Containerization Technologies – TD 2

Create a HTML file

First, we create the HTML file that our web server will use. We simply prompt a message on this file, named ‘index.html’:

Containerization Technologies > TD 2 > index.html > html

```html
<!DOCTYPE html>
<html>

<head>
    <title>Simple Web Server</title>
</head>

<body>
    <h1>Hello from the Web Server!</h1>
</body>

<footer>
    <p>Containerization Technologies</p>
</footer>

</html>
```

Create a Dockerfile for nginx

This Dockerfile is a set of instructions for building our Docker image. Our HTML file is copied into the nginx HTML directory, and using the port 80, the nginx server is started:

Containerization Technologies > TD 2 > Dockerfile.nginx > ...

```
# use the nginx image
FROM nginx:latest

# copy the HTML file created before to the nginx default HTML directory
COPY index.html /usr/share/nginx/html

# expose port 80
EXPOSE 80

# start nginx server
CMD ["nginx", "-g", "daemon off;"]
```

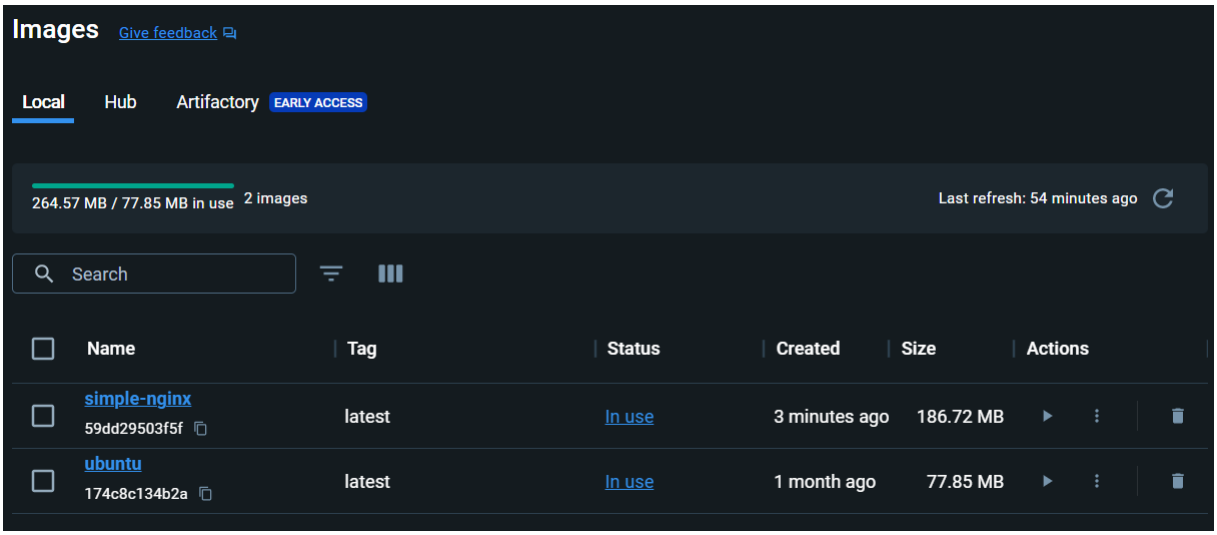
Build and run the nginx container

Now we build the container using the Dockerfile created before, with the latest nginx image, etc… Note that, from here, the steps are similar to the previous practical work.

