Exercise Handbook

Session 1 Exercise 1: Finding Images on DockerHub

1) **Create a Docker Hub Account**

If you have an existing Docker Hub account, please feel free to use it for this exercise and skip this step.

You can sign up for an account at https://hub.docker.com/account/signup/. You will need a valid email address.

2) Pull an image

First, we can search for the image we want to pull. You can also search via https://registry.hub.docker.com/

```
docker search ubuntu
```

Now that we have verified the name, pull the image:

```
docker pull ubuntu
```

Session 1 Exercise 2: Running a Container

1) Run the Ubuntu image as a container

Start by looking at the list of images available locally:

```
$ docker images
REPOSITORY       TAG      IMAGE ID      CREATED      VIRTUAL SIZE
ubuntu           latest   6d4946999d4f  4 days ago   188.3 MB
```

Now, run the container using the following command:

```
$ sudo docker run -t -i ubuntu
```

Lets break that command down:

Component	Function
`docker run <imagename>`	The bare minimum required to run an image.
`-t`	Allocates a new psuedo-tty to display output. Used if you intend to use the terminal.

`-i`	Keeps stdin open even if not attached
`docker run <imagename> -it`	Starts a new container, allocates a tty and connects stdin. The combination of i and t is needed if you intend to use the container in interactive mode.

After running the command, you should see a shell for the Ubuntu instance. Feel free to try some standard commands to inspect the state of the machine.

2) Install curl

You may have noticed that some common commands are not present, as the Ubuntu image is kept as small as possible by default.

Install curl. You will need to respond to some prompts during the install process.

```
# apt-get update
# apt-get install curl
```

3) Stop the container

Use 'exit' to exit the bash prompt for your container and return to your host OS.

Note that once the bash prompt is closed, the container will stop, as it's main process (PID 1) has terminated, We will be exploring the container lifecycle in further detail in the Containers session.

4) Create a new image

Now, we can turn our modified Ubuntu with curl into a new image. Recall that the installation of curl on our container will not record any changes to our local Ubuntu image.

Identify the container that we made the change to using the following command. This will show all containers, running or stopped.

```
$ docker ps -a

CONTAINER ID IMAGE   COMMAND  CREATED   STATUS      PORTS  NAMES
942e06e20ab1 ubuntu:latest "/bin/bash" 5 minutes ago Exited (0) 2 minutes ago
condescending_wozniak
```

Based on this output, we will want to work with the container with id 942e06e20ab1.

We can create an image from this stopped container using the following command:

```
$ docker commit 942e06e20ab1 YourDockerHubUser/ubuntucurl:1.0
```

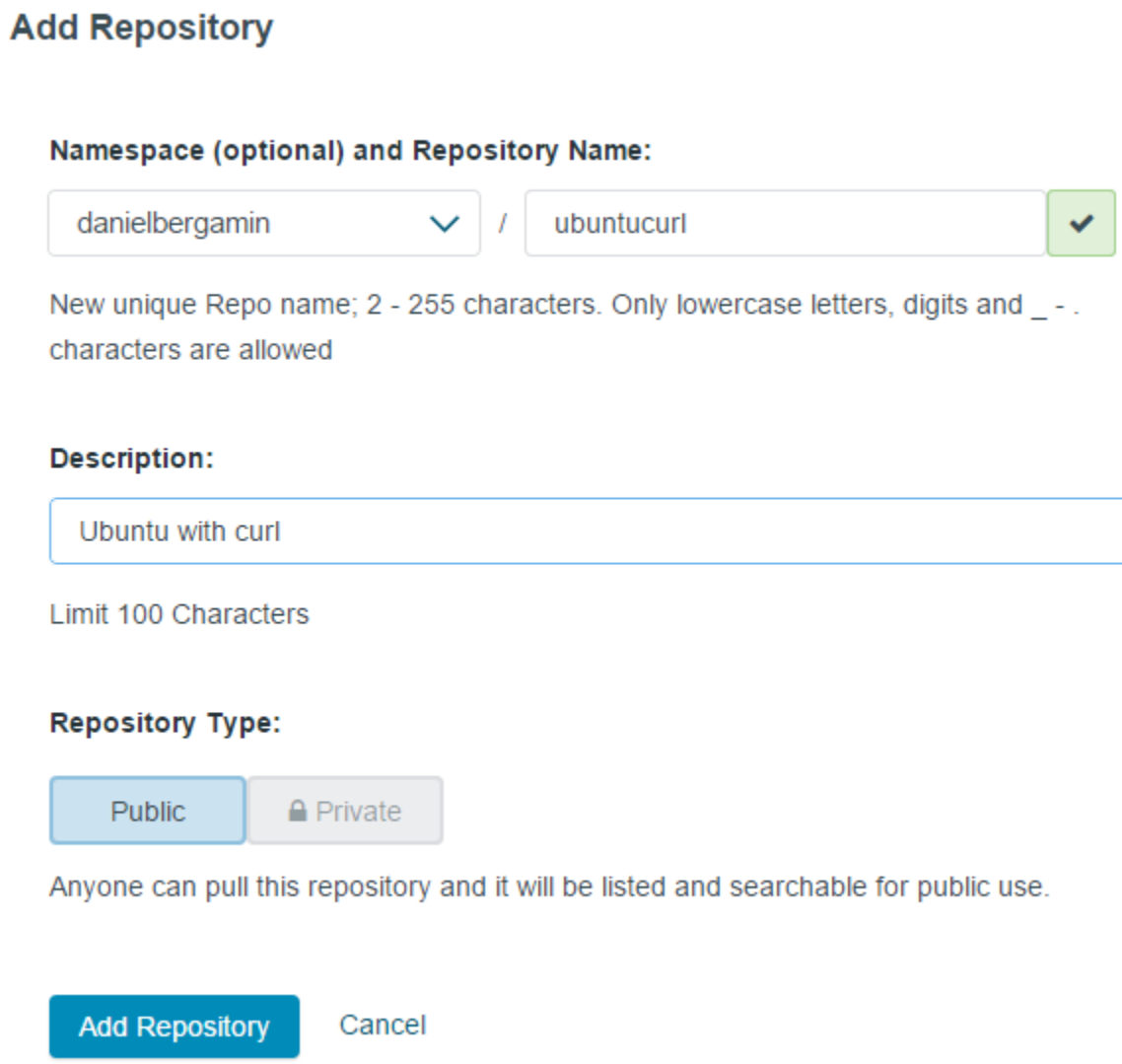
The DockerHubUser/ubuntucurl:1.0 parameters 'tags' the image with relevant details and prepares for upload to Docker Hub.

Verify that the image was tagged correctly using docker images:

```
$ docker images
REPOSITORY           TAG    IMAGE ID       CREATED      VIRTUAL SIZE
YourDockerHubUser/ubuntucurl 1.0  3e8b4d4eb571    2 minutes ago 220.8 MB
ubuntu               latest  6d4946999d4f 4 days ago   188.3 MB
```

5) Push the image to a public repository on Docker Hub

First, you will need to set up a public repository. Login to Docker Hub and click 'Add Repository'.



Now, use the following command to push to your repository:

```
docker push YourDockerHubUser/ubuntucurl
```

Session 1 Exercise 3: Tag and Push to DockerHub

Session 2 Exercise 1: Creating Dockerfiles

Session 3 Exercise 1: Container Lifecycles

Session 3 Exercise 2: Detached Mode

Session 3 Exercise 3: Managing Containers

Session 4 Exercise 1: Multi-container applications with Compose