Assignment 2: Container Specification for RideShare

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 ride 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",
            ...
]
```

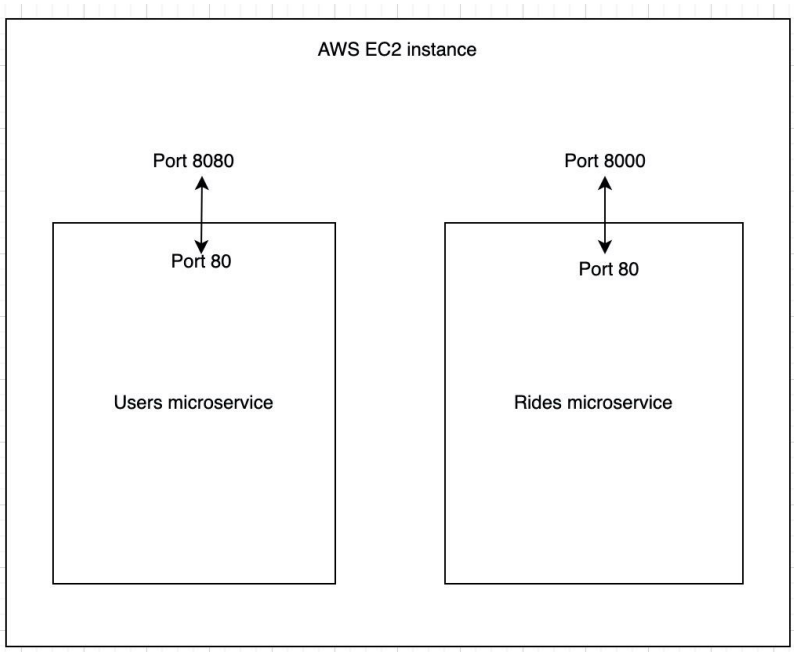
2) On the Rides management microservice, you must have the rest of APIs. This microservice must call the **List all users** API on the previous microservice in order to verify a given username actually exists. (Eg: when an ride is being created/joined).

3) You have to create another API, for clearing the db. This API will be implemented on both microservices.

Clear db
Route: `/api/v1/db/clear`
HTTP Request Method: `POST`
Relevant HTTP Response Codes: `200, 400, 405`
Request: `{}`
Response: `{}`

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 16.04).
- Within your Ubuntu AWS EC2 instance:
    - You must have a valid user with sudo access enabled. You will have to give the username for evaluation
    - 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 submit the RSA private key file (.pem) used for SSH
    - 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 if you have setup a proxy web server in the previous assignment) to another port on to the localhost of the AWS instance.
    - For the rides 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 (8000 and 8080) on the external IP of the AWS instance as well (using security groups).



- Install the `docker-ce` and `docker-ce-cli (docker-compose if needed)` 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 can use `any image` as the base image for each container.
- You can also use the existing images for “postgres”, “mongo”, “nginx” etc
- For the user management container, you must name the image “`users`”
- For the rides management container, you must name the image “`rides`”
- 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_NAME=CC_123_456_789` defined within your containers.
- You can use docker compose if you have multiple services, running. In that case, you will have one docker-compose file for each microservice.
- Each microservice must have their own database, shared database cannot be used.
- If using nginx/apache, each microservice must have their own web server.
- The database must be running in the container itself, you cannot connect to the database running on your host machine from your container.

- Any concern wrt the specifications must be brought up within 3 days of posting the assignment, after that it will not be possible to update/change the specifications.

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
- 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: Feb 27, 2019

Total Marks : 5

Marks Distribution:

1. Running 2 microservices with the given container name and tag and ENV variable – 2 marks
2. Exposing 8000 and 8080 on public ip – 1 mark
3. Implementing and calling the “List all users” API from the “rides” container – 1 mark
4. Ensuring previous APIs don’t break – 1 mark