- **Name**: Iskander Nafikov
- **E-mail**: i.nafikov@innopolis.university
- **Username**: `i.nafikov`
- **Hostname**: `macbook-KN70WX2HPH`

Task 1: Get familiar with Docker Engine

> 1. Pull Nginx v1.23.3 image from dockerhub registry and confirm it is listed in local images

- I pulled `nginx:1.23.3` image

```
docker pull nginx:1.23.3
```

```
i.nafikov@macbook-KN70WX2HPH ~ % docker pull nginx:1.23.3
1.23.3: Pulling from library/nginx
fcdb9667c46b: Pull complete
d9f3ddb89cd9: Pull complete
7bae30547136: Pull complete
3cf7afb4b643: Pull complete
e20f1c808987: Pull complete
e40c7e401531: Pull complete
Digest: sha256:f4e3b6489888647ce1834b601c6c06b9f8c03dee6e097e13ed3e28c01ea3ac8c
Status: Downloaded newer image for nginx:1.23.3
docker.io/library/nginx:1.23.3
i.nafikov@macbook-KN70WX2HPH ~ %
```

- And confirmed it is listed in my local images

```
docker image ls | grep nginx
```

```
i.nafikov@macbook-KN70WX2HPH ~ % docker image ls | grep nginx
nginx                                                 1.23.3          25d0f5e5d6da   22 months ago   135MB
```

> 2. Run the pulled Nginx as a container with the below properties
>   a. Map the port to 8080.
>   b. Name the container as `nginx-<stX>`.
>   c. Run it as daemon .

- I launched Nginx container

```
docker run -d -p 8080:80 --name nginx-st15 nginx:1.23.3
```

```
i.nafikov@macbook-KN70WX2HPH ~ % docker run -d -p 8080:80 --name nginx-st15 nginx:1.23.3
98efdb9da1b0291eee417dd81ad0a7c7ba8f1abf77a46410fc87d28683dcd9f8
```

- Explanation about the options used:
  - `-d` : Runs the container in detached mode (in the background).
  - `-p 8080:80` : Maps port 8080 on the host to port 80 in the container (Nginx listens on port 80 by default).
  - `--name nginx-st15` : Assigns a name to the container (in my case the suffix is `st15`).

3. Confirm port mapping.
   a. List open ports in host machine.
   b. List open ports inside the running container.
   c. Access the page from your browser.

- I listed open ports on my host machine using `lsof` and confirmed that 8080 is open on TCP

```
sudo lsof -nP -iTCP:8080 -sTCP:LISTEN
```

```
i.nafikov@macbook-KN70WX2HPH ~ % sudo lsof -nP -iTCP:8080 -sTCP:LISTEN
COMMAND   PID      USER   FD   TYPE             DEVICE SIZE/OFF NODE NAME
ssh     49840 i.nafikov   12u  IPv4 0x65dfe194bb6bd91      0t0  TCP *:8080 (LISTEN)
i.nafikov@macbook-KN70WX2HPH ~ %
```

- I listed open ports inside the running container
  1. First I installed `lsof` to the container

```
docker exec nginx-st15 apt install -y lsof
```

