HAUTE ÉCOLE
D'INGÉNIERIE ET DE GESTION
DU CANTON DE VAUD
www.heig-vd.ch

June 2018

Laboratory Report

HTTP Infrastructure

RES course

Prof. Olivier Liechti

Max Caduff
Doriane Tedongmo

Step 1:

The objective of this first part is to implement a static HTTP Apache server, using existing Docker images.
We used the Docker image PHP 7.0 (https://hub.docker.com/_/php/) as a base, and the free bootstrap template « agency », slightly modified to display custom content. The website is located in the site/ folder, which is copied in the /var/www/html/ folder of the Container.

The content of the Dockerfile to create this image is the following:

```
FROM php:7.0-apache

COPY content/ var/www/html/
```

To build the image, we run the following command from the folder containing content/ and the Dockerfile:
« docker build -t res18/1-static docker-images/1-static/ »
Then, to run the container, we should give it the local port to communicate with it (here 2345) and the port of the container on which it should listen (here 80). Hence the next command:
« docker run --name static1 -p 2345:80 res18/1-static »

The address to access the generated website is: http://demo.res.ch:2345/ , it requires to add the following line in the host file (/etc/hosts) from the machine: "192.168.99.100 demo.res.ch"

In order to intact with the machine, we can launch a bash instance when running it with:
« docker run -i -p 2345:80 res18/1-static /bin/bash » or « docker exec -it static1 /bin/bash » to avoid running another container.

we can then explore the filesystem and find the location of the Apache config files, which can be found under:
/var/lib/apache2/conf

Step 2:

In this part we needed to implement a dynamic server in parallel to the Static server from Step 1, based on a node.js image to be able to write the code in javascript, and express.js to manage the server behavior.

To initialize node and install the dependencies we need, the following commands were run from the src folder:

```
npm init
npm install --save chance
npm install --save express
```

The files to run the server are located under src/, they are copied in the /opt/app folder of the vm, and the main file (index.js) is launched from there with the command in the Dockerfile, which is the following:

```
FROM node:6.10

COPY src /opt/app

CMD ["node", « opt/app/index.js »]
```

The command to build the image is:
« docker build -t res18/2-dynamic docker-images/2-dynamic/»

The command to run the container is:
« docker run -p 3456:3000 res18/2-dynamic »
We need once again to map the correct ports of the machine, since the container is listening on his port 3000.

When the server gets a request, he generates between 5 and 10 random animals with a job and a country of origin, and sends them as JSON.

Step 3 :

In this step we implement a reverse proxy, which will redirect traffic to various other servers.

Still using the PHP 7.0 image as a base, the reverse proxy virtual hosts are configured as follow :

The configuration file, 001-reverse-proxy.conf, is first hardcoded with the IP adresses automatically attributed to the containers when launched. Those are found by first running a bash on the server-hosting containers with « docker exec -it <container-name> /bin/bash » and then « docker inspect <container-name> l grep -i ipaddress ».

001-reverse-proxy.conf :

```
<VirtualHost *:80>

        ServerName demo.res.ch

        ProxyPass             "/hire/" "http://172.17.0.3:3000/hire/"
        ProxyPassReverse      "/hire/" "http://172.17.0.3:3000/hire/"

        ProxyPass             "/" "http://172.17.0.2:80/"
        ProxyPassReverse      "/" "http://172.17.0.2:80/"

</VirtualHost>
```

Any request to the url /hire/ will be redirected to the dynamic container, requests to / will be redirected to the static container, and any other request will be redirected to nothing since we kept the original configuration file.

To run the static server : «docker run -d --name staticSrv res18/1-static »
The attributed ip address is :

"IPAddress": "172.17.0.2",

To run the dynamic server : « docker run -d --name dynamicSrv res18/2-dynamic »
The attributed ip address is :
"IPAddress": "172.17.0.3".

