Demo 1: Basics of Docker

Introduction

The learning objectives of this demonstration are as follows:

- Writing a Dockerfile to create the Nginx docker image from the Ubuntu base image
- Understanding the use of ADD versus COPY and CMD versus ENTRYPOINT

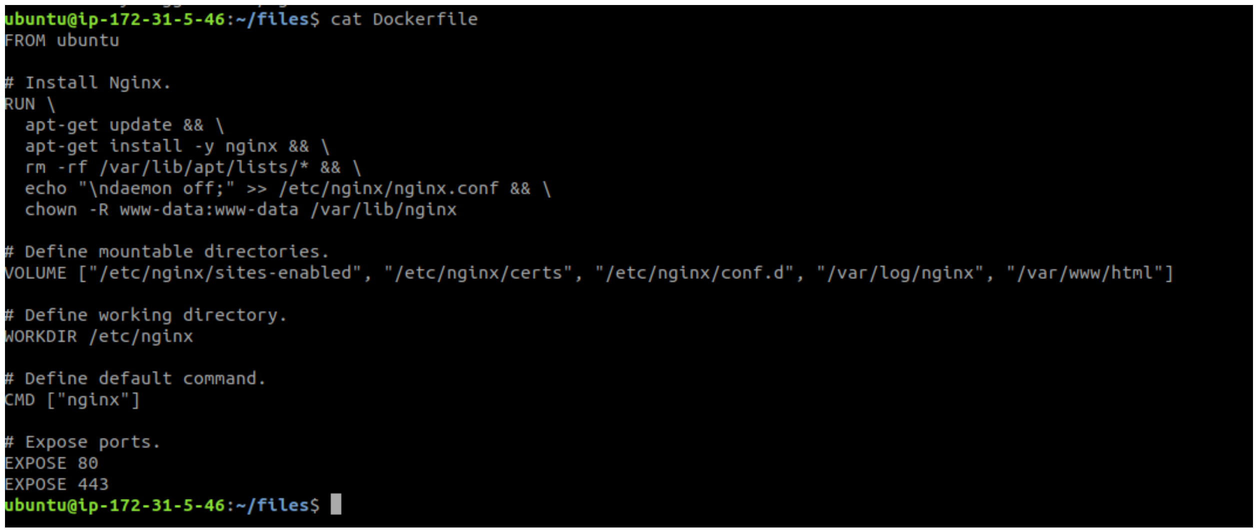
Writing a Dockerfile (A container that runs the Nginx web server)

1. The first step would be writing a Dockerfile for any application (Nginx in our demo). The base image used is the Ubuntu image.
2. Docker provides a set of standard instructions to be used in the Dockerfile, such as FROM, COPY, RUN, ENV, EXPOSE and CMD, which are a few basic ones.
3. Every Dockerfile must start with the FROM instruction. The idea behind this is that you need a starting point (mostly base OS) to build your image.
4. In this example, we will take Ubuntu as the base image. No-tag is used along with the image name, which means that we are using the latest image. You can try with the *ubuntu:18.04* image or some other versions simply by changing the tag after the semicolon.
5. The RUN command is used to run instructions against the image. In our case, we first updated our Ubuntu system and then installed the Nginx server on our Ubuntu image.
6. CMD instructions define the command that gets executed when the container starts. CMD instruction is overridden if a command is specified in the terminal as a part of the Docker run command.
7. The EXPOSE instruction does not actually publish the port. It functions as a type of documentation between the person who builds the image and the person who runs the container.
8. The VOLUME instruction creates a mount point with the specified name and marks it as holding externally mounted volumes from the native host or other containers.