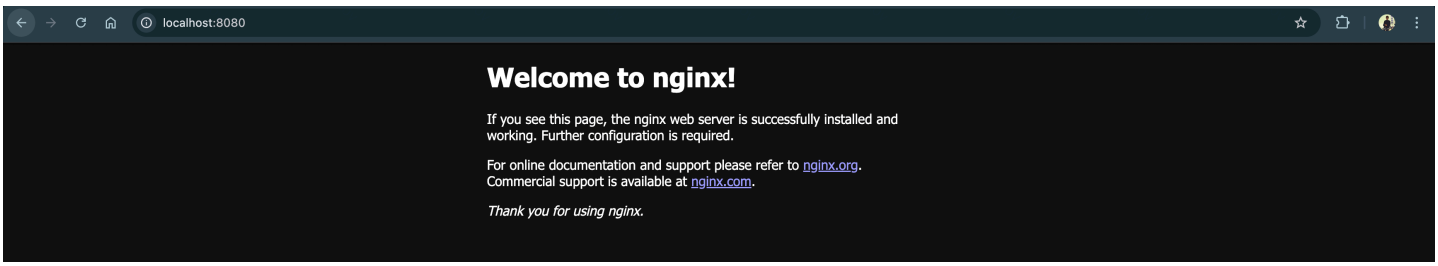
```
i.nafikov@macbook-KN70WX2HPH ~ % docker exec nginx-st15 apt install -y lsof

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists...
Building dependency tree...
Reading state information...
Suggested packages:
  perl
The following NEW packages will be installed:
  lsof
0 upgraded, 1 newly installed, 0 to remove and 48 not upgraded.
Need to get 313 kB of archives.
After this operation, 478 kB of additional disk space will be used.
Get:1 http://deb.debian.org/debian bullseye/main arm64 lsof arm64 4.93.2+dfsg-1.1 [313 kB]
debconf: delaying package configuration, since apt-utils is not installed
Fetched 313 kB in 0s (753 kB/s)
Selecting previously unselected package lsof.
(Reading database ... 7815 files and directories currently installed.)
Preparing to unpack .../lsof_4.93.2+dfsg-1.1_arm64.deb ...
Unpacking lsof (4.93.2+dfsg-1.1) ...
Setting up lsof (4.93.2+dfsg-1.1) ...
```

  2. Then I listed open ports running `lsof` inside the container and confirmed that it 80 is open

```
i.nafikov@macbook-KN70WX2HPH ~ % docker exec nginx-st15 lsof -nP -iTCP:80 -sTCP:LISTEN
COMMAND PID USER   FD   TYPE DEVICE SIZE/OFF NODE NAME
nginx     1 root    6u  IPv4 430293      0t0  TCP *:80 (LISTEN)
nginx     1 root    7u  IPv6 430294      0t0  TCP *:80 (LISTEN)
```

- I accessed the page from my browser and it worked

```
i.nafikov@macbook-KN70WX2HPH ~ % curl localhost:8080
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
html { color-scheme: light dark; }
body { width: 35em; margin: 0 auto;
font-family: Tahoma, Verdana, Arial, sans-serif; }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```



4. Create a Dockerfile similar to the below properties (let's call it container A).
   a. Image tag should be Nginx v1.23.3.
   b. Create a custom index.html file and copy it to your docker image to replace the Nginx default web page.
   c. Build the image from the Dockerfile, tag it during build as `nginx:<stX>`, check/validate local images, and run your custom made docker image.
   d. Access via browser and validate that your custom page is hosted.

- First I created my custom `index.html` page with the help of `vim`

```
i.nafikov@macbook-KN70WX2HPH lab-01 % vim index.html
```

```
1 <html>
2
3 <h1>Hello, World!</h1>
4
5 <p>Iskander Nafikov</p>
6
7 <html>
~
```

- Then I created `Dockerfile`. I derived my image from `nginx:1.23.3` and pointed to copy my `index.html` to his image

```
i.nafikov@macbook-KN70WX2HPH lab-01 % vim Dockerfile
```

```
1 FROM nginx:1.23.3
2
3 COPY index.html /usr/share/nginx/html/index.html
4
5 ENTRYPOINT ["nginx", "-g", "daemon off;"]
6
~
```

```
FROM nginx:1.23.3

COPY index.html /usr/share/nginx/html/index.html

ENTRYPOINT ["nginx", "-g", "daemon off;"]
```

- I built the image from the `Dockerfile` and tagged it as `nginx:st15`

```
i.nafikov@macbook-KN70WX2HPH lab-01 % docker build --tag=nginx:st15 .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.
            Install the buildx component to build images with BuildKit:
            https://docs.docker.com/go/buildx/

