Assignment 3: Container Specification for SelfieLessActs

In this assignment, your now monolithic REST service should be split up into two **microservices** - one catering to the user management, and another catering to the act management. These two microservices should be started in separate docker containers, running on one AWS instance. The microservices will talk to each other via their respective REST interfaces.

You will reuse the APIs you built in last assignment.

1) On the User management microservice, you must have APIs 1 and 2 (Add user and Remove user). In addition you must implement one more API on this microservice:

List all users
Route: `/api/v1/users`
HTTP Request Method: `GET`
Relevant HTTP Response Codes: `200, 204, 405`
Request: `{}`
Response:

```
[
      "username1",
      "username2",
      ...
]
```

2) On the Acts management microservice, you must have the rest of APIs. This microservice must call the **List all users** API on the previous microservice in order to verify a given username actually exists. (Eg: when an act is being uploaded).

You must build the microservices in the following way:

- Both the microservices will be in a single AWS EC2 instance. It must be an Ubuntu image (at least version 14.04).
- Within your Ubuntu AWS EC2 instance:
    - You must have a valid user with sudo access enabled. When your evaluator asks, you must give them this username.
    - You must have installed and configured an SSH server on your instance so that the above user can login with their password. Make sure port 22 is accessible from the external IP.
    - You must give the RSA private key file (.pem) used for SSH to your evaluators.
    - If not already present, the AWS instance must have the `tcpdump` command line tool installed.

    - You must map the web server port inside the each of the containers (it's usually 80) to another port on to the localhost of the AWS instance.
    - For the acts microservice, map web server port within the container (usually 80) to localhost 8000.
    - For the users microservice, map web server port within the container (usually 80) to localhost 8080.
    - You must expose both of these ports on the external IP of the AWS instance as well (using security groups).

- Install the `docker-ce` and `docker-ce-cli` Ubuntu packages on your AWS instance.
- You must embed both of your web servers (and databases if any) inside Docker containers within the AWS EC2 instance.
- You must use `alpine:3.7` as the base image for each container.
- For the user management container, you must name the image "`users`"
- For the acts management container, you must name the image "`acts`"
- These images must at least be tagged as "`latest`" locally on the AWS instance.
- In your Dockerfiles, use the `ENV` parameter to have the environment variable `TEAM_ID=CC_123_456_789` defined within your containers.
- Create an account on Docker Hub (https://hub.docker.com/).
- You must also publish your docker images (one for each container) to Docker Hub registry. You must give the names of these images to your evaluators. These names will be prefixed with your Docker Hub ID, eg: `dockerId/acts:latest` and `dockerId/users:latest`

Resources:

- Docker overview
    - https://docs.docker.com/engine/docker-overview/
- Install Docker on your Ubuntu AWS instance
    - https://docs.docker.com/install/linux/docker-ce/ubuntu/#install-docker-ce
- How to build your own docker image
    - https://docs.docker.com/get-started/part2/
- Map container port to localhost of AWS instance
    - https://docs.docker.com/get-started/part2/#run-the-app
- Publish image to Docker Hub
    - https://hackernoon.com/publish-your-docker-image-to-docker-hub-10b826793faf
- What is Alpine?
    - https://alpinelinux.org/
- Alpine docker image
    - https://hub.docker.com/_/alpine
- Install packages on Alpine
    - https://www.cyberciti.biz/faq/10-alpine-linux-apk-command-examples/

Deadline: March 11, 2019

Marks : 5