Dockerfile of reverse proxy:

```
FROM php:7.0-apache

COPY conf/ /etc/apache2/

RUN a2enmod proxy proxy_http
RUN a2ensite 000-* 001-*
RUN service apache2 restart
```

then: « docker build -t res18/3-rp docker-images/3-reverse-proxy/ »
and « docker run -p 4567:80 res18/3-rp »
to access both websites: http://demo.res.ch:4567/ and http://demo.res.ch:4567/hire/

Step 4 :

In this step we need to make the static and dynamic communicate, with the server static sending ajax requests regularly to the dynamic server to update a part of the page.

A javascript function (hireAnimals.js) was added in the js folder of the static site, as well as some code in index.html:
to call the function updating content every 5 sec:
<script src="js/hireAnimals.js"></script>

to do the request and display the first animal sent:

```
<script>
        function loadAnimals () {
                $.getJSON(  "/hire/",
                 function (animals) {
                        console.log(animals);
                        var message = "Click to get a new pet!";
                        if (animals.length > 0)  {
                            message = animals[0];
                        }
                        $(".btn-xl").text(message);
          });
        };
</script>
```

We need to rebuild the static image:
« docker build -t res18/4-ajax-on-static docker-images/1-static/ »
run:
« docker run -d --name staticAjaxSrv res18/4-ajax-on-static »

the other containers have no modifications.

Step 5:

The goal of this step was to create a way to give the ip adresses of the static and dynamic servers to the proxy without having to rebuild the image each time. I found a way to do this without even having to restart the container, with a simple script to execute on the reverse proxy:

this script is the following:

```
#!/bin/bash
# this script should be called with 2 arguments, the first being the static ip and the second being the dynamic ip. (should include ports, example below.)

echo "<VirtualHost *:80>

        ServerName demo.res.ch

        ProxyPass               /hire/ http://$2/hire/
        ProxyPassReverse        /hire/ http://$2/hire/

        ProxyPass               / http://$1/
        ProxyPassReverse        / http://$1/
</VirtualHost>" > /etc/apache2/sites-available/001-reverse-proxy.conf

service apache2 reload
```

The Dockerfile of the RP is modified as follows to include the script:

```
FROM php:7.0-apache

COPY conf/ /etc/apache2/
COPY changeAddresses.sh /bin/

RUN a2enmod proxy proxy_http
RUN a2ensite 000-* 001-*
RUN service apache2 restart
```

Rebuild the RP:
« docker build -t res18/5-rp-dynamic docker-images/3-reverse-proxy/ »

run:
« docker run -d --name dynamicRP -p 4567:80 res18/5-rp-dynamic »

launch script:
« docker exec -it dynamicRP /bin/changeAddresses.sh 172.17.0.3:80 172.17.0.2:3000 »

(the 001 configuration file wasn't modified, so the script needs to be run at least once to get the RP functioning well.)

Step 6:

This step aims to enable multiple static and multiple dynamic servers handled by the RP, which will distribute the traffic to them with apache load-balancing abilities. (in round-robin mode)

The script from step 5 was modified to define clusters of static and dynamic servers, and take more arguments: (see below)

```
#!/bin/bash
# this script should be called with 3 or more arguments, the firsts being the static container's ips, followed by the dynamic container's ips (should include ports). The last argument is the number of static servers, the other half being deduced from the args num.


echo "<Proxy balancer://staticCluster>
" > txt
for ((i=1;i<= ${!#};i++));
do
  echo "          BalancerMember http://${!i}" >> txt
done

echo "
</Proxy>

<Proxy balancer://dynamicCluster>
" >> txt
for ((i=$((${!#} + 1));i<$#;i++));
do
  echo "          BalancerMember http://${!i}" >> txt
done

echo "
</Proxy>

<VirtualHost *:80>

        ServerName demo.res.ch

        ProxyPass               /hire/ balancer://dynamicCluster/hire/
        ProxyPassReverse    /hire/ balancer://dynamicCluster/hire/

        ProxyPass               / balancer://staticCluster/
        ProxyPassReverse    / balancer://staticCluster/

</VirtualHost>" >> txt

cat txt > /etc/apache2/sites-available/001-reverse-proxy.conf

rm txt

service apache2 reload
```

The Dockerfile was modified to enable load balancing:

```
FROM php:7.0-apache

COPY conf/ /etc/apache2/
COPY changeAddresses.sh /bin/

RUN a2enmod proxy proxy_http proxy_balancer lbmethod_byrequests
RUN a2ensite 000-* 001-*
RUN service apache2 restart
```

The static and dynamic servers were modified to display their IP adresses to identify them.