Sending build context to Docker daemon  7.168kB
Step 1/3 : FROM nginx:1.23.3
 ---> 25d0f5e5d6da
Step 2/3 : COPY index.html /usr/share/nginx/html/index.html
 ---> 4025db4078dc
Step 3/3 : ENTRYPOINT ["nginx", "-g", "daemon off;"]
 ---> Running in 5fd8d8377330
 ---> Removed intermediate container 5fd8d8377330
 ---> 38b36c4e54c4
Successfully built 38b36c4e54c4
Successfully tagged nginx:st15
```

- I checked my local images and saw that there were 2 containers with the name `nginx`. However tags (after colon) were different.

```
i.nafikov@macbook-KN70WX2HPH lab-01 % docker image ls | grep nginx
nginx                                                             st15              38b36c4e54c4   About a minute ago   135MB
nginx                                                             1.23.3            25d0f5e5d6da   22 months ago        135MB
```

- I run container from this image

```
i.nafikov@macbook-KN70WX2HPH lab-01 %  docker run -d -p 8081:80 --name nginx-st15-custom nginx:st15
6382eefb62e0571b691d441cb8e8467a0bba4bef59245c8d9a1c2ad5f0be4cd5
```

- Finally I could access my custom page

```
i.nafikov@macbook-KN70WX2HPH lab-01 % curl localhost:8081
<html>

<h1>Hello, World!</h1>

<p>Iskander Nafikov</p>

<html>
```

Task 2: Work with multi-container environment

> 1. Create another Dockerfile similar to step 1.4 (Let's call it container B), and an index.html with different content.

- First I placed `Dockerfile` and `index.html` files inside the new directory `container-A/` on my host machine

```
i.nafikov@macbook-KN70WX2HPH lab-01 % ls container-A
Dockerfile      index.html
i.nafikov@macbook-KN70WX2HPH lab-01 % vim container-A/Dockerfile
```

- Then I created `container-B/` directory and copied there `Dockerfile` and `index.html` files from the `container-B`

```
i.nafikov@macbook-KN70WX2HPH lab-01 % cp container-A/Dockerfile container-A/index.html container-B/
```

```
i.nafikov@macbook-KN70WX2HPH lab-01 % ls container-B
Dockerfile      index.html
```

- I changed the `container-B/index.html` as the following:

```
i.nafikov@macbook-KN70WX2HPH lab-01 % vim container-B/index.html
```

```
1 <html>
2
3 <h1>Hello from the container B</h1>
4
5 <p>Iskander Nafikov</p>
6
7 <html>
```

2. Write a docker-compose file with the below properties

- Afterwards, I created `docker-compose.yaml` inside the direcotyr `lab-01/` which is the parent directory for the `container-A` / and `container-B/` directories

```
i.nafikov@macbook-KN70WX2HPH lab-01 % ls
container-A                     docker-compose.yaml             iu-devsecops-lab-01-Iskander_Nafikov.md
container-B                     images
i.nafikov@macbook-KN70WX2HPH lab-01 % vim docker-compose.yaml
```

```
services:
  a:
    build:
      context: ./container-A
      dockerfile: Dockerfile
    ports:
      - "8080:80"
    volumes:
      - ./container-A/index.html:/usr/share/nginx/html/index.html

  b:
    build:
      context: ./container-B
      dockerfile: Dockerfile
    ports:
      - "9090:80"
    volumes:
      - ./container-B/index.html:/usr/share/nginx/html/index.html
```


