# **Docker – Building a Web Server Docker File**

We have already learnt how to use Docker File to build our own custom images. Now let’s see how we can build a web server image which can be used to build containers.

In our example, we are going to use the Apache Web Server on Ubuntu to build our image.

Let’s follow the steps given below, to build our web server Docker file.

**Step 1**: The first step is to build our Docker File. Let’s use **vim** and create a Docker Filewith the following information.

FROM ubuntu

RUN apt-get update

RUN apt-get install –y apache2

RUN apt-get install –y apache2-utils

RUN apt-get clean

EXPOSE 80

CMD [“apache2ctl”, “-D”, “FOREGROUND”]

The following points need to be noted about the above statements:

* We are first creating our image to be from the Ubuntu base image.
* Next, we are going to use the RUN command to update all the packages on the Ubuntu system.
* Next, we use the RUN command to install apache2 on our image.
* Next, we use the RUN command to install the necessary utility apache2 packages on our image.
* Next, we use the RUN command to clean any unnecessary files from the system.
* The EXPOSE command is used to expose port 80 of Apache in the container to the Docker host.
* Finally, the CMD command is used to run apache2 in the background.

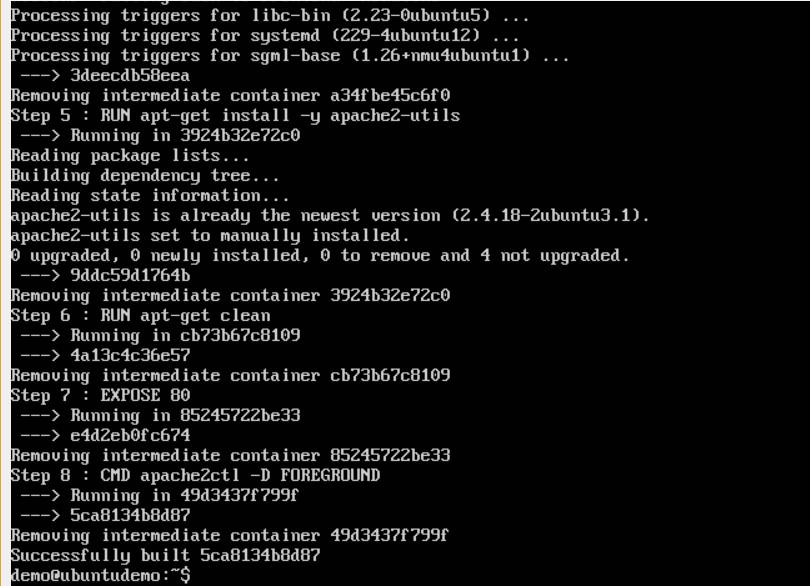


Now that the file details have been entered, just save the file.

**Step 2**: Run the Docker **build** command to build the Docker file. It can be done using thefollowing command:

sudo docker build –t=”mywebserver” .

We are tagging our image as **mywebserver**. Once the image is built, you will get a successful message that the file has been built.



**Step 3**:Now that the web server file has been built, it’s now time to create a containerfrom the image. We can do this with the Docker **run** command.

