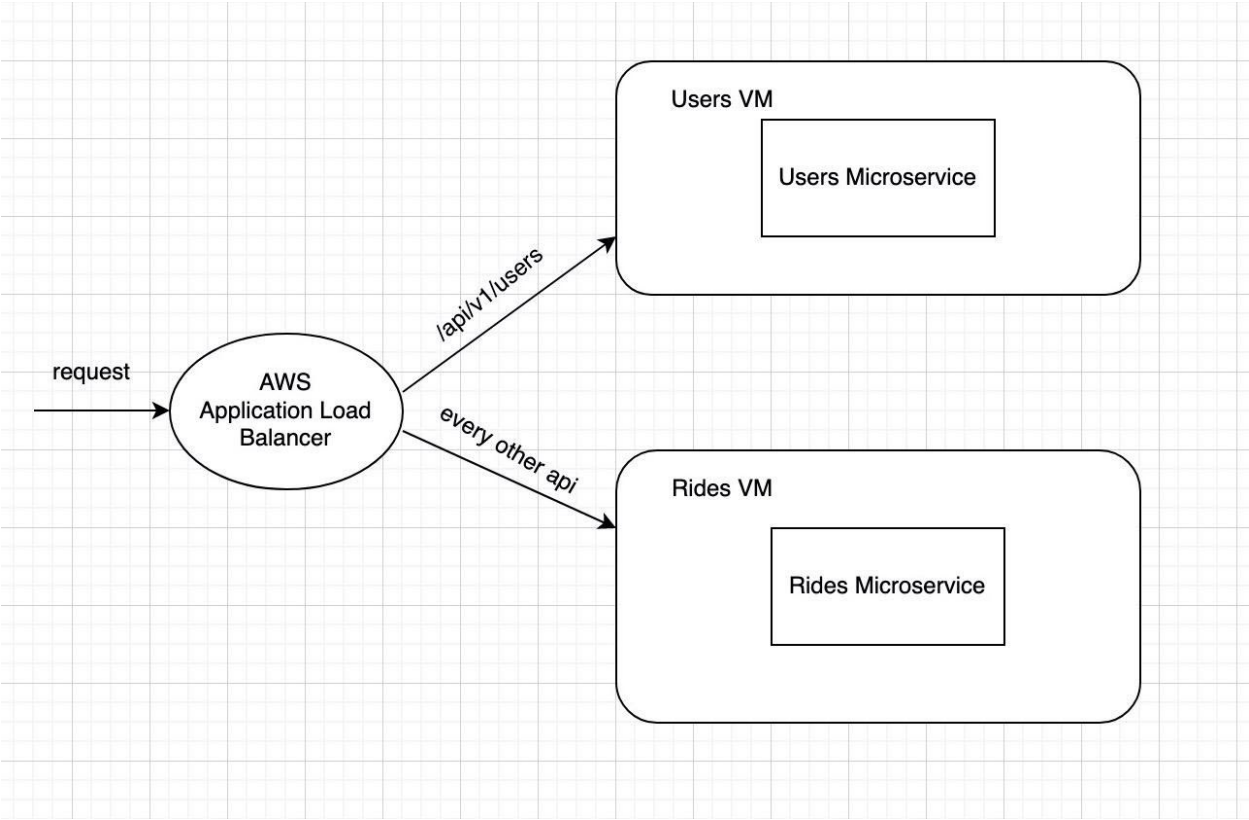
Assignment 3: Load Balancing for RideShare

In this assignment, you will put your two microservices (containers) into two different AWS EC2 instances. But you will also make them accessible from under the same public IP address and also the same port (80). This is only possible by using a load balancer that supports path-based routing. Hence, you'll create an AWS Application Load Balancer which will distribute incoming HTTP requests to one of the two EC2 instances based on the URL route of the request.



Some additional APIs to be defined for both microservices

Steps:

1. Create two EC2 instances (of same instance type as in previous assignment). Make port 22 and 80 accessible for both of them.
2. Place your *rides* container in one EC2 instance and *users* container in the other instance.
3. The web servers inside the containers must be accessible through the public IPs of the instances.
4. Note again that this time you must expose the APIs through port 80 of the EC2 instance IP address.
5. Create two AWS target groups, one for each instance.

6. Create an AWS Application Load Balancer with the following rules:
    a. If an incoming request's route matches `/api/v1/users`, then forward it to the *users* instance.
    b. For all other routes, forward the request to the *rides instance*.
7. Make sure the security group of the load balancer exposes ports 22 and 80 only.
8. This time as well, the *rides microservice* must make a call to the *users* microservice to check if a user exists. While making this request, make sure the HTTP request from the *rides instance* has the `Origin` header set to either the public IP address or public DNS name of the rides instance. This will be checked for by the testing script. This call should be made over the load balancer, not directly to the ip of the users instance.
9. You must add the following two APIs to both of the microservices/instances:
    a. **Get total HTTP requests made to microservice**

        Route: `/api/v1/_count`
        HTTP Request Method: `GET`
        Relevant HTTP Response Codes: `200, 405`
        Request: `{}`
        Response:
```
[ 603 ] // example, total number of HTTP
           requests made to only this
           microservice
```
        Comment:
        A call to this API must return the total number of HTTP requests made to only this microservice. If no requests are made yet, return 0.

    b. Reset HTTP requests counter

        Route: `/api/v1/_count`
        HTTP Request Method: `DELETE`
        Relevant HTTP Response Codes: `200, 405`
        Request: `{}`
        Response: `{}`
        Comment:
        A call to this API must reset the total number of HTTP requests made to only this microservice back to 0.

Note 1: The two APIs above, along with the existing clear db APIs, will be called on the public IP of each microservice directly, and not on the load balancer IP, as these APIs are microservice-specific.

Note 2: Calls to the two APIs above **<u>should not</u>** be counted towards the HTTP request count returned. Only count the requests made to the rest of the APIs (users, rides).

Note 3: Count API requests whether they failed and or were successful.

10. You must also add an extra API only to the rides microservice/instance:

Get total number of rides
Route: `/api/v1/rides/count`
HTTP Request Method: `GET`
Relevant HTTP Response Codes: `200, 405`
Request: `{}`
Response: `[ 258 ] // example, total number of rides`

Comment:
A call to this API must return the total number of rides created

This API will be called on the load balancer public IP, and must be routed to the rides instance.

Resources:

- About AWS Application Load Balancer
  - https://docs.aws.amazon.com/elasticloadbalancing/latest/application/introduction.html
- Tutorial on how to create AWS target groups and a load balancer with path routing
  - https://hackernoon.com/what-is-amazon-elastic-load-balancer-elb-16cdcedbd485

Deadline: 17/3/2020
Marks: 5

Marks Distribution:

- New APIs - 2 marks
- Setting up the load balancer - 1 mark
- Viva - 2 marks
- Submit a 1 page report as per following template

Honesty Policy

PES University follows an academic honesty policy that requires all students to ensure that they work on the assignments by themselves so that they can make get the best learning experience from these projects. While we understand and encourage students to discuss approaches and help in debugging, the work done must be original and not copied/outsourced.

Assignment Submission Report for assignments 1-3 : Rideshare on AWS 17CS 352

Team ID: CC_

SNo	Name	USN	Section

Assignment No	Submission ID	Public IP
1		
2		
3		Rides: Users:

Please fill the individual contribution by each team member for each assignment

	USN1:	USN2:	USN3:	USN4:
Assignment 1				
Assignment 2				
Assignment 3				