Accessing Application Logs in Docker

Application logs from container

We'll cover the following

- Container logs
- Real-time logs

Here, we will see how to access application logs from the container.

Container logs

When we run our app in Docker, we might need to access logs of the server. If your container is running in daemon mode or in the background, type `docker ps` to list all running containers.

```
CONTAINER ID        IMAGE               COMMAND             CREATED
             STATUS              PORTS               NAMES
64dcb356b5e4        flask_app:1.0       "flask run"         12 hours ag
o        Up 12 hours                             charming_solomon
```

Now, on the command line, type

`docker logs <CONTAINER ID>`

This should show all the logs our application server has created so far. Below is a snapshot of my logs:

```
 * Serving Flask app "app.py" (lazy loading)
 * Environment: development
 * Debug mode: on
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 310-087-588
 * Detected change in '/code/app.py', reloading
 * Restarting with stat
 * Debugger is active!
```

```
 * Debugger PIN: 310-087-588
 * Detected change in '/code/app.py', reloading
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 310-087-588
```

You can further save these logs in any file by redirecting stdout to a file.

`docker logs <CONTAINER ID> >output.log`

Real-time logs

Although in production, the log management strategy will be different, for startups or a small case project, there might be a situation where you have to watch real-time logs rather than access old logs.

There are two ways to do that:

1. This is a little bit critical. You can attach a daemon container directly to a command prompt. In short, you will bring the container foreground from a daemon mode.

Open a new terminal window like so:

- `docker ps` to check running containers
- `docker attach <CONTAINER ID>`

This will show a blank line with the cursor at the starting position if no one is accessing your app.

Next, refresh the app in the browser.

You should see an entry for the log is now generated.

```
Venkateshs-MacBook-Air:Online_Projects venkateshachintalwar$ docker attach 204
df32b9504
172.17.0.1 - - [03/Apr/2020 15:49:39] "POST / HTTP/1.1" 200 -
```

> Why is this critical and not recommended? If you accidentally press ctrl+c or any kill signal to get back to the shell, the application will stop.

2. In this method, we will use the earlier-used command to see container logs.

`docker logs -f <CONTAINER ID>`

This code above will print all the logs and listen for new activity.

```
 * Serving Flask app "app.py"
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deploy
ment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
172.17.0.1 - - [03/Apr/2020 18:13:39] "POST / HTTP/1.1" 200 -
```

This is safe because when you exit and get back to the shell, the app will continue running.