The command: '**docker build -t simple-nginx -f Dockerfile.nginx .**'.

```
PS C:\Users\Loeva\OneDrive\Bureau\ESILV\A4 cycle ingé DIA\Semestre 8\~ programmation\Containerization Technologies\TD 2> docker build -t simple-nginx -f Dockerfile.ngin
x .
[+] Building 146.2s (8/8) FINISHED                                                                                                          docker:default
 => [internal] load .dockerignore                                                                                                                   0.0s
 => => transferring context: 2B                                                                                                                     0.0s
 => [internal] load build definition from Dockerfile.nginx                                                                                          0.0s
 => => transferring dockerfile: 290B                                                                                                                0.0s
 => [internal] load metadata for docker.io/library/nginx:latest                                                                                     3.5s
 => [auth] library/nginx:pull token for registry-1.docker.io                                                                                        0.0s
 => [internal] load build context                                                                                                                   0.0s
 => => transferring context: 257B                                                                                                                   0.0s
 => [1/2] FROM docker.io/library/nginx:latest@sha256:4c0fdaa8b6341bfdeca5f18f7837462c80cff90527ee35ef185571e1c327beac                             142.4s
 => => resolve docker.io/library/nginx:latest@sha256:4c0fdaa8b6341bfdeca5f18f7837462c80cff90527ee35ef185571e1c327beac                               0.0s
 => => sha256:8b7dd3ed1dc34cc1caba38bbbf22bceda5bd2e7c4e1b3c11ca64cda2ed186a2c 41.37MB / 41.37MB                                                  140.1s
 => => sha256:161ef4b1bf7effb350a2a9625cb2b59f69d54ec6059a8a155a1438d0439c593c 1.99kB / 1.99kB                                                      0.0s
 => => sha256:a8758716bb6aa4d90071160d27028fe4eaee7ce8166221a97d30440c8eac2be6 7.02kB / 7.02kB                                                      0.0s
 => => sha256:35497dd96569b9139cd388fd7107df32ccdc1449b205536bce0968b2dec3e7dc 628B / 628B                                                          1.9s
 => => sha256:4c0fdaa8b6341bfdeca5f18f7837462c80cff90527ee35ef185571e1c327beac 7.59kB / 7.59kB                                                      0.0s
 => => sha256:2f44b7a888fa005d07c031d3cfad2a1c0344207def2ab9dbb97712425ff812c1 29.13MB / 29.13MB                                                  117.3s
 => => sha256:36664b6ce66b304efa7ba48eb960133a085c2ec800a9f8887df94a82679334c1 955B / 955B                                                          4.9s
 => => sha256:2d455521f76cee8b8b2e21457075cc500c60373d70acb217f12838818fc3da90 366B / 366B                                                          6.9s
 => => sha256:dc9c4fdb83d69ef5986ec344c6b75606b3a417c7434268cb6995962be5312f14 1.21kB / 1.21kB                                                      8.7s
 => => sha256:8056d2bcf3b682573ee5b0c176c1209df285d5be0df98ec6ae08bf7421179b74 1.40kB / 1.40kB                                                     11.1s
 => => extracting sha256:2f44b7a888fa005d07c031d3cfad2a1c0344207def2ab9dbb97712425ff812c1                                                           2.5s
 => => extracting sha256:8b7dd3ed1dc34cc1caba38bbbf22bceda5bd2e7c4e1b3c11ca64cda2ed186a2c                                                           1.9s
 => => extracting sha256:35497dd96569b9139cd388fd7107df32ccdc1449b205536bce0968b2dec3e7dc                                                           0.0s
 => => extracting sha256:36664b6ce66b304efa7ba48eb960133a085c2ec800a9f8887df94a82679334c1                                                           0.0s
                                                                                                                                                    0.0s
 => => writing image sha256:59dd29503f5fda64279121bf4aad42037acdc8c7fbd3ee000f54fdc711620a79                                                        0.0s
 => => naming to docker.io/library/simple-nginx                                                                                                     0.0s
```

As usual, we can verify the good installtion of our image in Docker:

