MODULARIZATION WORKSHOP WITH VIRTUALIZATION AND INTRODUCTION TO DOCKER AND AWS

Juan Sebastian Nieto Molina
March 2021

Abstract

This document is the solution of a guide that has to objective make a virtual machine in AWS with EC2 and make a container with Docker, using a load balancer wit REST service and using three nodes that save the data into a mongo database, this three objects are images of docker.

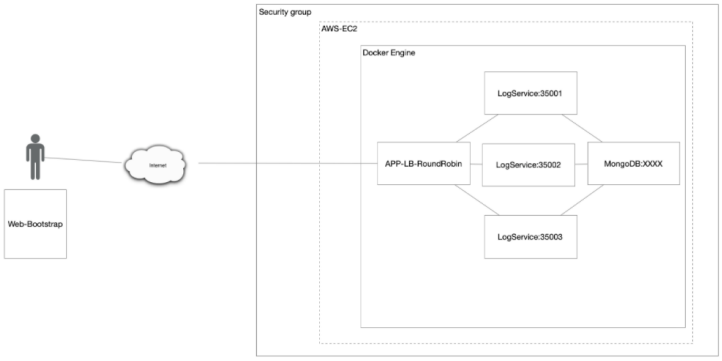
Contents

1 Introduction

This laboratory was made using the Spark micro-framework, a web client and a couple of REST services, the idea is to learn how to use Docker containers to consume resources such as databases, in this case MongoDB, we must create a load balancer that will consume the image of our project to be able to display 3 different nodes that consume the data we send from the web page and save it in the database, then it will send us the result of the updated table where the notes taken will be shown on the date on which were made.

2 Design

For this architecture, an AWS EC2 machine will be created, which will contain a couple of images from my Docker hub, and a Docker compose to be able to deploy the APP-LB-RoundRobin that will be our load balancer and will distribute the data entered from the application to one of the 3 nodes called LogService that what they will do is save this information in the database and will also return the updated data list to be able to display it. As you can see the largest group, the Security group, here we must enable the ports in AWS so that it can read the balancer and the mongo database. You can see better the architecture of this project in this image

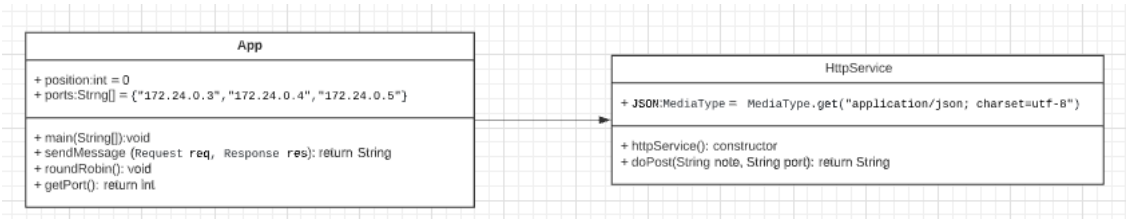


This project has two maven projects, the first is called RoundRobin and is where we have the load balancer, and our first docker image, that contain two post method, one in front, it call into the load balancer to make the call into the second post inside of our java code, to call the nodes, where we are going to store the data into the mongo database. All this happened into a EC2 virtual machine that deploy two docker images, that we can use because we got the permission of this ports.

You have to complete the path that is deployed in heroku to see the views correctly

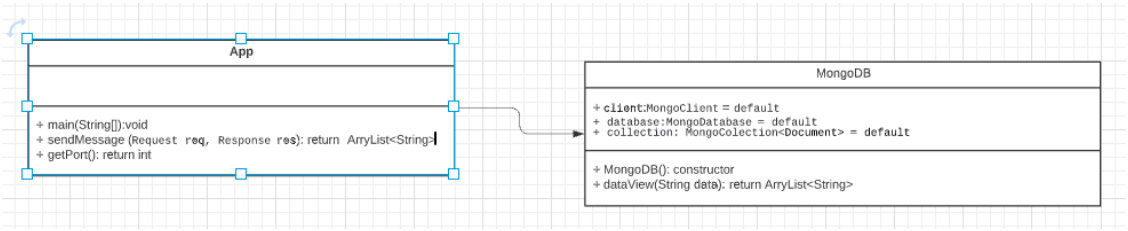
2.1 RoundRobin

This is the load balancer, here it receives data of a post method that is launch in front, we catch this data, and send into a class named HttpService, its work is make a post to a one of three nodes using round robing rule tho be storage into a database, when it got the request, App class send it to front, where where it map the data to deploy it into a table, here is the class diagram.



