CSE 138: Distributed Systems

Fall 2019 - Assignment #1

Assigned: Friday, 10/04/19
Due: Friday, 10/11/19

Instructions

General

- You must do your own individual work and submit this assignment as an individual.
- You will use **Docker** to create a container that runs a **RESTful** (provides a REST interface) web server.
- Your `RESTful` web server must respond to `GET` and `POST` requests for the end points `/hello` and `/check`.

Building and testing your container

- We provide a test script `test_assignment1.py` that you **should** use to test your work before submitting your assignment.
- The provided tests are similar to the tests we will use to evaluate your submitted assignment.

Requirements for Submission

- A GitHub account (https://github.com/join) associated with your UCSC email address.
  - GitHub provides free perks to students (https://education.github.com/pack)
- A private GitHub repository (https://help.github.com/en/articles/create-a-repo).

Submission workflow

- Sign in to your GitHub account.
- Create a private repository. For convenience, we recommend it be named `cse138_assignment1`.
- Invite `ucsc-cse138-staff` as a collaborator to your repository.
  - https://help.github.com/en/articles/inviting-collaborators-to-a-personal-repository
- Clone your repository on your machine.
- Add your project files to your repository.
- Create a `Dockerfile`, which defines how to create your container, at the top level of your project directory.
- Commit your files and push your commits to the master branch on GitHub. We recommend committing and pushing often.
- Submit your CruzID, repository URL, and the commit ID (aka commit hash) to be evaluated here: https://forms.gle/WA3vYERUYCCRjbTCA

  - https://help.github.com/en/articles/github-glossary#commit
  - The commit timestamp **must be no later than 10/11/2019 11:59 PM PDT**
  - The google form must be submitted within a reasonable time of the due date (preferably 10 minutes).

Evaluation and grading

- Course staff will evaluate your assignment using the Dockerfile in your repository to create a Docker image:

```
docker build -t <name-of-project-image> <path-to-root-of-project-code>
docker run -p 8081:8081 <name-of-project-image>
```

We will test your project by sending GET and POST requests to port 8081. We will be checking that the correct response and status are sent back from your web server.

REST API

Description

Your REST web server must have two endpoints: `/hello` and `/check`.

The endpoint, `/hello`, accepts a GET request (with no query string) and returns the string `Hello, world!` with HTTP status code 200. An example that shows both the response and the status code:

```
$ curl --request GET --write-out "\n%{http_code}\n" http://localhost:8081/hello
Hello, world!
200
```

The endpoint, `/hello`, responds to a POST request with a 405 error:

```
$ curl --request POST --write-out "\n%{http_code}\n" http://localhost:8081/hello
This method is unsupported.
405
```

The endpoint, `/check`, accepts a GET request (with no query string) and returns the string `GET message received` with status code 200:

```
$ curl --request GET --write-out "\n%{http_code}\n" http://localhost:8081/check
GET message received
200
```

The endpoint, `/check`, also accepts a POST request with a `msg` query parameter and returns `POST message received: <msg>` with status code 200, where `<msg>` is the value of the `msg` query parameter:

```
$ curl --request   POST                    \
       --write-out "\n%{http_code}\n"      \
       http://localhost:8081/check?msg=foo
POST message received: foo
200
```

The endpoint, `/check`, responds to a POST request with no `msg` query parameter with status code 405:

```
$ curl --request POST --write-out "\n%{http_code}\n"  http://localhost:8081/check
This method is unsupported.
405
```