Docker is probably one of the easiest environments to create a virtualised instance based on a number of flavours of operating systems. Rather that having to install an operating system yourself, you can download one of the many guests templates or ‘images’ available directly from the Docker community.
See my blog post on installing Docker on Ubuntu 14.04 if you don’t currently have Docker installed.
There are a number of commands which are required to manage Docker containers and images. First off, let’s see if we have any images in our local Docker library.

docker images

The docker images command lists the available local images which you can use to create a Docker container. The above output does not show any local images so lets download one from the central Docker repository.
We must choose which image to download and use to create our first Docker container. There are literally thousands of images available on the central repository and all can be downloaded through the docker command. Let’s use the search command to find an image to download

docker search ubuntu

This will display a huge list of all the images available containing the word ubuntu. As you can imagine, there will be hundreds because not only are base OS images available, but customised images containing specific applications or set ups.

docker pull ubuntu:14.04

You can check this has downloaded the image to your local store with the above docker images command. We will also need to make a note of the image ID so that we can use it to create a container from it.

docker images

The next step is to create a container and make the required changes. Creating a container is Docker is done with the run command followed by, amongst other things, a command to run within the container. We are going to create a new container and use a bash session to customise the container before saving it as a new image for use in the future

Create the Docker container with the **run** command and specify the **bash** shell to be executed on completion. This will leave us with a bash session which we can use the customise the image. Replace the **ad892dd21d60** ID with the ID of the image we downloaded in the previous step.

```
docker run -i -t ad892dd21d60  /bin/bash
```

You now have an active shell on the container which has been created with the id 3a09b2588478. Type **exit** to end the session in your guest container and the container will be stopped and kept available on your Docker system.

Run the ps Docker command to see what containers are known to your Docker system.

```
docker ps -a
```

The above output shows 3 containers which are available in my Docker system with the container ID on the left. We can re-enter one of these containers to make our changes, but first we need to start it. I'm going to use container ID 3a09b2588478 for the rest of this example but yours will be a different ID.

```
docker start 3a09b2588478
```

We can now attach to the container to create a shell where we can make our modifications**.**

```
docker attach 3a09b2588478
```

You now have a shell running on the container which you can use to make your changes to the container. Let's keep it simple and just run an upgrade with apt-get and then exit. In the real world, you might install an application, or define your configuration such as LDAP SSH login.

```
$ apt-get update
$ apt-get upgrade
$ exit
```

The last step in our example is to save the container as a new image which can be used to create future Docker containers. You'll need to specify the container ID as well as the name of the image to use. You can specify a new image name or overwrite the existing image name.

```
$ docker commit 3a09b2588478 ubuntu:14.04
```

And that's all there is to it! You have created a new Docker container, from one of the images available from Docker, made some changes and saved it locally for future use. Of cause, there are plenty more ways to use Docker, but I hope this has been useful for getting a basic understanding of how Docker works.

Using Dockerfiles to build new Docker images

Docker has a scripting language which can be used to create a new instance with a predefined list of commands and properties which will be used to create your new Docker instance.You could, for example, use a docker file to install Apache, configure the firewall and any further configurations we may need to make.
The benefits to using a Dockerfile, rather than making all the changes directly and saving the image are that the underlying OS and the additions that you wish to make are completely independent. You can, for example, run a Dockerfile on any OS image. Using the example that follows, you could run the Dockerfile on either a Debian or Ubuntu OS without changing a thing.
Create a directory to hold your DockerFile project, which we'll call apache2 for this example. I'll be placing all my DockerFiles in their own project directory under dockerfiles in my home directory.

mkdir -p /home/james/dockerfiles/apache2

Open a text file named Dockerfile in your favourite text editor in the project folder we just created. This is the standard file structure that Docker expects when creating DockerFiles.

vi /home/james/dockerfiles/apache2/Dockerfile

There are various commands we can use within a Dockerfile. The first command is the FROM statement which indicates which image should be used when creating your instance. I'm going to use the ubuntu image which I have previously downloaded to my local Docker server.

FROM ubuntu:14.04

Add MAINTAINER or author for the template. This is your name, username or whatever handle you'd like to be known as.

MAINTAINER James Coyle <james.coyle@jamescoyle.net>

We are now going to use the RUN command to specify the commands that should be executed on the instance during creation. The commands will be executed in the order they appear in the Dockerfile. We will be installing Apache2 so we'll be using the apt-get command to install.

```
RUN apt-get update
RUN apt-get upgrade -y
RUN apt-get install -y apache2
```

Next we'll make a data directory on the host where we will keep our web files that are to be served by Apache.

```
RUN mkdir -p /data/apache/www
RUN chown -R root:www-data /data/apache/www
```

It's a good idea to separate the Docker container from any user data so that a the container can be used for different purposes. What this mean in our example is that we will keep all the website data (HTML files, etc.) out of our container, leaving only the Apache software and general configuration within the container. This means that we can reuse our Docker image to create containers for other websites.
Using the Docker VOLUME command we can map a directory on the Docker host to a folder inside the container which will be configured once your container is created. The below example makes the directory /data/apache/www available for mapping later.
Add the VOLUME reference to your Dockerfile.

```
VOLUME /data/apache/www
```

We will need to be able to reach our container on port 80 so that we can use the Apache service over the network. Docker uses the EXPOSE command followed by a port number to allow incoming traffic to the container. Add the below entry to allow port 80.

```
EXPOSE 80
```

We now need to do some find and replace magic to change the Apache default site configuration to point to our new location, rather than the Apache default. This isn't a Docker specific command, but is required for this example.

```
RUN sed -i 's#DocumentRoot /var/www#DocumentRoot /data/apache/www#'
/etc/apache2/sites-available/000-default.conf
```

Finally we'll need to tell Docker what should be executed in order to 'run' this container. For this example, we use the apache2ctl command with the FOREGROUND switch.

```
ENTRYPOINT /usr/sbin/apache2ctl -D FOREGROUND
```

And that’s it, your first DockerFile. Run your newly created DockerFile to build the image by changing to the project directory and using the docker build command to create it. Use the -t switch to specify a tag for the image.

```
cd /home/james/dockerfiles/apache2
docker build -t apache2:test .
```

It will take a few minutes for the image to build. Once complete, you’ll be able to see it in the Docker image list by using the command docker image.

```
root@docker:~/apache2# docker images
```

The whole DockerFile:

```
FROM ubuntu:14.04

MAINTAINER James Coyle <james.coyle@jamescoyle.net>

RUN apt-get update
RUN apt-get upgrade -y
RUN apt-get install -y apache2

RUN mkdir -p /data/apache/www
RUN chown -R root:www-data /data/apache/www

VOLUME /data/apache/www

EXPOSE 80

RUN sed -i 's#DocumentRoot /var/www#DocumentRoot /data/apache/www#'
/etc/apache2/sites-available/000-default.conf

ENTRYPOINT /usr/sbin/apache2ctl -D FOREGROUND
```