Build and run RP, stat. and dynam. containers

```
docker build -t res18/6-rp-load-bal docker-images/3-reverse-proxy/
docker run -d --name dynamicRP-LB -p 4567:80 res18/6-rp-load-bal
docker build -t res18/6-dynamic docker-images/2-dynamic/
docker run -d --name dynamicLB1 res18/6-dynamic
docker run -d --name dynamicLB2 res18/6-dynamic
docker build -t res18/6-static-lb docker-images/1-static/
docker run -d --name staticLB1 res18/6-static-lb
```

docker run -d --name staticLB2 res18/6-static-lb
docker run -d --name staticLB3 res18/6-static-lb

then, inspect containers to get IP addresses, and launch script with rights arguments:

« docker exec -it dynamicRP-LB /bin/changeAddresses.sh 172.17.0.3:80 172.17.0.4:80 172.17.0.5:80 172.17.0.6:3000 172.17.0.7:3000 3 »

(the 001 configuration file was still not modified, so the script needs to be run at least once to get the RP functioning well.)

Step 7:

The goal of this step is to implement sticky sessions to have clients served by the same static server, while the dynamic ones are still in round-robin mode. One again, the script was modified to use apache built-in sticky session management.

script:

```
#!/bin/bash
# this script should be called with 3 or more arguments, the firsts being the static container's ips, followed by the dynamic container's ips (should include ports). The last argument is the number of static servers, the other half being deduced from the args num.


echo 'Header add Set-Cookie "ROUTEID=.%{BALANCER_WORKER_ROUTE}e; path=/" env=BALANCER_ROUTE_CHANGED
<Proxy balancer://staticCluster>
' > txt
for ((i=1;i<= ${!#};i++));
do
  echo "          BalancerMember http://${!i} route=$i" >> txt
done

echo "
        ProxySet stickysession=ROUTEID
</Proxy>

<Proxy balancer://dynamicCluster>
" >> txt
for ((i=$((${!#} + 1));i<$#;i++));
do
  echo "          BalancerMember http://${!i}" >> txt
done

echo "
</Proxy>

<VirtualHost *:80>

        ServerName demo.res.ch
```

```
        ProxyPass               /hire/ balancer://dynamicCluster/hire/
        ProxyPassReverse        /hire/ balancer://dynamicCluster/hire/

        ProxyPass               / balancer://staticCluster/
        ProxyPassReverse        / balancer://staticCluster/

</VirtualHost>" >> txt

cat txt > /etc/apache2/sites-available/001-reverse-proxy.conf

rm txt

service apache2 reload
```

the RP Dockerfile was also modified:

```
FROM php:7.0-apache

COPY conf/ /etc/apache2/
COPY changeAddresses.sh /bin/

RUN a2enmod proxy proxy_http proxy_balancer lbmethod_byrequests headers
RUN a2ensite 000-* 001-*
RUN service apache2 restart
```

build and run:

```
docker build -t res18/7-rp-lb-sticky docker-images/3-reverse-proxy/

docker run -d --name dynamicRP-LB-sticky -p 4567:80 res18/7-rp-lb-sticky

docker exec -it dynamicRP-LB-sticky /bin/changeAddresses.sh 172.17.0.3:80 172.17.0.4:80 172.17.0.5:80 172.17.0.6:3000 172.17.0.7:3000 3
```

(and the 001 configuration file was still not modified, so the script needs once again to be run at least once to get the RP functioning well.)