```
services:
  a:
    build:
      context: ./container-A
      dockerfile: Dockerfile
    ports:
      - "8080:80"
    volumes:
      - ./container-A/index.html:/usr/share/nginx/html/index.html

  b:
    build:
      context: ./container-B
      dockerfile: Dockerfile
    ports:
      - "9090:80"
    volumes:
      - ./container-B/index.html:/usr/share/nginx/html/index.html
```

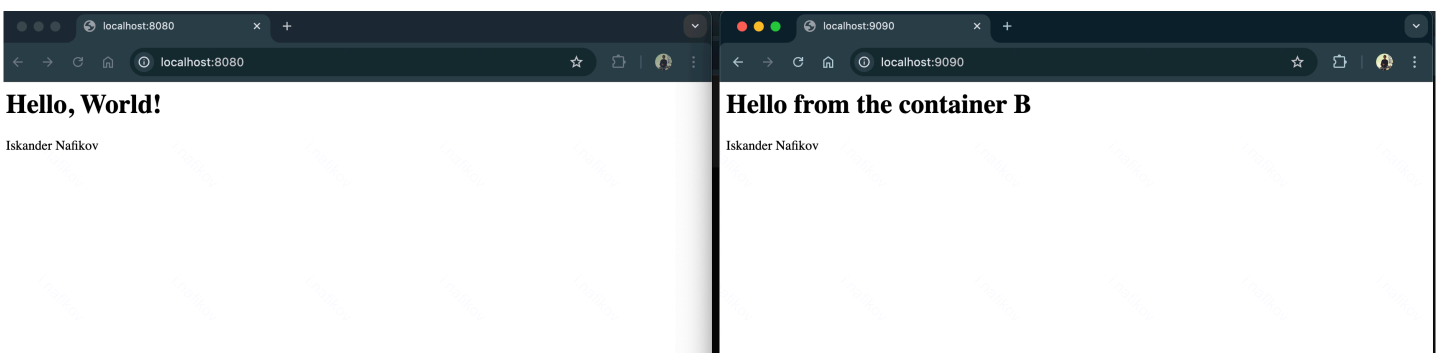

- I launched my containers using `docker-compose`

```
docker-compose up
```

```
i.nafikov@macbook-KN70WX2HPH lab-01 % docker-compose up
[+] Building 0.1s (8/8) FINISHED                                                                                docker:default
 => [b internal] load build definition from Dockerfile                                                                    0.0s
 => => transferring dockerfile: 99B                                                                                       0.0s
 => [a internal] load build definition from Dockerfile                                                                    0.0s
 => => transferring dockerfile: 99B                                                                                       0.0s
 => [a internal] load metadata for docker.io/library/nginx:1.23.3                                                         0.0s
 => [b internal] load .dockerignore                                                                                       0.0s
 => => transferring context: 2B                                                                                           0.0s
 => [a internal] load .dockerignore                                                                                       0.0s
 => => transferring context: 2B                                                                                           0.0s
 => [b 1/1] FROM docker.io/library/nginx:1.23.3                                                                           0.0s
 => [a] exporting to image                                                                                                0.0s
 => => exporting layers                                                                                                   0.0s
 => => writing image sha256:44432efe49a57e940e9c80bc86091c81aeb675ceb86ed1fd536582820c9e7483                              0.0s
 => => naming to docker.io/library/lab-01-a                                                                               0.0s
 => [b] exporting to image                                                                                                0.0s
 => => exporting layers                                                                                                   0.0s
 => => writing image sha256:218f3aa836d8fe8247b560c3b96dc45940ba3a5de4c398c4ba7f9547620debb2                              0.0s
 => => naming to docker.io/library/lab-01-b                                                                               0.0s
[+] Running 3/3
 ✔ Network lab-01_default  Created                                                                                        0.0s
 ✔ Container lab-01-a-1    Created                                                                                        0.0s
 ✔ Container lab-01-b-1    Created                                                                                        0.0s
Attaching to a-1, b-1
a-1  | 2025/01/26 16:13:53 [notice] 1#1: using the "epoll" event method
a-1  | 2025/01/26 16:13:53 [notice] 1#1: nginx/1.23.3
a-1  | 2025/01/26 16:13:53 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
a-1  | 2025/01/26 16:13:53 [notice] 1#1: OS: Linux 6.8.0-31-generic
a-1  | 2025/01/26 16:13:53 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
b-1  | 2025/01/26 16:13:53 [notice] 1#1: using the "epoll" event method
b-1  | 2025/01/26 16:13:53 [notice] 1#1: nginx/1.23.3
a-1  | 2025/01/26 16:13:53 [notice] 1#1: start worker processes
b-1  | 2025/01/26 16:13:53 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
a-1  | 2025/01/26 16:13:53 [notice] 1#1: start worker process 6
b-1  | 2025/01/26 16:13:53 [notice] 1#1: OS: Linux 6.8.0-31-generic
b-1  | 2025/01/26 16:13:53 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
b-1  | 2025/01/26 16:13:53 [notice] 1#1: start worker processes
b-1  | 2025/01/26 16:13:53 [notice] 1#1: start worker process 7
b-1  | 2025/01/26 16:13:53 [notice] 1#1: start worker process 8
b-1  | 2025/01/26 16:13:53 [notice] 1#1: start worker process 9
b-1  | 2025/01/26 16:13:53 [notice] 1#1: start worker process 10
a-1  | 2025/01/26 16:13:53 [notice] 1#1: start worker process 7
a-1  | 2025/01/26 16:13:53 [notice] 1#1: start worker process 8
a-1  | 2025/01/26 16:13:53 [notice] 1#1: start worker process 9
load: 2.91  cmd: docker-compose 62708 running 0.07u 0.06s
```

- Finally, I could access my pages