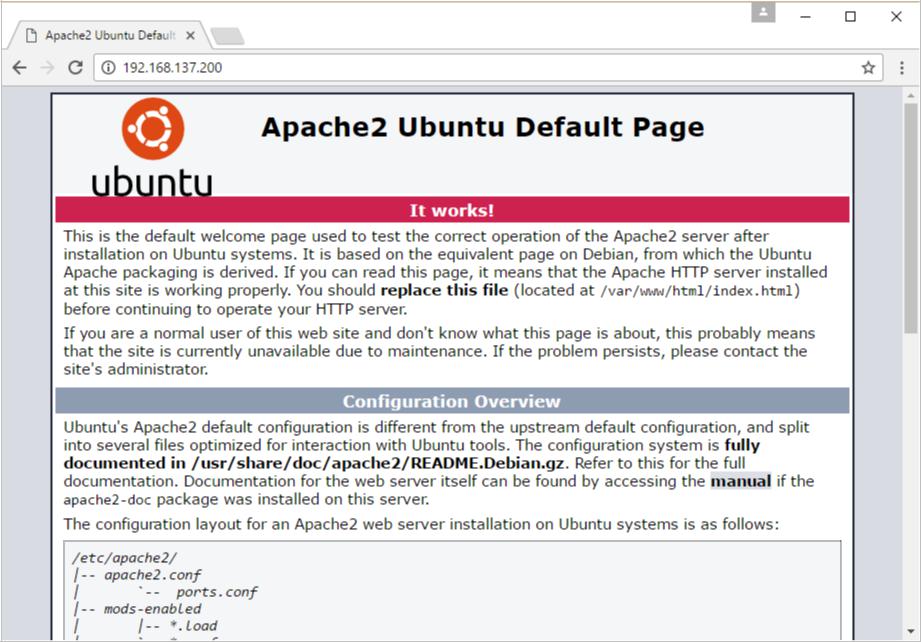
sudo docker run –d –p 80:80 mywebserver



The following points need to be noted about the above command:

* The port number exposed by the container is 80. Hence with the **–p** command, we are mapping the same port number to the 80 port number on our localhost.
* The **–d** option is used to run the container in detached mode. This is so that the container can run in the background.

If you go to port 80 of the Docker host in your web browser, you will now see that Apache is up and running.



**Docker – Instruction Commands**

Docker has a host of instruction commands. These are commands that are put in the Docker File. Let’s look at the ones which are available.

**CMD Instruction**

This command is used to execute a command at runtime when the container is executed.

**Syntax**

CMD command param1

**Options**

* **command** –This is the command to run when the container is launched.
* **param1** –This is the parameter entered to the command.

**Return Value**

The command will execute accordingly.

**Example**

In our example, we will enter a simple **Hello World** echo in our Docker File and create an image and launch a container from it.

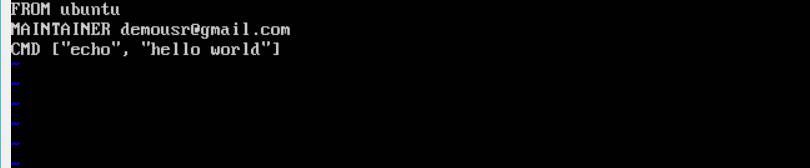
**Step 1**: Build the Docker File with the following commands:

FROM ubuntu

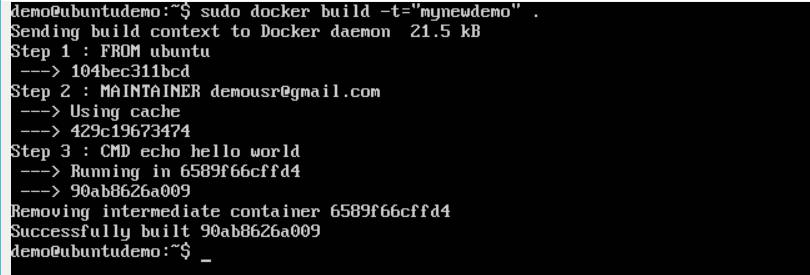
MAINTAINER [demousr@gmail.com](mailto:demousr@gmail.com)

CMD [“echo” , “hello world”]

Here, the CMD is just used to print **hello world**.



**Step 2**: Build the image using the Docker **build** command.



**Step 3**: Run a container from the image.



**ENTRYPOINT**

This command can also be used to execute commands at runtime for the container. But we can be more flexible with the ENTRYPOINT command.

**Syntax**

ENTRYPOINT command param1

**Options**

* **command** –This is the command to run when the container is launched.
* **param1** –This is the parameter entered into the command.

**Return Value**

The command will execute accordingly.

