Tampereen yliopisto
Tampere University

Kari Systä

COMP.SE.140 – NGINX hands on

Version history

V1.0 23.10.2024	First internal version for course staff
V1.1 24.10.2024	Staff feedback
V1.2 28.10.2024	Corrected fixes pointed out by the students
V1.3 30.10.2024	Still one mistake found.

Synopsis

The task is to add nginx[1] gateway and Web interface in front of your previous exercise (docker compose hands on). The gateway implements a basic authentication and load balancing.

Learning goals

The students will learn

- about load-balancing, access control and others gateway functionalities,
- hands-on about nginx and how it can be integrated to application

Task definition

Services 1 and 2 should run similarly as in the previous exercise, but

- Service 1 sleeps for 2 seconds after responding to the request. During that time the service cannot respond to next request.
- There are three instances of Service1

Nginx is added as a new service to the docker compose and listens in port 8198. 8198 is now the only port that is exposed outside. Nginx acts as Web server and a browser will be used for testing instead of curl.

Load-balancing functionality is added to nginx to distribute requests to all three incarnations of service1. The default round-robin algorithm is ok.

Basic authentication is added to nginx and one user with password is initialized.

The behaviour of the system is the following:

1. When a browser enters the system (URL http://localhost:8198) a login page is given.
2. If user enters valid credentials (username + password) a new page is shown: buttons "REQUEST", "STOP", and an empty text area.
   In case of wrong credentials, either an error is shown or the credentials are just asked again.
3. If the user click the REQUEST button, a request is sent to service1 (one of them as set by the load balancer). The response is shown on the text area. The new text replaces the possible old content in the text area.
4. When the user presses "STOP" the system closes down, all containers close down, and the process execution "docker compose" process exits (i.e. control is returned to command shell). Obviously, nginx also exits. (In real life this behaviour may not be recommended, but help in assessment of the task. And is also challenge to our student.)

[1] https://nginx.org

Kari Systä

This is known to be hard, and you are free to select the way this is implemented. If you fail, the reduction in points is not major.

Deadlines
See course moodle.

Submitting for grading
After the system is ready the student should return (in the git repository – in branch "exercise4").

- Content of all Docker-files and docker-compose.yaml
- Source codes of the applications
- Output of "**docker container ls**" and "**docker network ls**" (executed when the services are up and running.) in a text file "docker-status.txt"
- File "login.txt" containing the username and password.

Please do not include extra files in the repository.
These files are returned with some git service. Courses-gitlab repositories can created to course-gitlab if you request one. Any git-repo that the staff can access without extra effort is ok.
You should prepare your system in a way that the course staff can test the system with the following procedure (on Linux):

```
$ git clone -b exercise4 <the git url you gave>
$ docker compose up --build
… wait for 10s
A browser is pointed at localhost:8198
$ docker compose down
```

Hints

https://medium.com/@aedemirsen/load-balancing-with-docker-compose-and-nginx-b9077696f624

https://medium.com/@nickjabs/secure-your-app-with-nginx-basic-authentication-94db2d28d154