Lab Exercise: Working With Container Utilities

Preamble

This lab exercise uses some of the commands you have learnt about in this unit, to help us retrieve information about some of the containers we ran in the exercise you conducted in Unit 4.

Step 1 – docker events

Use the `docker ps` command to retrieve the ID of the container that was created to echo 'hello from my container', by filtering those containers that have an exited status, and formatting the output to show the container ID and command only. Using the ID as an argument, list the events that Docker has retained concerning the container.

You'll remember that we paused one of our containers briefly during the exercise in Unit 4, what time did this take place?

Step 2 – docker inspect

Which command line option do you need to provide to the `docker ps` command in order to list the last created container (which has not been destroyed)?

Using the ID as an argument to `docker inspect`, and using the `Config` fieldname and `Hostname` sub-fieldname, construct a a Go text/template for the `--format` command line option, in order to retrieve the container's hostname.

How would you use `docker inspect` to retrieve the PID of the container's process from the container's configuration? Test out your hypothesis. Why is the PID `0`?

Step 3 – docker logs

Using `docker logs` command line option, retrieve the messages that were sent to `stdout` and `stderr` for the container referenced in step 2 above.

The container is in a stopped state; in another `bash` command shell, restart it using the `docker start` command, remembering to attach to `stdin`, `stdout` and `stderr`. In the other terminal, run the `docker logs` command again, this time using the `--follow` command line option. Execute some commands (`ls`, `ps`, `top`) to verify that the container's streams are being logged in real-time by the `docker logs` command you've just started.

Step 4 – docker cp

Start a new container based on the `ubuntu:latest` image with the `/bin/bash` command (don't specify a read-only filesystem) this time).

In a temporary directory on your local host, create a simple directory tree (perhaps by copying a directory from elsewhere on your system). Use the `tar` command to create an archive of the directory tree:

```
$ tar -cvf tree.tar .
```

Now use the `docker cp` command to copy the files to a location on the new container's filesystem (e.g. `/home`), remembering to use the — symbol as the `hostpath` (substitute `<container>` with the ID or name of the new container) :

```
$ cat tree.tar | docker cp - <container>:/home
```

Return to the terminal attached to the container, and verify that the directory tree has been reproduced at the location specified in the container's filesystem.