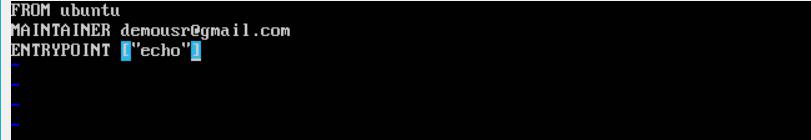
**Example**

Let’s take a look at an example to understand more about ENTRYPOINT. In our example, we will enter a simple **echo** command in our Docker File and create an image and launch a container from it.

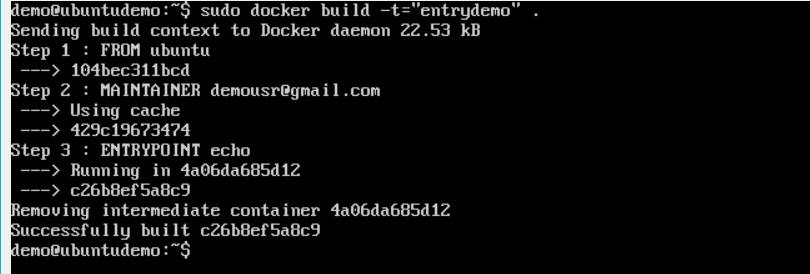
**Step 1**: Build the Docker File with the following commands:

FROM ubuntu

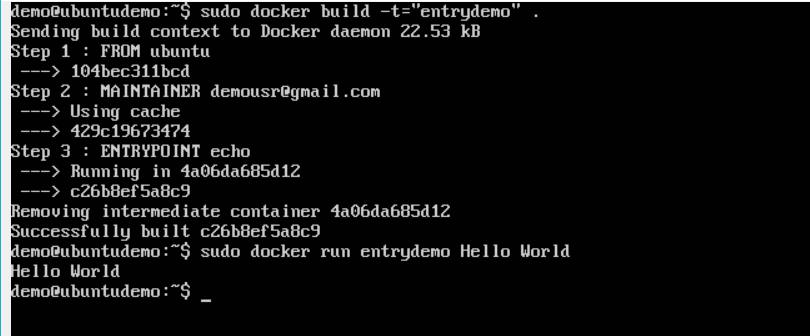
MAINTAINER [demousr@gmail.com](mailto:demousr@gmail.com)

ENTRYPOINT [“echo”]

**Step 2**: Build the image using the Docker **build** command.



**Step 3**: Run a container from the image.



**ENV**

This command is used to set environment variables in the container.

**Syntax**

ENV key value

**Options**

* **Key** –This is the key for the environment variable.
* **value** –This is the value for the environment variable.

**Return Value**

The command will execute accordingly.

**Example**

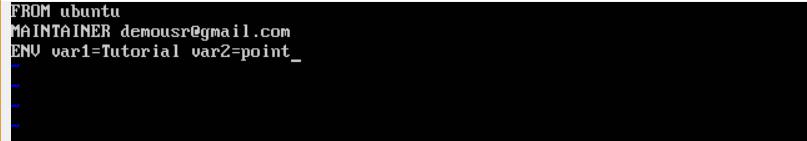
In our example, we will enter a simple **echo** command in our Docker File and create an image and launch a container from it.

**Step 1**: Build the Docker File with the following commands:

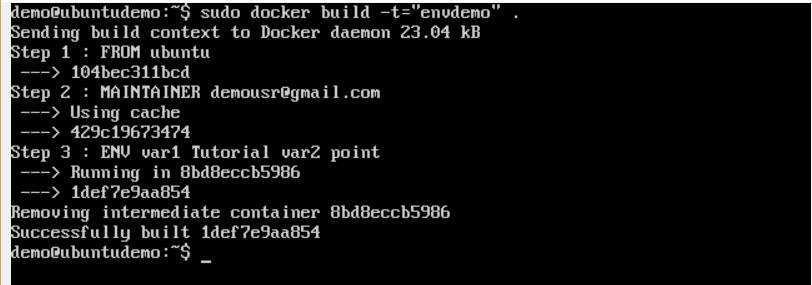
FROM ubuntu

MAINTAINER [demousr@gmail.com](mailto:demousr@gmail.com)