Dockerfile used in aforementioned demo

```
FROM ubuntu

# Install Nginx.
```

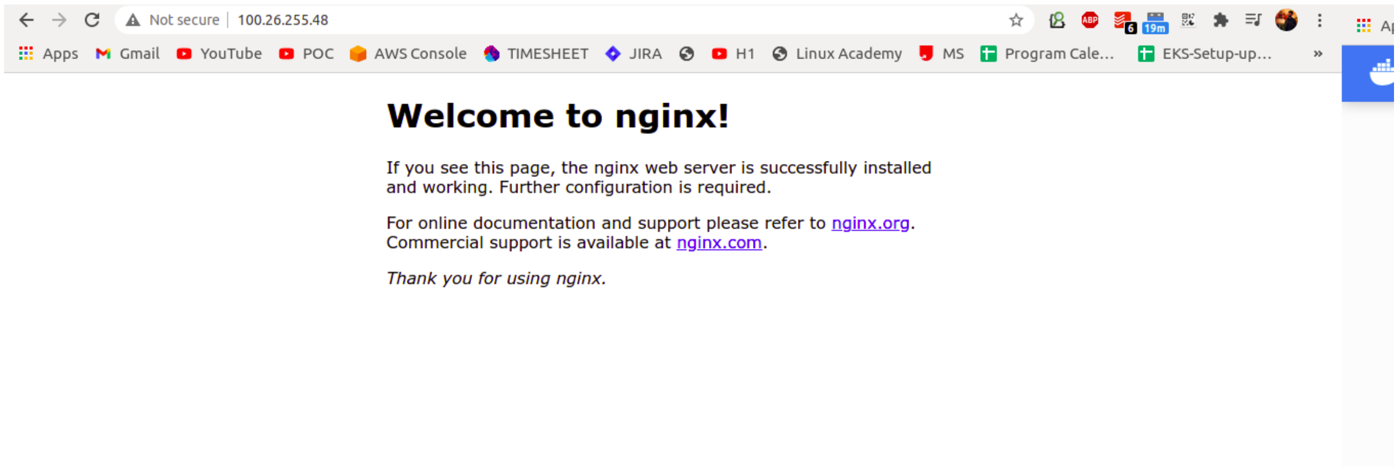
```
RUN \
  apt-get update && \
  apt-get install -y nginx && \
  rm -rf /var/lib/apt/lists/* && \
  echo "\ndaemon off;" >> /etc/nginx/nginx.conf && \
  chown -R www-data:www-data /var/lib/nginx

# Define mountable directories.
VOLUME ["/etc/nginx/sites-enabled", "/etc/nginx/certs", "/etc/nginx/conf.d",
"/var/log/nginx", "/var/www/html"]


# Define the working directory.
WORKDIR /etc/nginx



# Define default command.
CMD ["nginx"]

# Expose ports.
EXPOSE 80
EXPOSE 443
```



```
ubuntu@ip-172-31-5-46:~/files$ cat Dockerfile
FROM ubuntu

# Install Nginx.
RUN \
  apt-get update && \
  apt-get install -y nginx && \
  rm -rf /var/lib/apt/lists/* && \
  echo "\ndaemon off;" >> /etc/nginx/nginx.conf && \
  chown -R www-data:www-data /var/lib/nginx

# Define mountable directories.
VOLUME ["/etc/nginx/sites-enabled", "/etc/nginx/certs", "/etc/nginx/conf.d", "/var/log/nginx", "/var/www/html"]

# Define working directory.
WORKDIR /etc/nginx

# Define default command.
CMD ["nginx"]

# Expose ports.
EXPOSE 80
EXPOSE 443
ubuntu@ip-172-31-5-46:~/files$
```

9. Next is creating the image and running the container using the following commands:
   docker build . -t demo/nginx
   docker run -it -p 80:80 demo/nginx

10. The last step is hitting the container port in the browser to show **Nginx Welcome Page.**



Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

ADD versus COPY, CMD versus ENTRYPOINT

COPY Command vs ADD Command

Some important points to note are as follows:

1. COPY takes in a source and a destination. It lets you copy a local file or directory from your host (the machine building the Docker image) into the Docker image itself.
2. ADD allows you to do this as well, but it also supports two other sources. First, you can use a URL instead of a local file/directory. Second, you can extract a Tar file from the source directly into the destination.

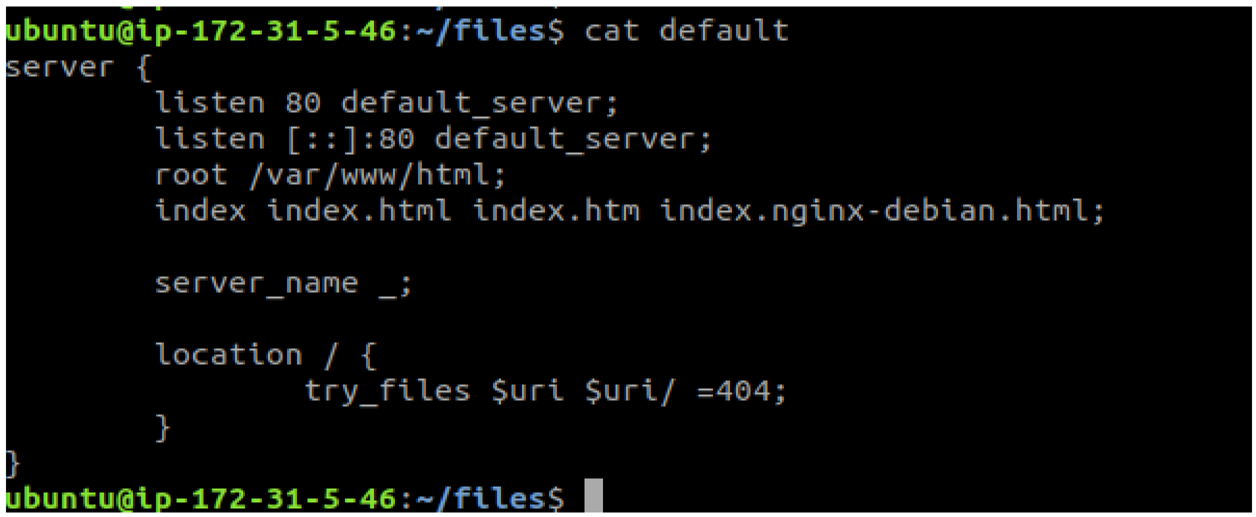
The following demonstration will help you understand the above-mentioned concepts better.

