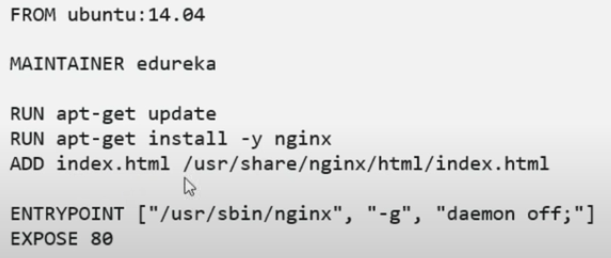
**Bash script to install docker on Ubuntu:**

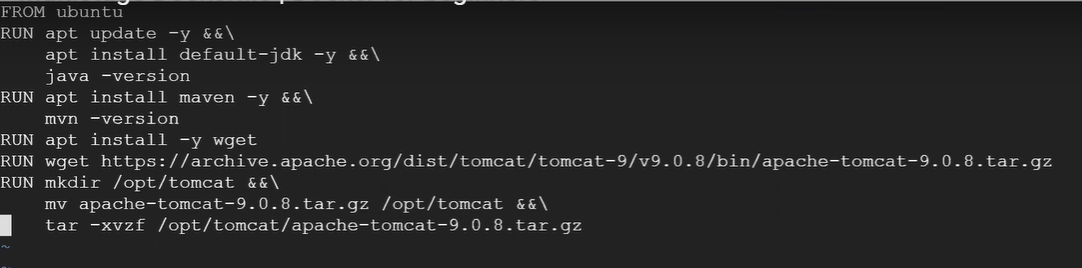
#!/bin/bash

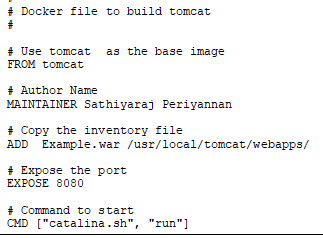
sudo apt update

sudo apt install docker.io -y

sudo service docker start







Docker Compose

version: "3.8"

services:

web:

build: .

ports:

- "5000:5000"

redis:

image: "redis:alpine"

version: "3.8"

services:

web:

build: .

ports:

- "5000:5000"

volumes:

- .:/code

environment:

FLASK\_ENV: development

redis:

image: "redis:alpine"