**Dockerfile Example**

Mentioned below is a Dockerfile example that we have already created, for CentOS with webserver (apache) installed on it.

FROM centos:7

MAINTAINER linuxtechlab

LABEL Remarks=”This is a dockerfile example for Centos system”

RUN yum -y update && \

yum -y install httpd && \

yum clean all

COPY data/httpd.conf /etc/httpd/conf/httpd.conf

ADD data/html.tar.gz /var/www/html/

EXPOSE 80

ENV HOME /root

WORKDIR /root

ENTRYPOINT [“ping”]

CMD [“google.com”]

Parameters

We will now discuss all the parameters mentioned here one by one so that we have an understanding as to what they actually mean,

================================================= ==========

FROM centos:7

FROM tells which base you would like to use for creating your dockerimage. Since we are using Centos:7, its mentioned there. We can use other OS like centos:6 , ubuntu:16.04 etc

MAINTAINER linuxtechlab

LABEL Remarks=”This is a dockerfile example for Centos system”

These both fields MAINTAINER & LABEL Remarks are called labels. They are used to pass information like Maintainer of docker image, Version number, purpose or some other remarks. We can add a number of labels but its recommended to avoid unnecessary labels.

RUN yum -y update && \

yum -y install httpd && \

yum clean all

Now RUN command is responsible for installing or changing the docker image . Here we have asked RUN to update our system & than install apcahe on it. We can also ask it to create a directory or to install some other packages.

COPY data/httpd.conf /etc/httpd/conf/httpd.conf

ADD data/html.tar.gz /var/www/html/

COPY & ADD command almost serve the same purpose i.e. they are used to copy the files to docker image with one difference. Here we have used the COPY command to copy httpd.conf from data directory to the default location of httpd.conf on docker image.

And we than used ADD command to copy a tar.gz archive to apache’s document directory to serve content on the webserver. ***But you might have noticed we didn’t extract it & that’s the one difference ADD & COPY command have, ADD command will automatically extract the archive at the destination*** folder. Also we could have used ADD in place of copy, “ADD data/httpd.conf /etc/httpd/conf/httpd.conf “. Another point regarding add is that we can use remote location with it, like amazon s3 etc.

EXPOSE 80

EXPOSE command will open the mentioned port on the docker image to allow access to outside world. We could also use EXPOSE 80/tcp or EXPOSE 53/udp.

ENV HOME /root

ENV command sets up environment variables, here we have used it to set HOME to /root. Syntax for using ENV is

ENV key value

Some examples of ENV usage are,

ENV user admin, ENV database=testdb, ENV PHPVERSION 7 etc etc.

WORKDIR /root

With WORKDIR, we can set working directory for the docker image. Here it has been set to /root.

ENTRYPOINT [“ping”]

CMD [“google.com”]

ENTRYPOINT & CMD are both used to define executable that should run once docker is up. On ENTRYPOINT, we define an executable & with CMD, we define additional parameters that are required for ENTRYPOINT. Like here, we have used ping with ENTRYPOINT but it requires additional parameter, which we provided with CMD. These both commands are used in conjunction with each other.

We can also used CMD alone with something like CMD [“bash”].

Note:- Not all these parameters are required to pass while creating a Dockerfile, you can only the ones you need.

Apart from the commands discussed above, there are some other commands as well that can be used in the Dockerfile & that are mentioned below,

USER

With USER, we can define the user to be used to execute a command like USER dan. We can specify USER with RUN, CMD or with ENTRYPOINT as well.

ONBUILD

ONBUILD command lets you add a trigger that will be executed at a later time when the current image is being used as a base image for another. For example, we have added our own content for website using the dockerfile but we might not want it to be used for other docker images. So we will add ,

ONBUILD RUN rm -rf /var/www/html/\*

This will remove the contents when the image is being re-purposed.

So these were all the commands that we can use with our Dockerfiles. Mentioned below Dockerfile examples for Ubuntu & Fedora, for reference,