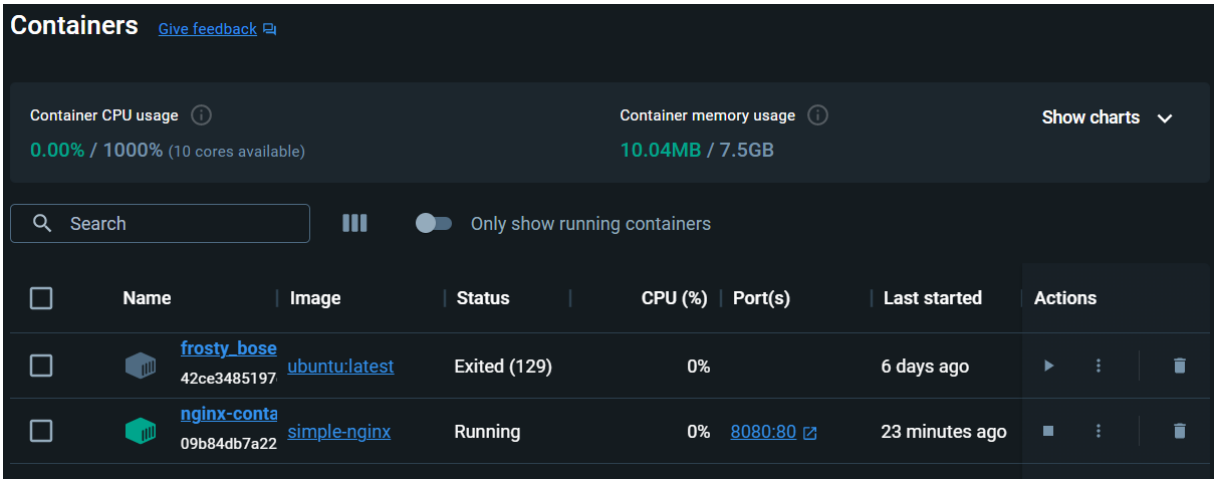


And here, we run our container we just created on the chosen ports.

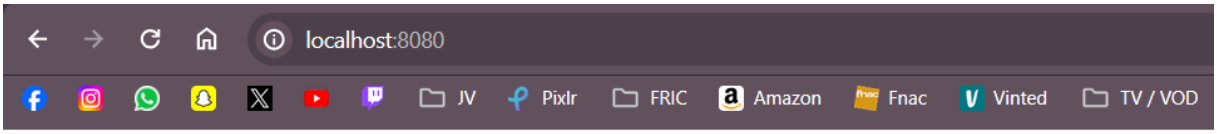
The command: '**docker run -d -p 8080:80 --name nginx-container simple-nginx**'.

```
PS C:\Users\Loeva\OneDrive\Bureau\ESILV\A4 cycle ingé DIA\Semestre 8\~ programmation\Containerization Technologies\TD 2>
 docker run -d -p 8080:80 --name nginx-container simple-nginx
09b84db7a22559f5341b1ed49743d66c825925b4263681e15e8ea006d48ed2ba
```

Our container, currently running, in Docker:



Now, on http://localhost:8080/, we can see that our web server is correctly running:



Hello from the Web Server!

Containerization Technologies

Bonus access the webpage using the terminal

Using this command: '**curl** http://localhost:8080', we can access the web page using the terminal. It shows us some basic informations and the content:

```
PS C:\Users\Loeva\OneDrive\Bureau\ESILV\A4 cycle ingé DIA\Semestre 8\~ programmation\Containerization Technologies\TD 2>
 curl http://localhost:8080


StatusCode        : 200
StatusDescription : OK
Content           : <!DOCTYPE html>

                    <html>

                    <head>
                        <title>Simple Web Server</title>
                    </head>

                    <body>
                        <h1>Hello from the Web Server!</h1>
                    </body>

                    <footer>
                        <p>Containerization Technologies</p>
                    </...
RawContent        : HTTP/1.1 200 OK
                    Connection: keep-alive
                    Accept-Ranges: bytes
                    Content-Length: 218
                    Content-Type: text/html
                    Date: Mon, 15 Jan 2024 17:14:31 GMT
                    ETag: "65a55d4d-da"
                    Last-Modified: Mon, 15 Jan 2024 1...
Forms             : {}
Headers           : {[Connection, keep-alive], [Accept-Ranges, bytes], [Content-Length, 218], [Content-Type, text/html]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 218
```

