Lab: Building an Image using Dockerfile

Introduction:

The docker build command builds Docker images from a Dockerfile and a "context". A build's context is the set of files located in the specified PATH or URL. The build process can refer to any of the files in the context. For example, your build can use a COPY instruction to reference a file in the context.

DockerFile defines what type of environment need to go inside your container. In this lab, you will learn how to create a simple DockerFile and you can expect that the build of your app defined in this DockerFile behave the same wherever it runs.

Objectives:

- **Create a Dockerfile**
- **Building an Image from a Dockerfile**
- **Run a Container using a newly built image**

1 Let's Create a **directory** with the name of **example.**

```
# mkdir example
```

1.1 Let's get inside the **example** directory.

```
# cd example/
```

Output:

```
[root@docker-engine ~]#cd example/
[root@docker-engine example]#
```

1.2 Let's create a **Dockerfile**.

```
# cat > Dockerfile <<EOF
FROM centos
RUN sed -i 's/mirrorlist/#mirrorlist/g'
/etc/yum.repos.d/CentOS-Linux-*
RUN sed -i
's|#baseurl=http://mirror.centos.org|baseurl=http://vault.cent
os.org|g' /etc/yum.repos.d/CentOS-Linux-*
RUN yum -y install epel-release
RUN yum -y update
RUN yum -y install nginx
RUN mkdir -p /data/storage
WORKDIR /data/storage
ADD index.html /usr/share/nginx/html/index.html
EXPOSE 80/tcp
CMD ["nginx", "-g", "daemon off;"]
EOF
```

Output:

```
[root@docker-engine example]#cat > Dockerfile <<EOF
FROM centos
RUN sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-Linux-*
RUN sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.repos.d/CentOS-Linux-*
RUN yum -y install epel-release
RUN yum -y update
RUN yum -y install nginx
RUN mkdir -p /data/storage
WORKDIR /data/storage
ADD index.html /usr/share/nginx/html/index.html
EXPOSE 80/tcp
CMD ["nginx", "-g", "daemon off;"]
EOF
```

1.3 Let's **create** a custom web page name index.html.

```
# cat > index.html << EOF
Welcome to Docker Learning
EOF
```

Output:

```
[root@docker-engine example]#cat > index.html << EOF
Welcome to Docker Learning
EOF
```

2 Take Dockerfile and use the docker build command to **build** an image.

```
# docker image build -t centos:nginx .
```

Output:

```
[root@docker-engine example]#docker image build -t centos:nginx .
[+] Building 74.7s (14/14) FINISHED
 => [internal] load .dockerignore                                                0.0s
 => => transferring context: 2B                                                  0.0s
 => [internal] load build definition from Dockerfile                             0.0s
 => => transferring dockerfile: 459B                                             0.0s
 => [internal] load metadata for docker.io/library/centos:latest                 4.4s
 => [1/9] FROM docker.io/library/centos@sha256:a27fd8080b517143cbbbab9dfb7c8571c  7.4s
 => => resolve docker.io/library/centos@sha256:a27fd8080b517143cbbbab9dfb7c8571c  0.0s
 => => sha256:a27fd8080b517143cbbbab9dfb7c8571c40d67d534bbdee55bd6c473f432b177  762B / 762B  0.0s
 => => sha256:a1801b843b1bfaf77c501e7a6d3f709401a1e0c83863037fa3aab063a7fdb9dc  529B / 529B  0.0s
 => => sha256:5d0da3dc976460b72c77d94c8a1ad043720b0416bfc16c52c45d4847e53fadb6  2.14kB / 2.14kB  0.0s
 => => sha256:a1d0c75327776413fa0db9ed3adcdbadedc95a662eb1d360dad82bb913f8a1d1  83.52MB / 83.52MB  2.5s
 => => extracting sha256:a1d0c75327776413fa0db9ed3adcdbadedc95a662eb1d360dad82bb913f8a1d1  4.3s
 => [internal] load build context                                                0.0s
 => => transferring context: 64B                                                 0.0s
 => [2/9] RUN sed -i 's/mirrorlist/#mirrorlist/g' /etc/yum.repos.d/CentOS-Linux-*  0.7s
 => [3/9] RUN sed -i 's|#baseurl=http://mirror.centos.org|baseurl=http://vault.centos.org|g' /etc/yum.re  0.4s
 => [4/9] RUN yum -y install epel-release                                         5.7s
 => [5/9] RUN yum -y update                                                      43.0s
 => [6/9] RUN yum -y install nginx                                               9.8s
 => [7/9] RUN mkdir -p /data/storage                                             0.5s
 => [8/9] WORKDIR /data/storage                                                  0.0s
 => [9/9] ADD index.html /usr/share/nginx/html/index.html                        0.0s
 => exporting to image                                                           2.7s
 => => exporting layers                                                          2.6s
 => => writing image sha256:d42879a336f6b056a12355fbb901f3bc0c993a8d1d81dd38eb643b12bdc8f82d  0.0s
 => => naming to docker.io/library/centos:nginx                                  0.0s
```

2.1 Let's **list** the image that has built.

```
# docker image ls
```

Output:

```
[root@docker-engine example]#docker image ls
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
centos       nginx     d42879a336f6   16 minutes ago   706MB
```

2.2 Let's **run** a container using a **newly built image**.

```
# docker container run --name web-nginx -dit centos:nginx
```

Output:

```
[root@docker-engine example]#docker container run --name web-nginx -dit centos:nginx
c8b64dedec8c386d29baed0612f074edce85c82912fbbc507c174a089d8d084e
```

2.3 Let's inspect the container to capture the **ip** address.

```
# docker container inspect web-nginx | grep IPAddress
```

Output:

```
[root@docker-engine example]#docker container inspect web-nginx | grep IPAddress
            "SecondaryIPAddresses": null,
            "IPAddress": "172.17.0.2",
                    "IPAddress": "172.17.0.2",
```

2.4 Let's **access** the containerized webserver by executing the below command.

```
# curl 172.17.0.2
```

Output:

```
[root@docker-engine example]#curl 172.17.0.2
Welcome to Docker Learning
```

3 Cleanup.

3.1 To **remove all** the containers run the below commands.

```
# docker container rm `docker container ls -a -q` -f
```

Output:

```
[root@docker-engine example]#docker container rm `docker container ls -a -q` -f
c8b64dedec8c
```

3.2 To **remove all** the images run the below commands.

```
# docker image rm `docker image ls -q` -f
```

Output:

```
[root@docker-engine example]#docker image rm `docker image ls -q` -f
Untagged: centos:nginx
Deleted: sha256:d42879a336f6b056a12355fbb901f3bc0c993a8d1d81dd38eb643b12bdc8f82d
```

3.3 Let's verify if the **containers** are removed.

```
# docker container ls -a
```

Output:

```
[root@docker-engine example]#docker container ls -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

3.4 Let's **verify** that **images** are removed.

```
# docker image ls
```

Output:

```
[root@docker-engine example]#docker image ls
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
```

3.5 Exit from the example directory

```
# cd ~
```