2.2 LogService

Here the App class receives data from a method post that was did it into the RoundRobin project, call the mongo database named logService and insert the notes into the collection named userNotes, how it use docker-compose it was built with a volumes that storage the information and when we deploy it in another it preserves the data, then it return the collection with the last 10 elements inserted, here is the class diagram.

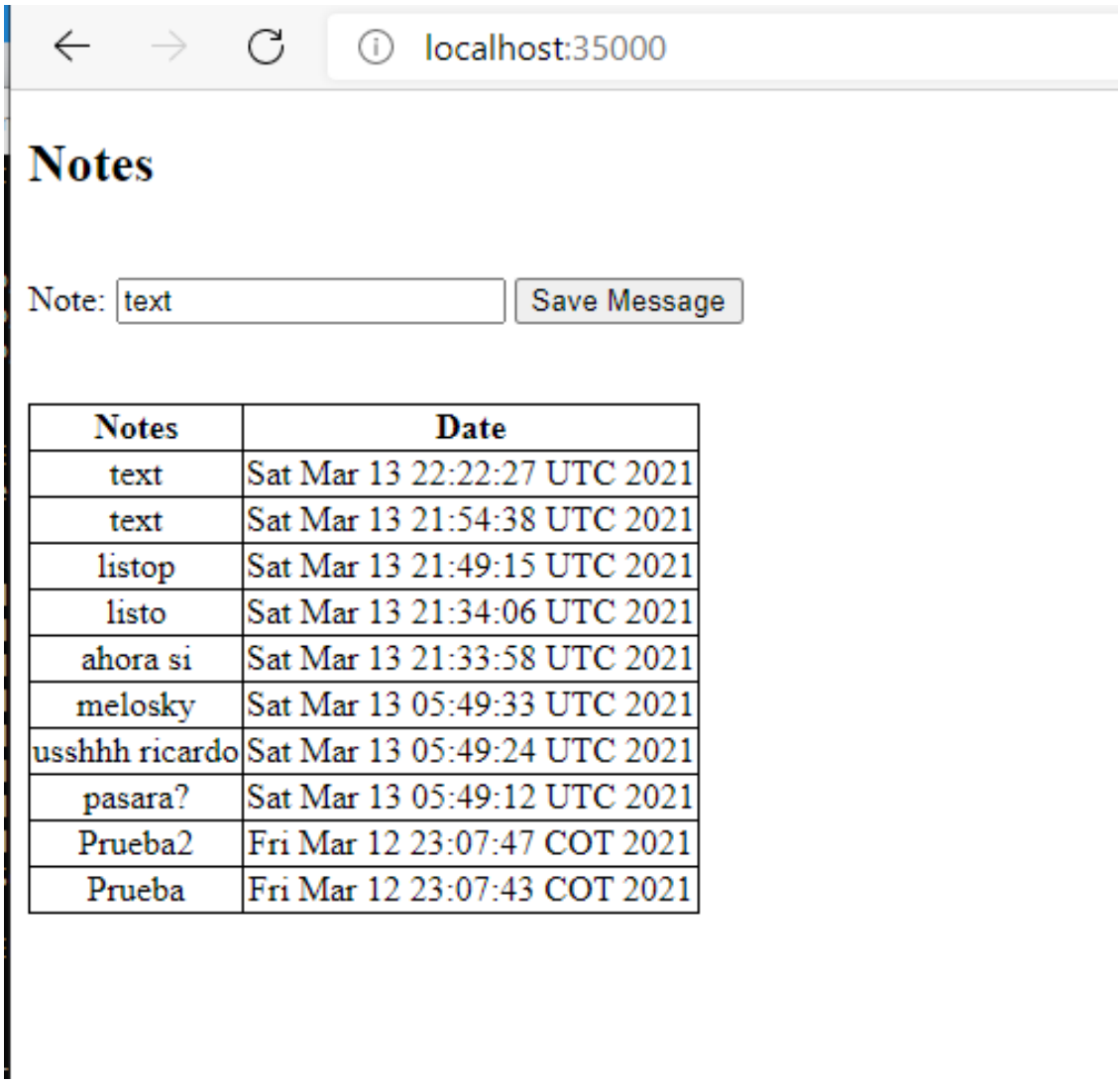


2.3 Test

When we used this project locally, we have to generate the docker images, for this reason we have to create into each maven project a image with this commands, here we can see that each project have a Dockerfile, is important that insert the command ”mvn clean package” into each project because it generate a target folder that we use to deploy the project, so for the images, we need to insert this commands because the docker-compose.yml use it:

- docker build –tag logservice .
- docker build –tag balancer .
- docker-compose up -d

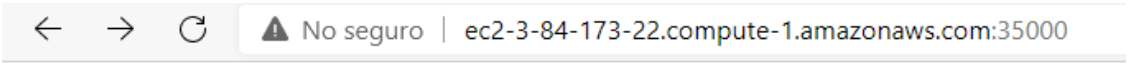
  Locally we can see this view.



In AWS the logic is the same, but, is necessary modify the docker-compose, because it use images that come from dockerhub, it was created before, with docker build, and it was pushed to mi repository. That is how it have to be.

```
[ec2-user@ip-172-31-36-103 lab5]$ cat docker-compose.yml
version: '2'

services:

  balancer:
    image: juanmol/balancer
    container_name: balancer
    networks:
      arep-lab5_default:
        ipv4_address: 172.24.0.2
    ports:
      - "35000:6000"