```
i.nafikov@macbook-KN70WX2HPH lab-01 % curl localhost:8080
<html>

<h1>Hello, World!</h1>

<p>Iskander Nafikov</p>

<html>
i.nafikov@macbook-KN70WX2HPH lab-01 % curl localhost:9090
<html>

<h1>Hello from the container B</h1>

<p>Iskander Nafikov</p>

<html>
```



- Now if I update `index.html` for the container `A`

```
i.nafikov@macbook-KN70WX2HPH lab-01 % echo "<h4>This is container A<h4>" >> container-A/index.html
```

- We can notice changes immediately in my browser

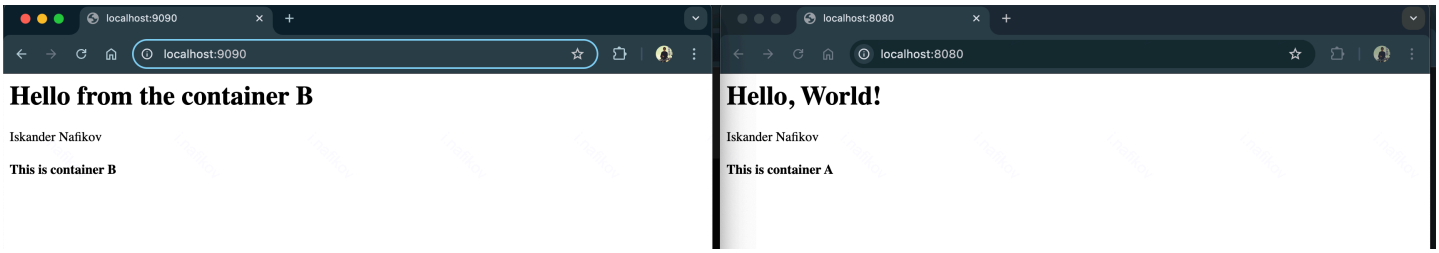
- The same works for the container `B`

```
i.nafikov@macbook-KN70WX2HPH lab-01 % echo "<h4>This is container B<h4>" >> container-B/index.html
```

- This became possible due to mount binding
- Let's redeploy our services and check it changes were saved