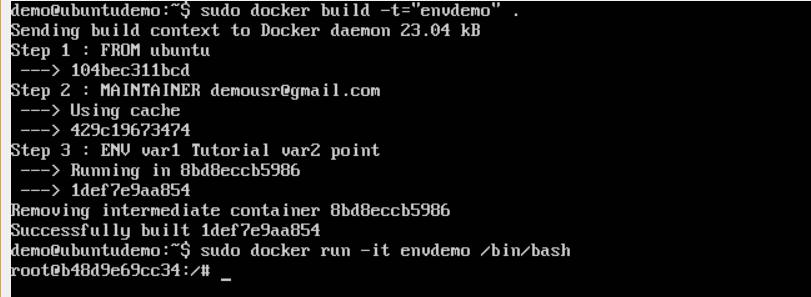
ENV var1=Tutorial var2=point



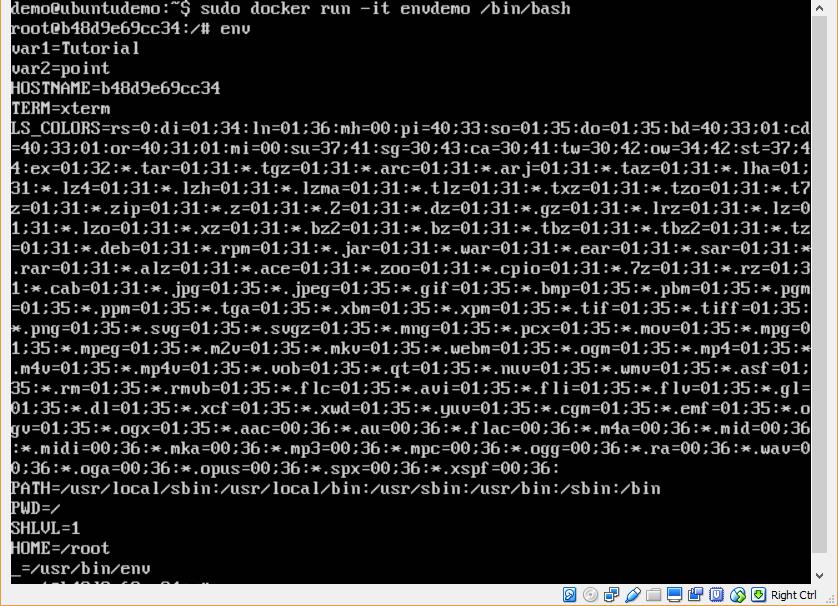
**Step 2**: Build the image using the Docker **build** command.



**Step 3**: Run a container from the image.



**Step 4**: Finally, execute the **env** command to see the environment variables.



**WORKDIR**

This command is used to set the working directory of the container.

**Syntax**

WORKDIR dirname

**Options**

* **dirname** –The new working directory. If the directory does not exist, it will beadded.

**Return Value**

The command will execute accordingly.

**Example**

In our example, we will enter a simple **echo** command in our Docker File and create an image and launch a container from it.

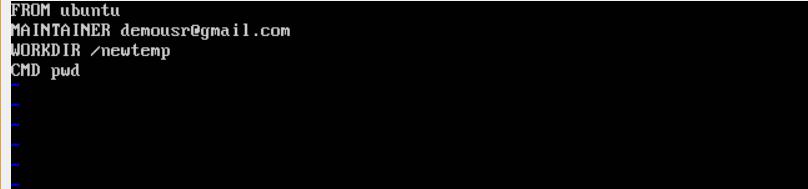
**Step 1**: Build the Docker File with the following commands:

FROM ubuntu

MAINTAINER [demousr@gmail.com](mailto:demousr@gmail.com)

WORKDIR /newtemp

CMD pwd



**Step 2**: Build the image using the Docker **build** command.



**Step 3**: Run a container from the image.