In this demonstration, we will customise the Nginx web server by replacing the default file at **/etc/nginx/sites-enabled/default** with the following content so that Nginx can host files present in the **/var/www/html** folder. We will then put a custom **index.html file** in the **/var/www/html** folder of the container.

This will be done by using the RUN and COPY commands in the Dockerfile.

default

```
server {
        listen 80 default_server;
        listen [::]:80 default_server;
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                try_files $uri $uri/ =404;
        }
}
```

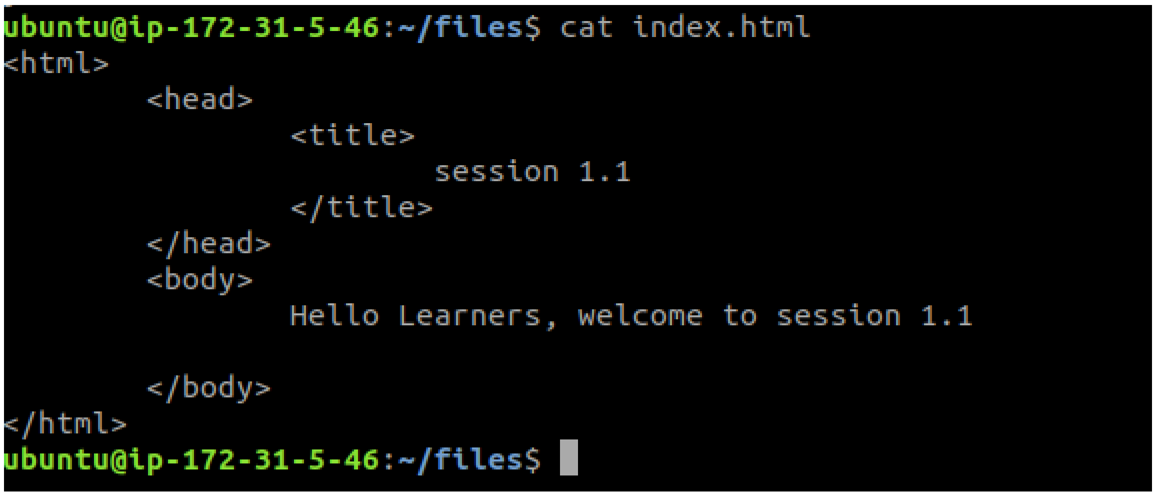


```
ubuntu@ip-172-31-5-46:~/files$ cat default
server {
        listen 80 default_server;
        listen [::]:80 default_server;
        root /var/www/html;
        index index.html index.htm index.nginx-debian.html;

        server_name _;

        location / {
                try_files $uri $uri/ =404;
        }
}
ubuntu@ip-172-31-5-46:~/files$
```

Following it is the “index.html” that will be placed in **/var/www/html** directory.