```
^CGracefully stopping... (press Ctrl+C again to force)
[+] Stopping 2/2
 ✔ Container lab-01-a-1  Stopped                                                    0.2s
 ✔ Container lab-01-b-1  Stopped                                                    0.1s
canceled
i.nafikov@macbook-KN70WX2HPH lab-01 % docker-compose up
[+] Running 2/0
 ✔ Container lab-01-b-1  Created                                                    0.0s
 ✔ Container lab-01-a-1  Created                                                    0.0s
Attaching to a-1, b-1
a-1  | 2025/01/26 16:29:19 [notice] 1#1: using the "epoll" event method
a-1  | 2025/01/26 16:29:19 [notice] 1#1: nginx/1.23.3
a-1  | 2025/01/26 16:29:19 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
a-1  | 2025/01/26 16:29:19 [notice] 1#1: OS: Linux 6.8.0-31-generic
a-1  | 2025/01/26 16:29:19 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
a-1  | 2025/01/26 16:29:19 [notice] 1#1: start worker processes
a-1  | 2025/01/26 16:29:19 [notice] 1#1: start worker process 7
a-1  | 2025/01/26 16:29:19 [notice] 1#1: start worker process 8
a-1  | 2025/01/26 16:29:19 [notice] 1#1: start worker process 9
a-1  | 2025/01/26 16:29:19 [notice] 1#1: start worker process 10
b-1  | 2025/01/26 16:29:19 [notice] 1#1: using the "epoll" event method
b-1  | 2025/01/26 16:29:19 [notice] 1#1: nginx/1.23.3
b-1  | 2025/01/26 16:29:19 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
b-1  | 2025/01/26 16:29:19 [notice] 1#1: OS: Linux 6.8.0-31-generic
b-1  | 2025/01/26 16:29:19 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
b-1  | 2025/01/26 16:29:19 [notice] 1#1: start worker processes
b-1  | 2025/01/26 16:29:19 [notice] 1#1: start worker process 6
b-1  | 2025/01/26 16:29:19 [notice] 1#1: start worker process 7
b-1  | 2025/01/26 16:29:19 [notice] 1#1: start worker process 8
b-1  | 2025/01/26 16:29:19 [notice] 1#1: start worker process 9
```

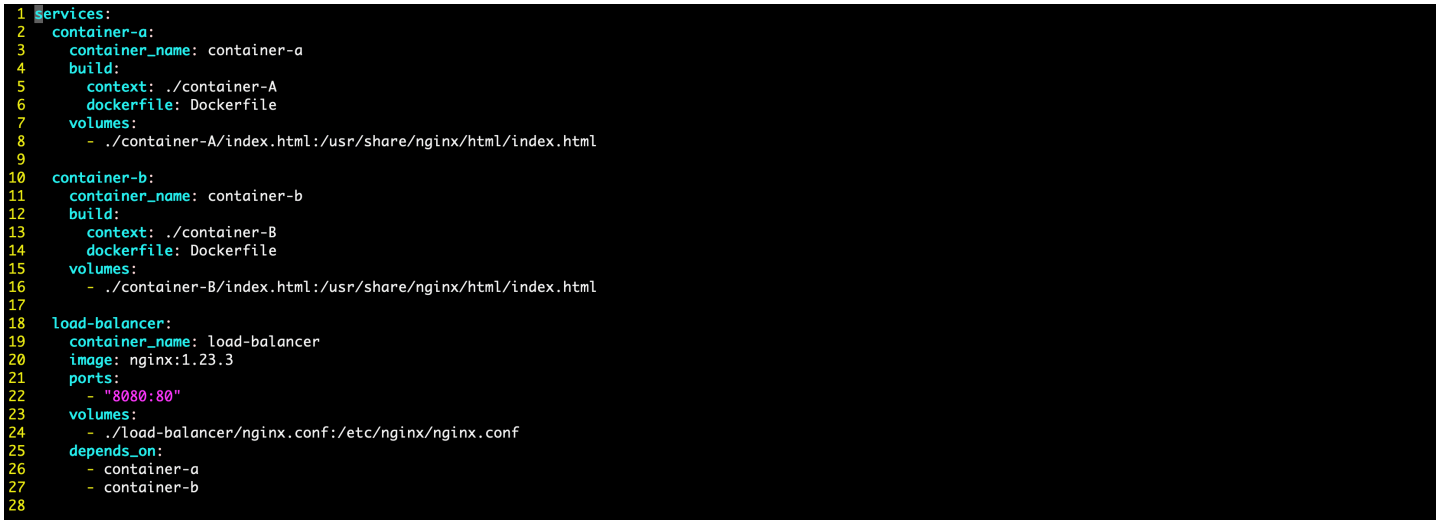
- We can see that they were



Task 3: Configure L7 Loadbalaner

> 1. Install Nginx in the host machine or add a third container in the docker-compose that will act as loadbalancer, and configure it in front of two containers in a manner that it should distribute the load in a Weighted Round Robin approach.

- Firstly I updated `docker-compose.yaml` file as following



