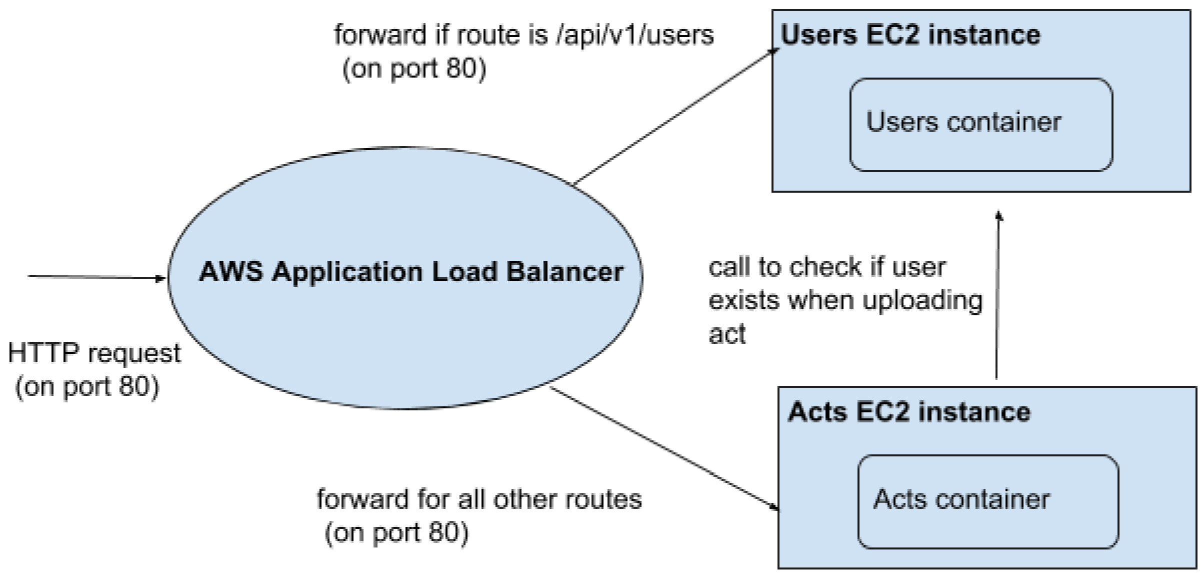
Assignment 4: Load Balancing for SelfieLessActs

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 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.


forward if route is /api/v1/users (on port 80)
Users EC2 instance
Users container
AWS Application Load Balancer
call to check if user exists when uploading act
HTTP request (on port 80)
Acts EC2 instance
Acts container
forward for all other routes (on port 80)

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 *acts* 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 *acts* instance.
7. Make sure the security group of the load balancer exposes ports 22 and 80 only.
8. This time as well, the *acts* 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

acts instance has the `Origin` header set to either the public IP address or public DNS name of the acts instance. This will be checked for by the testing script.

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 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, categories, acts).

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

10. You must also add an extra API only to the acts microservice/instance:
    Get total number of acts
    Route: `/api/v1/acts/count`
    HTTP Request Method: `GET`
    Relevant HTTP Response Codes: `200, 405`
    Request: `{}`
    Response: `[ 258 ] // example, total number of acts`
    Comment:
    A call to this API must return the total number of acts uploaded across all categories.

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

Deadline: 27/3/2019
Marks: 5

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