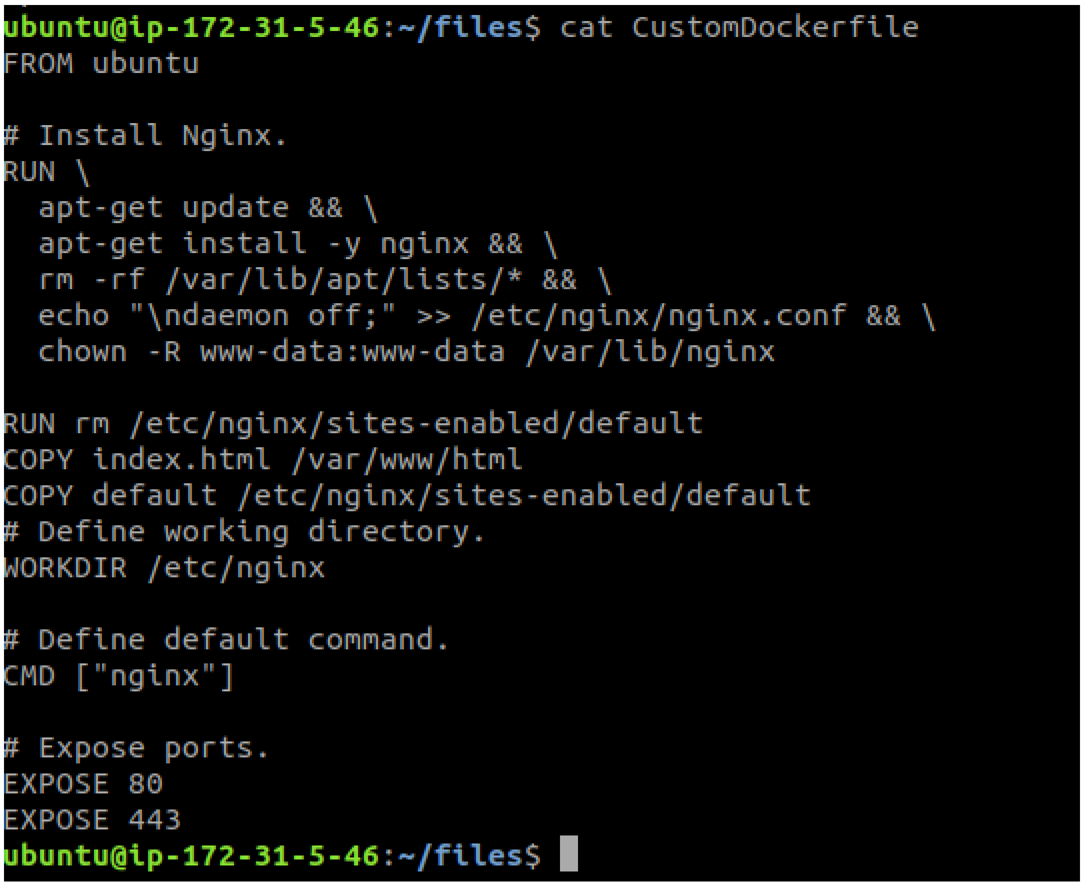
index.html

```
<html>
        <head>
                <title>
                        session 1.1
                </title>
        </head>
        <body>
                Hello Learners, welcome to session 1.1
        </body>
</html>
```



```
ubuntu@ip-172-31-5-46:~/files$ cat index.html
<html>
        <head>
                <title>
                        session 1.1
                </title>
        </head>
        <body>
                Hello Learners, welcome to session 1.1

        </body>
</html>
ubuntu@ip-172-31-5-46:~/files$
```

The final Dockerfile will look like this:



```
ubuntu@ip-172-31-5-46:~/files$ cat CustomDockerfile
FROM ubuntu

# Install Nginx.
RUN \
  apt-get update && \
  apt-get install -y nginx && \
  rm -rf /var/lib/apt/lists/* && \
  echo "\ndaemon off;" >> /etc/nginx/nginx.conf && \
  chown -R www-data:www-data /var/lib/nginx

RUN rm /etc/nginx/sites-enabled/default
COPY index.html /var/www/html
COPY default /etc/nginx/sites-enabled/default
# Define working directory.
WORKDIR /etc/nginx

# Define default command.
CMD ["nginx"]

# Expose ports.
EXPOSE 80
EXPOSE 443
ubuntu@ip-172-31-5-46:~/files$
```

CustomDockerfile

```
FROM ubuntu

# Install Nginx.
RUN \
  apt-get update && \
  apt-get install -y nginx && \
  rm -rf /var/lib/apt/lists/* && \
  echo "\ndaemon off;" >> /etc/nginx/nginx.conf && \
  chown -R www-data:www-data /var/lib/nginx

RUN rm /etc/nginx/sites-enabled/default
COPY index.html /var/www/html
COPY default /etc/nginx/sites-enabled/default
# Define working directory.
WORKDIR /etc/nginx

# Define default command.
CMD ["nginx"]

# Expose ports.
EXPOSE 80
EXPOSE 443
```

Now, build the image, run the container and hit the server.

```
docker build -f CustomDockerfile . -t demo/nginx2
```

```
docker run -it -p 80:80 demo/nginx2
```

Output: **Hello Learners welcome to session 1.1**

CMD vs ENTRYPOINT in a Dockerfile

Both CMD and ENTRYPOINT instructions are used to define the command that gets executed while starting a container. Some important points to note about their usage are as follows:

1. The docker file should have at least one command specified either in CMD or in ENTRYPOINT.
2. ENTRYPOINT should be defined while using the container as an executable.
3. CMD should be used as a way of defining default arguments for an ENTRYPOINT command or for executing an ad-hoc command in a container.
4. CMD will be overridden when running the container with alternative arguments.

The following demonstration illustrates the above-mentioned points:

1. Create a Dockerfile.

CMDDockerfile

```
FROM ubuntu
ENTRYPOINT ["echo", "Learners"]
CMD ["echo", "Hello", "World"]
```

2. Run the following command:

```
docker build -f CMDDockerfile . -t demo/cmd
```

```
docker run -it demo/cmd
```

or

```
docker run -it demo/cmd Welcome
```

3. Output: Watch the difference between CMD and ENTRYPOINT.