```yaml
services:
  container-a:
    container_name: container-a
    build:
      context: ./container-A
      dockerfile: Dockerfile
    volumes:
      - ./container-A/index.html:/usr/share/nginx/html/index.html

  container-b:
    container_name: container-b
    build:
      context: ./container-B
      dockerfile: Dockerfile
    volumes:
      - ./container-B/index.html:/usr/share/nginx/html/index.html

  load-balancer:
    container_name: load-balancer
    image: nginx:1.23.3
    ports:
      - "8080:80"
    volumes:
      - ./load-balancer/nginx.conf:/etc/nginx/nginx.conf
    depends_on:
      - container-a
      - container-b
```

- Here is the list of what was changed in the `docker-compose.yaml`:
  - I renamed my services to `container-a` and `container-b` respectively
  - I added `container_name` option for my services to make their name explicitly and statically set
  - I removed port binding for `container-a` and `contaier-b` because now I do not need to access them separately from my host, while load balancer can access them through docker

network created by docker compose

- I added `load-balancer` service that:
  - is derived from the base `nginx:1.23.3` image
  - exposes `8080` port to my host machine
  - has bound mount to `nginx.conf` file that stores setting for my Nginx load-balancing server
  - has `depends_on` option that is practically useless but is necessary to see inter-service dependings explicitly

- Let's check `nginx.conf` file

```
i.nafikov@macbook-KN70WX2HPH load-balancer % ls
nginx.conf
i.nafikov@macbook-KN70WX2HPH load-balancer % vim nginx.conf
```

```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    upstream backend {
        server container-a:80 weight=5;
        server container-b:80 weight=5;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://backend;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
    }
}
```


```
worker_processes 1;

events {
    worker_connections 1024;
}

http {
    upstream backend {
        server container-a:80 weight=5;
        server container-b:80 weight=5;
    }

    server {
        listen 80;

        location / {
            proxy_pass http://backend;
            proxy_set_header Host $host;
            proxy_set_header X-Real-IP $remote_addr;
            proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
            proxy_set_header X-Forwarded-Proto $scheme;
        }
```

```
    }
}
```

- We see there the following options that are interesting for us:
  - `http.upstream = backend` is our upstream that consists of two servers: `container-a:80` and `container-b:80`. The traffic distributes between these servers equally because they both have the same weight $= 5$ what means 50% of the load. Therefore this algorithm of load balancing is actually a Weighted Round Robin (because we I use weights) that becomes a simple Round Robin because of equal weights.
  - `http.server.listen = 80` means our load balancing server works on 80 port
  - `http.server.location = /` shows that our server is accessible by root path and it is actually a proxy that passes all the traffic to `backend` upstream with all the necessary headers.
- Now let's run my services using docker compose

