Deploying a Private Docker Registry - Exercise

Goal:

The goal of this exercise is to deploy a private Docker registry. Additionally, you will learn how to pull and push images to the registry you create.

Instructions:

Create and Start a Registry Container

First, create a volume that will be used to store the registry data. Name the volume "registry".

```
docker volume create registry
```

Next, start a container named "registry" based on the "registry" image with a tag of "2". Mount the "registry" volume to "/var/lib/registry" inside the container. Finally, publish port 5000 on the host and map it to port 5000 in the container.

Here is the command: (NOTE: Type this all on one line as this is a single command.)

```
docker run -d --name registry --mount src=registry,dst=/var/lib/registry
-p 5000:5000 registry:2
```

(NOTE: For more information about the official Docker registry image, visit https://hub.docker.com/_/registry.)

Copy an Image from Docker Hub to Your Registry

Pull down the "nginx:latest" image to your Docker host system.

```
docker pull nginx:latest
```

(NOTE: You could also use "`docker pull nginx`" as the "`latest`" tag is assumed if no tag is specified.)

Tag the image as "localhost:5000/nginx:latest". Note that the registry name is "localhost:5000", the repository is "nginx" and the tag is "latest".

```
docker tag nginx:latest localhost:5000/nginx:latest
```

Check that the image has been tagged.

```
docker images
```

You'll notice that the "nginx:latest" image and the "localhost:5000/nginx:latest" images have the same Image ID.

Push the retagged nginx image into the private registry.

```
docker push localhost:5000/nginx:latest
```

You can check that the image has been pushed to the private registry using the registry's HTTP API. (For details on the API, see https://docs.docker.com/registry/spec/api/.)

```
curl http://localhost:5000/v2/_catalog
```

You should see "nginx" listed as a repository.

```
{"repositories":["nginx"]}
```

Remove the Image from the Docker Host

Remove the "nginx:latest" image you downloaded from Docker Hub:

```
docker rmi nginx:latest
```

Confirm that the image has been removed with the following command:

```
docker images
```

Note that deleting the "nginx:latest" image does not remove the other image with the same Image ID, which is localhost:5000/nginx:latest. Of course, it also does not remove it from the private registry.

Next, let's remove the localhost:5000/nginx:latest image.

```
docker rmi localhost:5000/nginx:latest
```

You'll notice that instead of only untagging the image, it also actively deletes the layers that make up the image.

Confirm that the image has been removed with the following command:

```
docker images
```

Even though the image does not exist on our local Docker host machine, it still exists in the registry.

Pull the Image from the Private Docker Registry

To prove that the image is stored in the private registry, pull it from the private registry to your Docker host system.

```
docker pull localhost:5000/nginx
```

You will now see the image listed:

```
docker images
```

Start a Container Using the Image

Next, start a container using the image.

```
docker run -d localhost:5000/nginx
docker ps
```

OPTIONAL: Access the Private Docker Registry from a Remote Docker Host

First, determine the IP address of your Docker host machine. One way to get the IP address of your Docker host system is to use the "ip a" command:

```
ip a
```

(NOTE: If "`ip a`" doesn't work, try using the "`ifconfig`" command.)

Next, connect to another Docker host system. We'll call this one "Docker Host 2" (If you need to install Docker, see the exercises on installing Docker.)

Because the private registry we configured is not using a TLS certificate, we'll need to tell our Docker Host 2 system not to attempt to use HTTPS and instead use HTTP to communicate with the registry. To do that, we'll need to add the following to the /etc/docker/daemon.json file:

```
echo '{ "insecure-registries" : ["10.4.8.15:5000"] }' >> /etc/docker/daemon.json
```

(NOTE: If your Docker host is a Windows Server, the location of the daemon.json file is: `C:\ProgramData\docker\config\daemon.json`.)

Next, restart the Docker engine so that it picks up the new configuration:

```
systemctl restart docker
```

Now you are ready to pull the "nginx:latest" image from the original Docker Host system.

```
docker pull IP_ADDRESS_OF_YOUR_DOCKER_HOST:5000/nginx
```

In my case, the IP address of my original Docker host is 10.4.8.15, so I would use this command:

```
docker pull 10.4.8.15:5000/nginx
```

(NOTE: If you get an error such as "http: server gave HTTP response to HTTPS client", check the previous steps and ensure the contents of the /etc/docker/daemon.json file are correct and that the Docker engine has been restarted.)

Verify the image has been downloaded.

```
docker images
```

Next, start a container using this image.

```
docker run -d IP_ADDRESS_OF_YOUR_DOCKER_HOST:5000/nginx
```

Again, my IP is 10.4.8.15, so I would use this command:

```
docker run -d 10.4.8.15:5000/nginx
```

Check that the container is running.

```
docker ps
```

(NOTE: Because there is no authentication required to access the registry, be sure to only deploy it on a secure local network. Alternatively, you can add basic authentication as described in the Docker Registry documentation: https://docs.docker.com/registry/ The reason authentication was not covered in this lesson is that it requires DNS configuration which is beyond the scope of this course.)