Bonus display the logs

Using this command: '**docker logs nginx-container**', we can see the logs of the container, with its name. It shows us all the logs of the container, since its debut:

```
PS C:\Users\Loeva\OneDrive\Bureau\ESILV\A4 cycle ingé DIA\Semestre 8\~ programmation\Containerization Technologies\TD 2> docker logs nginx-container
/docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
/docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
/docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
10-listen-on-ipv6-by-default.sh: info: Getting the checksum of /etc/nginx/conf.d/default.conf
10-listen-on-ipv6-by-default.sh: info: Enabled listen on IPv6 in /etc/nginx/conf.d/default.conf
/docker-entrypoint.sh: Sourcing /docker-entrypoint.d/15-local-resolvers.envsh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
/docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
/docker-entrypoint.sh: Configuration complete; ready for start up
2024/01/15 16:39:06 [notice] 1#1: using the "epoll" event method
2024/01/15 16:39:06 [notice] 1#1: nginx/1.25.3
2024/01/15 16:39:06 [notice] 1#1: built by gcc 12.2.0 (Debian 12.2.0-14)
2024/01/15 16:39:06 [notice] 1#1: OS: Linux 5.15.133.1-microsoft-standard-WSL2
2024/01/15 16:39:06 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/01/15 16:39:06 [notice] 1#1: start worker processes
2024/01/15 16:39:06 [notice] 1#1: start worker process 29
2024/01/15 16:39:06 [notice] 1#1: start worker process 30
2024/01/15 16:39:06 [notice] 1#1: start worker process 31
2024/01/15 16:39:06 [notice] 1#1: start worker process 32
2024/01/15 16:39:06 [notice] 1#1: start worker process 33
2024/01/15 16:39:06 [notice] 1#1: start worker process 34
2024/01/15 16:39:06 [notice] 1#1: start worker process 35
2024/01/15 16:39:06 [notice] 1#1: start worker process 36
2024/01/15 16:39:06 [notice] 1#1: start worker process 37
2024/01/15 16:39:06 [notice] 1#1: start worker process 38
2024/01/15 16:39:06 [notice] 1#1: start worker process 39
2024/01/15 16:39:06 [notice] 1#1: start worker process 40
172.17.0.1 - - [15/Jan/2024:16:39:36 +0000] "GET / HTTP/1.1" 200 218 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
2024/01/15 16:39:36 [error] 30#30: *1 open() "/usr/share/nginx/html/favicon.ico" failed (2: No such file or directory), client: 172.17.0.1, server: localhost, request: "GET /favicon.ico HTTP/1.1", host: "localhost:8080", referrer: "http://localhost:8080/"
172.17.0.1 - - [15/Jan/2024:16:39:36 +0000] "GET /favicon.ico HTTP/1.1" 404 555 "http://localhost:8080/" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.17.0.1 - - [15/Jan/2024:16:39:46 +0000] "GET / HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/120.0.0.0 Safari/537.36" "-"
172.17.0.1 - - [15/Jan/2024:17:14:31 +0000] "GET / HTTP/1.1" 200 218 "-" "Mozilla/5.0 (Windows NT; Windows NT 10.0; fr-FR) WindowsPowerShell/5.1.22621.2506" "-"
```

Bonus add resource limitation

To add some limits when running our container, we could modify the command we use to run the container, using '--cpus' and '--memory'.

The command: '**docker run -d -p 8080:80 --name nginx-container --cpus 0.5 --memory 256m simple-nginx**' (change limits as you want).

Bonus using apache

As we did with nginx, we create the Dockerfile (same structure than before):

```
Containerization Technologies > TD 2 > Dockerfile.apache > ...
1   # use the apache image
2   FROM httpd:latest
3
4   # copy the HTML file created before to the apache default HTML directory
5   COPY index.html /usr/local/apache2/htdocs/
6
7   # expose port 80
8   EXPOSE 80
9
10  # start apache server
11  CMD ["httpd", "-D", "FOREGROUND"]
```

And we use the same commands than before.