```
i.nafikov@macbook-KN70WX2HPH load-balancer % docker-compose up
[+] Running 3/0
 ✔ Container container-b    Created                                                                                0.0s
 ✔ Container container-a    Created                                                                                0.0s
 ✔ Container load-balancer  Created                                                                                0.0s
Attaching to container-a, container-b, load-balancer
container-b    | 2025/01/26 17:35:07 [notice] 1#1: using the "epoll" event method
container-b    | 2025/01/26 17:35:07 [notice] 1#1: nginx/1.23.3
container-b    | 2025/01/26 17:35:07 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
container-b    | 2025/01/26 17:35:07 [notice] 1#1: OS: Linux 6.8.0-31-generic
container-b    | 2025/01/26 17:35:07 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
container-b    | 2025/01/26 17:35:07 [notice] 1#1: start worker processes
container-b    | 2025/01/26 17:35:07 [notice] 1#1: start worker process 7
container-b    | 2025/01/26 17:35:07 [notice] 1#1: start worker process 8
container-b    | 2025/01/26 17:35:07 [notice] 1#1: start worker process 9
container-b    | 2025/01/26 17:35:07 [notice] 1#1: start worker process 10
container-a    | 2025/01/26 17:35:07 [notice] 1#1: using the "epoll" event method
container-a    | 2025/01/26 17:35:07 [notice] 1#1: nginx/1.23.3
container-a    | 2025/01/26 17:35:07 [notice] 1#1: built by gcc 10.2.1 20210110 (Debian 10.2.1-6)
container-a    | 2025/01/26 17:35:07 [notice] 1#1: OS: Linux 6.8.0-31-generic
container-a    | 2025/01/26 17:35:07 [notice] 1#1: getrlimit(RLIMIT_NOFILE): 1048576:1048576
container-a    | 2025/01/26 17:35:07 [notice] 1#1: start worker processes
container-a    | 2025/01/26 17:35:07 [notice] 1#1: start worker process 7
container-a    | 2025/01/26 17:35:07 [notice] 1#1: start worker process 8
container-a    | 2025/01/26 17:35:07 [notice] 1#1: start worker process 9
container-a    | 2025/01/26 17:35:07 [notice] 1#1: start worker process 10
load-balancer  | /docker-entrypoint.sh: /docker-entrypoint.d/ is not empty, will attempt to perform configuration
load-balancer  | /docker-entrypoint.sh: Looking for shell scripts in /docker-entrypoint.d/
load-balancer  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/10-listen-on-ipv6-by-default.sh
load-balancer  | 10-listen-on-ipv6-by-default.sh: info: IPv6 listen already enabled
load-balancer  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/20-envsubst-on-templates.sh
load-balancer  | /docker-entrypoint.sh: Launching /docker-entrypoint.d/30-tune-worker-processes.sh
load-balancer  | /docker-entrypoint.sh: Configuration complete; ready for start up
```

2. Access the page of Nginx ALB and validate, it is load-balancing the traffic (you see two different content per page reload).

- Let's try to access my backend by `localhost:8080` several times

```
i.nafikov@macbook-KN70WX2HPH lab-01 % curl localhost:8080
<html>

<h1>Hello, World!</h1>

<p>Iskander Nafikov</p>

<html>
<h4>This is container A<h4>
i.nafikov@macbook-KN70WX2HPH lab-01 % curl localhost:8080
<html>

<h1>Hello from the container B</h1>

<p>Iskander Nafikov</p>

<html>
<h4>This is container B<h4>
i.nafikov@macbook-KN70WX2HPH lab-01 % curl localhost:8080
<html>

<h1>Hello, World!</h1>

<p>Iskander Nafikov</p>

<html>
<h4>This is container A<h4>
i.nafikov@macbook-KN70WX2HPH lab-01 % curl localhost:8080
<html>

<h1>Hello from the container B</h1>

<p>Iskander Nafikov</p>

<html>
<h4>This is container B<h4>
i.nafikov@macbook-KN70WX2HPH lab-01 % curl localhost:8080
<html>

<h1>Hello, World!</h1>

<p>Iskander Nafikov</p>

<html>
<h4>This is container A<h4>
i.nafikov@macbook-KN70WX2HPH lab-01 % curl localhost:8080
<html>

<h1>Hello from the container B</h1>
```

- We see that our load balancing actually works!