# Mesos/Marathon Assignment

## Description

This project implements a REST service with four methods that allow to list, create, get details and remove Docker-based http clients with no exposed ports.

## Requirements

In order to run this project, you will need a Python 3 client that can be executed as `python`, if you also have an associated pip client the required libraries can be installed with:

```
pip install -r requirements.txt
```

You wil also need a running Mesos/Marathon service, for example running the docker-based found in docker-compose.yml:

```
docker-compose up -d
```

If you wish, you can use any other Mesos/Marathon service, even a remote one but you'll have to update the data in settings.py.

## Basic Usage

Once all the requirements are fulfilled, you can start the REST service with:

```
sh launch_server.sh
```

The REST service will start running on port TCP 5000. To test the functionality of all four methods you could execute from terminal:

```
curl -X POST -H "Content-type: application/json" -d '{"id":
"myapp"}' http://localhost:5000/
curl http://localhost:5000/
curl http://localhost:5000/myapp/
curl -X DELETE http://localhost:5000/myapp/
curl http://localhost:5000/
```

## Functionality

The REST service is implemented in Flask with four methods, one for each desired functionality. Each method calls the respective endpoint of the Marathon API via an http client class called MarathonClient.

The method `create_application` differs from the other three in the sense that it makes use of the application described in basic.json allowing only to modify the application id.

All methods return the payload and status code sent by the Marathon API.

# Estimated Effort

The time used between investigating the Mesos/Marathon service, developing the REST service and documenting the project has been approximately six and a half hours.