Simon Gomez - Microservices

Play with docker

Using existing image

1. Launch a redis database with docker (https://hub.docker.com/_/redis)
2. log on the database and insert data

   ```
   redis-cli
   SET firstname "Randal"
   SET lastname "Munroe"
   SET URL "https://xkcd.com/1988/" EX 60
   ```

3. query the data you ve just inserted

```
redis-cli

GET firstname
GET URL
// wait 1 minute
GET URL
```

1. exit te container
2. stop the container
3. create and run another redis container
4. run the same request as point 3 in the new container
5. Are the data lost ?
6. display information about the first container : where are the data ?

TIPS :

- inside the container, redis-cli is the command to connect
- docker exec can help
- docker inspect can provide information
- try to connect to Redis from inside and outside the container

Image creation

docker is great, how do i package my Motus in docker ?

1. Create a Dockerfile
2. Find a suitable base image (base linux or base node)
3. Install the right dependencies
4. Run the right command to make it work locally!

Docker Compose - step 1

the great recipe goal is to have a Redis database and Redis insight interface running at once. Redis insight is a web UI to connect to Redis, allowing to manage the database.

Write a Docker-Compose with two services

- redis for the db
- redisinsight for the interface

Redis insight

1. Pull up the redisinsight images
2. setup a port mapping: 8001
3. Make it dependant on: redis service
4. Always restart service if anything happens
5. Create a volume called /db and attach it at this location: ./data/redisinsight it will store the database configuration for your redis insight

Redis

1. Pull up the latest redis images
2. Setup a port mapping 6379
3. Create a volume called /data and attach it at this location: ./data/redis

Test your interface on http://localhost:8001

Docker Compose - step 2

Let’s apply this to the motus project

1. Create a Docker-compose file in your motus App folder
2. import your docker image
3. Run your docker-compose file

some load balancing

1. add an haproxy loadbalancer in your project via a dockerfile
2. setup the docker-compose to have two instance of your app running and a loadbalancer up front to distribute request between the two instances

You will improve this Docker-compose later on

Published with GitHub Pages