How to Create a Docker Image

By **Swapnil Bhartiya** - January 22, 2018

In the previous article, we learned about how to get started with Docker on Linux, macOS, and Windows. In this article, we will get a basic understanding of creating Docker images. There are prebuilt images available on DockerHub that you can use for your own project, and you can publish your own image there.

We are going to use prebuilt images to get the base Linux subsystem, as it's a lot of work to build one from scratch. You can get Alpine (the official distro used by Docker Editions), Ubuntu, BusyBox, or scratch. In this example, I will use Ubuntu.

Before we start building our images, let's "containerize" them! By this I just mean creating directories for all of your Docker images so that you can maintain different projects and stages isolated from each other.

```
$ mkdir dockerprojects

cd dockerprojects
```

Now create a *Dockerfile* inside the *dockerprojects* directory using your favorite text editor; I prefer nano, which is also easy for new users.

```
$ nano Dockerfile
```

And add this line:

```
FROM Ubuntu
```

m7_f7No0pmZr2iQmEOH5_ID6MDG2oEnODpQZkUL7

Save it with Ctrl+Exit then Y.

Now create your new image and provide it with a name (run these commands within the same directory):

```
$ docker build -t dockp .
```

(Note the dot at the end of the command.) This should build successfully, so you'll see:

```
Sending build context to Docker daemon  2.048kB

Step 1/1 : FROM ubuntu

---> 2a4cca5ac898

Successfully built 2a4cca5ac898

Successfully tagged dockp:latest
```

It's time to run and test your image:

```
$ docker run -it Ubuntu
```

You should see root prompt:

```
root@c06fcd6af0e8:/#
```

This means you are literally running bare minimal Ubuntu inside Linux, Windows, or macOS. You can run all native Ubuntu commands and CLI utilities.

vpZ8ts9oq3uk--z4n6KP3DD3uD_P4EpG7fX06MC3

Let’s check all the Docker images you have in your directory:

```
$docker images
```

```
REPOSITORY          TAG                 IMAGE ID

dockp               latest              2a4cca5ac898

ubuntu              latest              2a4cca5ac898

hello-world         latest              f2a91732366c
```

You can see all three images: *dockp, Ubuntu,* and *hello-world*, which I created a few weeks ago when working on the previous articles of this series. Building a whole LAMP stack can be challenging, so we are going create a simple Apache server image with Dockerfile.

Dockerfile is basically a set of instructions to install all the needed packages, configure, and copy files. In this case, it’s Apache and Nginx.

You may also want to create an account on DockerHub and log into your account before building images, in case you are pulling something from DockerHub. To log into DockerHub from the command line, just run:

```
$ docker login
```

Enter your username and password and you are logged in.

Next, create a directory for Apache inside the dockerproject:

```
$ mkdir apache
```

Create a Dockerfile inside Apache folder:

```
$ nano Dockerfile
```

And paste these lines:

```
FROM ubuntu

MAINTAINER Kimbro Staken version: 0.1

RUN apt-get update && apt-get install -y apache2 && apt-ge

ENV APACHE_RUN_USER www-data

ENV APACHE_RUN_GROUP www-data

ENV APACHE_LOG_DIR /var/log/apache2

EXPOSE 80

CMD ["/usr/sbin/apache2", "-D", "FOREGROUND"]
```

Then, build the image:

```
docker build -t apache .
```

(Note the dot after a space at the end.)

It will take some time, then you should see successful build like this:

```
Successfully built e7083fd898c7

Successfully tagged ng:latest

Swapnil:apache swapnil$
```

Now let's run the server:

```
$ docker run –d apache

a189a4db0f7c245dd6c934ef7164f3ddde09e1f3018b5b90350df8be85
```

Eureka. Your container image is running. Check all the running containers:

```
$ docker ps

CONTAINER ID  IMAGE        COMMAND                  CREATED

a189a4db0f7 apache "/usr/sbin/apache2ctl"  10 seconds ago
```

You can kill the container with the *docker kill* command:

```
$docker kill a189a4db0f7
```

So, you see the "image" itself is persistent that stays in your directory, but the container runs and goes away. Now you can create as many images as you want and spin and nuke as many containers as you need from those images.

That's how to create an image and run containers.

To learn more, you can open your web browser and check out the documentation about how to build more complicated Docker images like the whole LAMP stack. Here is a Dockerfile file for you to play with.

In the next article, I'll show how to push images to DockerHub.

Learn more about Linux through the free "Introduction to Linux" course from The Linux Foundation and edX.

Swapnil Bhartiya