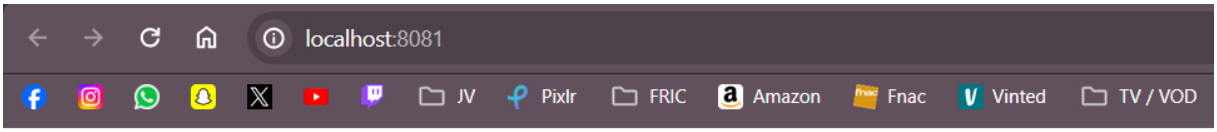
To build: '**docker build -t simple-apache -f Dockerfile.apache .**'.

```
Pdocker build -t simple-apache -f Dockerfile.apache . DIA\Semestre 8\~ programmation\Containerization Technologies\TD 2>
[+] Building 7.0s (8/8) FINISHED                                                                                docker:default
 => [internal] load build definition from Dockerfile.apache                                                              0.0s
 => => transferring dockerfile: 298B                                                                                     0.0s
 => [internal] load .dockerignore                                                                                        0.0s
 => => transferring context: 2B                                                                                          0.0s
 => [internal] load metadata for docker.io/library/httpd:latest                                                          1.6s
 => [auth] library/httpd:pull token for registry-1.docker.io                                                             0.0s
 => [internal] load build context                                                                                        0.0s
 => => transferring context: 251B                                                                                        0.0s
 => [1/2] FROM docker.io/library/httpd:latest@sha256:7765977cf2063fec486b63ddea574faf8fbed285f2b17020fa7ef70a4926cdec    5.2s
 => => resolve docker.io/library/httpd:latest@sha256:7765977cf2063fec486b63ddea574faf8fbed285f2b17020fa7ef70a4926cdec     0.0s
 => => sha256:7765977cf2063fec486b63ddea574faf8fbed285f2b17020fa7ef70a4926cdec 7.48kB / 7.48kB                           0.0s
 => => sha256:19f21e3171a03681ea52f3ac950549dd356586d297d18e2d0cafee7979531b64 1.79kB / 1.79kB                           0.0s
 => => sha256:92fa43a2ff60503fdf250ef0a16d4170aa2a47c4b5a60d427724283543a8792a 8.17kB / 8.17kB                           0.0s
 => => sha256:5abb3599da3424ec59f721c419363abc4a75cd4413b0b7ab568213c7f6406b6a 145B / 145B                               0.1s
 => => sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38e8acc1 32B / 32B                                 0.2s
 => => sha256:fa608a886227e72cca781dc4df2bddc16d53c3318e4c2a39bfe29e0e734859b3 4.20MB / 4.20MB                           0.8s
 => => extracting sha256:5abb3599da3424ec59f721c419363abc4a75cd4413b0b7ab568213c7f6406b6a                                 0.0s
 => => sha256:afe6bbf0043757b93f994f0dc1484fa4967341f3bc9aa237e043f4ff50aa76c7 31.19MB / 31.19MB                         3.1s
 => => extracting sha256:4f4fb700ef54461cfa02571ae0db9a0dc1e0cdb5577484a6d75e68dc38e8acc1                                 0.0s
 => => sha256:fd0ef2a496774dc7f72353423b04009eacb760bf81e1e65c9a05ac430a2ed9bf 293B / 293B                               0.4s
 => => extracting sha256:fa608a886227e72cca781dc4df2bddc16d53c3318e4c2a39bfe29e0e734859b3                                 0.4s
 => => extracting sha256:afe6bbf0043757b93f994f0dc1484fa4967341f3bc9aa237e043f4ff50aa76c7                                 1.9s
 => => extracting sha256:fd0ef2a496774dc7f72353423b04009eacb760bf81e1e65c9a05ac430a2ed9bf                                 0.0s
 => [2/2] COPY index.html /usr/local/apache2/htdocs/                                                                     0.1s
 => exporting to image                                                                                                   0.0s
 => => exporting layers                                                                                                  0.0s
 => => writing image sha256:891f06e460ba3167bdee45708908602d31ecfc17b48d83f887e25bf067271cc                             0.0s
 => => naming to docker.io/library/simple-apache                                                                         0.0s
```

To run: '**docker run -d -p 8081:80 --name apache-container simple-apache**'.

```
PS C:\Users\Loeva\OneDrive\Bureau\ESILV\A4 cycle ingé DIA\Semestre 8\~ programmation\Containerization Technologies\TD 2>
 docker run -d -p 8081:80 --name apache-container simple-apache
2fa64993a6a26d0fa0d7673f52657fcd11e801a8d3edad2b84a7da6729c6fba1
```

We can verify that our web page is running on the good address:



Hello from the Web Server!

Containerization Technologies