Instructions

For the first assignment, you will create a REST API service that is able to differentiate between GET and POST requests and which responds to requests for the resources 'hello' and 'test'.
You will use Docker to create an image which must expose a web server at port 8080 that implements the REST interface below.

Please note that for this project, team size is not relevant, but will be in the future. It is alright to submit this assignment as an individual or as a member of a team of up to four people. From assignment 2 onward, we require that you be a part of a team of 3-4 people in order to receive full credit for the assignment. This eases the burden of work for you, as the projects increase in difficulty, and for us, the TAs, as we grade fewer projects and can give feedback and grades sooner rather than later.

You will need to submit a new repo for each assignment. Each repo **must** contain a "members.txt" file and a "contribution-notes.txt" file, which lists the members and their contributions respectively. The format for these files are given below. Please read them carefully and adhere to the instructions exactly for full credit.

REST API

Hello:
Method: GET
Resource Identifier: http://localhost:8080/hello
Response message-body: Hello world!
Status code: 200

Method: POST
Resource Identifier: http://localhost:8080/hello
Should not support this method.
Status code returned should be 405.

Test:
Method: GET
Resource Identifier: http://localhost:8080/test
Response message-body: GET request received
Status code: 200

Method: POST
Resource Identifier: http://localhost:8080/test
Query parameters: msg

Parameter format: Alphanumeric
Response message-body: POST message received: <msg>
Call example: http://localhost:8080/test?msg=ACoolMessage
Example response: POST message received: ACoolMessage
Status code: 200

Building/Testing your container:

We will formally post test scripts closer to the due date (10/12/18), but ensure your container builds, runs, and responds to the call examples for the methods above during development until then.

It is critical that you run the test scripts before submitting your assignment, as the behavior should be predictable, and the tests we provide are similar to the further tests we will run on our side during grading.

Instructions how to submit your homework:

0. Preconditions - You have a project folder on your local machine.

1. Create a file members.txt. It will contain one member per line, member being the ucsc id. e.g.
members.txt:
palvaro
elaolive

2. Create a contribution-notes.txt file. This can be empty until the moment of final submission.
contribution-notes.txt, for example:
palvaro: created all the docker related file
elaolive: wrote the web server app in python

3. Create a local git repo (git init, add, commit)

4. Create a bitbucket account https://bitbucket.org/account/signup/

5. Create a team (**NOTE**: Do NOT create a repo before creating a team, it will be 'wasted')
   - Team ID needs to be unique at bitbucket, but prefix it with cmps128 e.g. cmps128CoolTeamNameHere.
   - Add your team members to the repo (if applicable)
   - **Add Cmps128teachingstaff to the repo**
6. Create a repository for the team. Ensure that it is private.

7. Commit files to the repository

```
cd /path/to/my/repo
git remote add origin <repo>
```

for example:

```
git remote add origin (url of repository)
git push -u origin --all # pushes up the repo and its refs for the first time
git push -u origin --tags # pushes up any tags
```

8. Create your Dockerfile, create your source folder for the server that will run in the docker image. Test output. Commit early, commit often.

9. When you are satisfied with your solution (or when you nearly run out of time :'( ), submit your latest commit id to the Google form that we will post closer to the due date.

To evaluate your homework submission, a TA will create a docker image using the Dockerfile in your project directory. It will be tested by sending GET and POST request to 8080 port. We will be checking that the correct response and status are sent back from your app.