  log1:
    image: juanmol/logservice
    networks:
      arep-lab5_default:
        ipv4_address: 172.24.0.3

  log2:
    image: juanmol/logservice
    networks:
      arep-lab5_default:
        ipv4_address: 172.24.0.4

  log3:
    image: juanmol/logservice
    networks:
      arep-lab5_default:
        ipv4_address: 172.24.0.5

  db:
    image: mongo:3.6.1
    container_name: db
    volumes:
      - mongodb:/data/db
      - mongodb_config:/data/configdb
    networks:
      arep-lab5_default:
        ipv4_address: 172.24.0.6
    ports:
      - 27017:27017
```

And then that was the result.



Notes

Note: lab terminado Save Message

Notes	Date
lab terminado	Sat Mar 13 22:23:42 UTC 2021
melosky	Sat Mar 13 22:15:02 UTC 2021
por fin	Sat Mar 13 22:14:55 UTC 2021
funciona bien	Sat Mar 13 22:14:51 UTC 2021
llegada a la base de datos	Sat Mar 13 22:14:46 UTC 2021
reinicio	Sat Mar 13 22:14:38 UTC 2021
ya llego a la base de datos	Sat Mar 13 22:14:29 UTC 2021
text	Sat Mar 13 22:14:06 UTC 2021
text	Sat Mar 13 22:14:05 UTC 2021
text	Sat Mar 13 22:14:05 UTC 2021

3 Conclusion

Docker is a power full tool that allow us maintain software more easily with the use of containers and images, in this case use the images of a mongo database and 4 more, one of the load balancer and three of the nodes, deploy it locally and easily deploy it in AWS with EC2

4 References

[1] Benavides, P. L. D. (2021). TALLER DE DE MODULARIZACIÓN CON VIRTUALIZACIÓN E INTRODUCCIÓN A DOCKER Y A AWS, Bogotá, Colombia.

[2] Sebastian Nieto (2021). Tomado de: https://github.com/sebastianNietoMolina/AREP-lab5