Lab 1-3 - Making our own image file

In this lab we will build on what we've learned so far about managing images and containers, running images and interacting with containers by creating our own image, which has a 'layer' on top of an already existing image.

All images are built in this layered way, just as nginx is installed as a layer on top of a minimal debian image, we will create an image which is a layer on top of nginx

Building our own image

1. get back to our home directory, if we've left it, then lets create a directory for our image files

   ```
   $ cd ~
   $ mkdir hello-nginx
   $ cd hello-nginx
   ```

2. Lets create a html file to have served up to us. feel free to copy the following, or get creative

   ```
   $ vim hello.html
   ```

   (if you are unsure how to use the vim editor, let the lab instructor know)

   ```
   <html>
   <head><title>Hello Tenable security experts</title></head>
   <body>
   <h1>Hello world, this is {use a distinctive name here, so you know it's your file} !!!</h1>
   <p>I hope you are having a great day</p>
   </body>
   ```

   You can be a little creative here, there will be a chance for your colleagues to try this image later on.

3. Now we need to create our dockerfile to let docker know how to build our image (the capital D is important)

   ```
   $ vim Dockerfile
   ```

   ```
   FROM nginx:latest

   COPY hello.html /usr/share/nginx/html
   ```

4. We've got the two files, hello.html and Dockerfile, we can build our image (don't forget the dot at the end, it tells docker to build an image from the local directory).

   ```
   $ docker build --tag hello-nginx:1.0 .

   Sending build context to Docker daemon  3.072kB
   Step 1/2 : FROM nginx
   ---> 9beeba249f3e
   Step 2/2 : COPY hello.html /usr/share/nginx/html
   ---> 52b1352d2dca
   Successfully built 52b1352d2dca
   Successfully tagged hello-nginx:1.0
   ```

5. We've built an image, check the local images

   ```
   $ docker image ls

   REPOSITORY          TAG                 IMAGE ID            CREATED
   SIZE
   hello-nginx         1.0                 52b1352d2dca        6 seconds ago
   127MB
   nginx               latest              9beeba249f3e        3 days ago
   127MB
   ```

   There is it, pretty much the same size as the image it's layered on top of. Exciting!!!

6. Run it just like we ran nginx

   ```
   $ docker run -p 80:80 --rm -d --name hello-nginx hello-nginx:1.0
   ```

   Now navigate to the page from the previous lab

   ```
   http://{ip address of your lab vm server}/hello.html
   ```

   You should see our new file. That html is now part of the new hello-nginx image

7. As an extension see if you can connect to the container, navigate to the directory, and see the file we added to the image (by modifying instructions from the previous lab)

Push our image to a registry

There are many different container registries, both hosted and self-hosted, dockerhub is one of the popular ones, and where public docker repositories are stored. In AWS, when your organisation has IAM configured, AWS Elastic Container Registry (ECR) is very popular. We will use Dockerhub to make this lab simple.

1. Go to https://hub.docker.com/ and create a dockerhub account (if you don't already have one). The free account will allow you to have one private repository and many public repositories. (make sure to generate a new password, without further configuration the lab will store it in plain text)

2. Once your account is created, go back to your VM and log in to dockerhub using the command

   ```
   $ docker login --username={your username}
   Password:

   WARNING! Your password will be stored unencrypted in
   /home/tscott/.docker/config.json.
   Configure a credential helper to remove this warning. See
   https://docs.docker.com/engine/reference/commandline/login/#credentials-
   store
   ```

   It will ask for your password and store that locally.

3. Create a tag for our new image. My dockerhub username is tjscott, use your dockerhub username in place of that here.

   ```
   $ docker tag hello-nginx:1.0 tjscott/hello-nginx:latest
   $ docker image ls

   REPOSITORY            TAG                 IMAGE ID            CREATED
   SIZE
   hello-nginx           1.0                 52b1352d2dca        35 minutes ago
   127MB
   tjscott/hello-nginx   latest              52b1352d2dca        35 minutes ago
   127MB
   nginx                 latest              9beeba249f3e        3 days ago
   127MB
   ```

   We can see the nginx image, our local hello-nginx image and a tag of that image ready to be pushed to our dockerhub account.

4. Push our image to dockerhub (again, my username is tjscott use your dockerhub id here instead)

   ```
   $ docker push tjscott/hello-nginx

   The push refers to repository [docker.io/tjscott/hello-nginx]
   796032674611: Pushed
   6c7de695ede3: Mounted from library/nginx
   2f4accd375d9: Mounted from library/nginx
   ```

```
ffc9b21953f4: Mounted from library/nginx
latest: digest:
sha256:8ba69c30c8796e5b1e26d158c963c220170204ddeddaff6ca7dbe5cb7c58f85c
size: 1155
```

5. It will take a moment to process, but you can visit

   ```
   https://hub.docker.com/repository/docker/{your dockerhub username}/hello-
   nginx/general
   ```

   to visit your new repository

6. You can download one of your colleagues images if they provide their dockerhub username, otherwise you can get my image using

   ```
   $ docker pull tjscott/hello-nginx
   ```

7. make sure your hello-nginx container isn't running and run this image instead, you will see whatever message they put in their hello.html page

   ```
   $ docker kill hello-nginx
   $ docker run -p 80:80 --rm -d --name hello-nginx tjscott/nginx:latest
   ```

Conclusion

In this lab you learned how to

- make your own docker image
- tag that image
- push it to a container registry
- download someone elses new image
- run someone elses image.

In later labs we will build upon these skills by automating the process.