Lab Exercise: Configuring the Docker Daemon

Preamble

This lab exercise demonstrates the steps involved in making configuration changes to the Docker daemon.

Step 1 – set the daemon logging level to debug and restart the Docker daemon

Use the method appropriate for your Docker host to check the configuration files, and change the level of logging if necessary.

Use the method appropriate for your Docker host to restart the Docker daemon in order for the configuration change to be established.

Check that your change has taken effect by showing a process listing, filtered for the Docker daemon. Depending on whether you used the `-D` or the `--log-level` configuration option, you should see this reflected in the process listing.

Step 2 – follow the daemon log

Use a method appropriate for your Docker host to follow the log associated with the Docker daemon.

Open another terminal, and issue the `docker info` and `docker version` commands. You should see multiple log messages for each of the related commands that correspond to different logging levels.

For good measure, we'll query the Docker host again, but this time interact with the Remote API directly, rather than using the Docker CLI commands. We will use three of the Remote API endpoints, in order to; ping the daemon, display system-wide information (equivalent to `docker info`), and to show the Docker version information (equivalent to `docker version`). The REST endpoints to use are:

Ping:	`GET /_ping`
Info:	`GET /info`
Version:	`GET /version`

How you do this, depends on whether your Docker daemon is listening on a local UNIX domain socket (`/var/run/docker.sock`), or a remote TCP socket.

UNIX Domain Socket

For the UNIX domain socket, we need to `echo` the API call and pipe it to the `netcat` utility, which can be directed to communicate with UNIX domain sockets; watch for the daemon receiving the API call in the logs, as before:

```
$ echo -e "GET /_ping HTTP/1.1\r\n" | nc -U /var/run/docker.sock
```

We should get a 200 OK response from the daemon, and a log message in the followed log.

Do the same for the other two endpoints, and observe the output (which should be identical to that retrieved using the Docker CLI commands), and the log messages.

Remote TCP Socket

For the remote TCP socket, we can achieve the same thing with the use of the `curl` utility, but in order for it to perform correctly, we need to take account of the TLS requirement. To address the `GET /_ping` endpoint, for example, use the following (substituting the correct IP address for the Docker daemon, which can be found with `docker-machine ip` <name>):

```
$ curl --insecure --cert $DOCKER_CERT_PATH/cert.pem \
--key $DOCKER_CERT_PATH/key.pem https://192.168.99.100:2376/_ping
```

Step 3 – restore the logging level back to info

Finally, restore the daemon's logging level back to `info`. Restart the Docker daemon.