Solution to Challenge-1

Steps to be reproduces to produce a solution for challenge-1:-

1. Check if the host machine have docker installed by executing the following command in the administrator terminal
   - docker --version

   If the terminal provides with a string showcasing a version of docker, the docker is installed in the host machine as

```
PS C:\Users\amrit> docker --version
Docker version 26.1.1, build 4cf5afa
```

   Otherwise Visit:-

   https://www.docker.com/products/docker-desktop/

   And download docker desktop from there and follow standard installation instructions.

2. Once confirmed the host machine have a docker stable version, run the docker desktop application
3. Now navigate to the directory of the challenge 1 folder.
4. Open this directory in the vs code.
5. Create a folder named '*public*'
6. In that folder proceed to create a index.html file.
7. In the index.html proceed to create a html static page by using ! emmet.
8. In the body write your name and student ID.
9. Create a Dockerfile with no extension in challenge 1 folder.
10. In that file proceed with the following code:-

    FROM nginx:latest // This creates an image of nginx distribution[1]

    COPY public /usr/share/nginx/html //This copies the public folder to the shared nginx folder to host it on a particular port

    EXPOSE 80 //This exposes the port 80 to the html requests for the localhost

    CMD ["nginx", "-g", "daemon off;"] // This command runs the default nginx container with the config of daemon off to run nginx in foreground
11. Now proceed to the terminal of VS code and build this image by running the following command[1]
    - docker build -t solution-01 .

    This command creates a docker image with the following command configured dockerfile

```
PS C:\OS\Solutions\docker-challenge-template> docker build -t solution-01 .
```

12. Now start this image with the following command:-
    - docker run -p 8080:80 -d –name docker-container solution-01
      This command runs a new container from the solution-01 image.

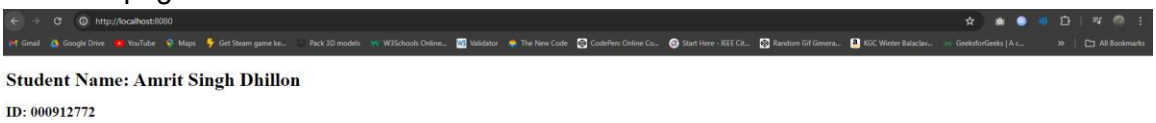
-p 8080:80: Maps port 8080 on the host machine to port 80 in the container.
-d: Runs the container in detached mode so the terminal does not gets bombarded with the logs.
--name docker -container: Gives the container the name docker -container

```
PS C:\OS\Solutions\docker-challenge-template> docker run -p 8080:80 -d --name docker-container solution-01
```

13. If the terminal produces a string of character then the docker container is running

```
PS C:\OS\Solutions\docker-challenge-template> docker run -p 8080:80 -d --name docker-container solution-01
53f00f92309163abc3617e972d871af13a464de182fbea1d76de86c677874a4a
```

14. Now if you open any browser in host machine and goto localhost:8080/ then you can see that html page rendered in the browser



And with this it concludes the ending of challenge 1.

Solution to Challenge-2

The following steps can be followed in order to proceed a solution to challenge 2

1. Proceed with the step 1 and 2 from the challenge 1 solution and this concludes the setting up of our docker environment.
2. As provided challenge2.zip, extract it into an appropriate directory.
3. Now proceed to open this folder in VS Code where we begin our journey.
4. Create a Dockerfile in this directory and write the following code :-

```
FROM node:latest
WORKDIR /app
COPY package.json ./
RUN npm i
COPY . .
```

```
EXPOSE 3000
CMD npm start
```

Explaining futher, this code works as follows, the first line FROM node:latest pulls the node image from the docker hub and then it instantiates this image into the docker once started. Now we proceed to separate our work directory to /app directory inside the image file structure. After that we copy the package.json into this app directory that provides us with all the dependencies required for this node server to execute properly and after that we install these packages by running npm I command that is also npm install and then we copy all the files particularly the sever.js into this /app folder. Now we expose port 3000 for this server and finally we proceed to start this server by executing npm start command.

5. Now once this dockerfile is successfully created, we go onto creating the docker compose file that is going to include the instructions for our api and nginx services.
6. Create new file named docker-compose.yml in the host folder with challenge2 files.
7. In this file write the following code:-

```
services:
    api:
      build: .
      ports:
          -“3000:3000”
      Environment:
          - NODE_ENV=production
    nginx:
      image: nginx:latest
      ports:
          - “8080:80”
      volumes:
          - ./nginx.conf:/etc/nginx/nginx.conf:ro
      depends_on:
          - api
```

This file includes the instructions for docker compose how it is going to operate once we initiate this docker compose

8. Once we have stated the instructions for the docker compose, we also need to instruct the nginx, how this container is going to map the services of the node server to the host machine for particular urls.
9. For that we create another file nginx.conf and write the following instructions:

```
events {
```

```
        worker_connections 1024;
    }

    http{
        upstream api{
            server api: 3000;
        }
        server{
            listen: 80;
            location /api {
                proxy_pass http://api;
                proxy_http_version 1.1;
                proxy_set_header Upgrade $http_upgrade;
                proxy_set_header Connection 'upgrade';
                proxy_set_header Host $host;
                proxy_cache_bypass $http_upgrade;
            }
        }
    }
```

This file includes the instructions of how a request from port 80 is going to be mapped to the server in simple sense this only instructs the structure of the http req sent to the server running on port 3000.

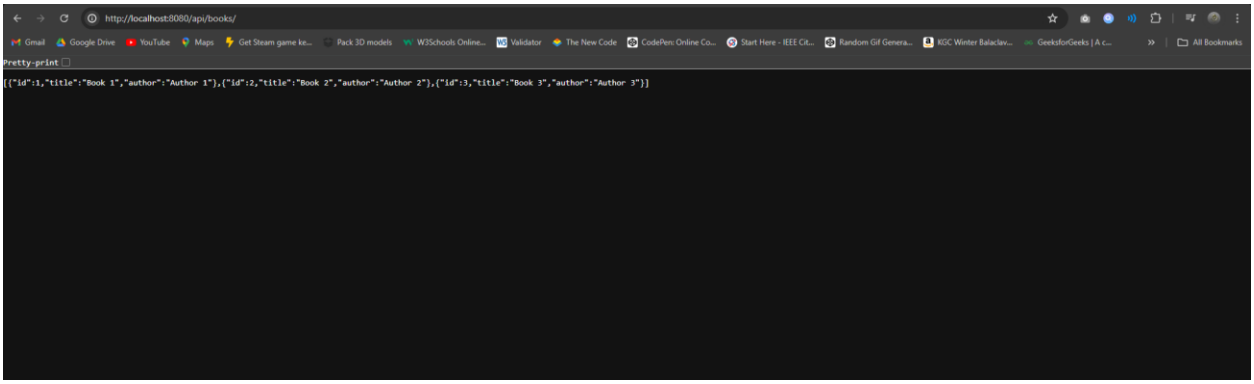
This nginx.conf file configures nginx to act as a reverse proxy, forwarding requests from http://localhost:8080/api to http://api:3000.

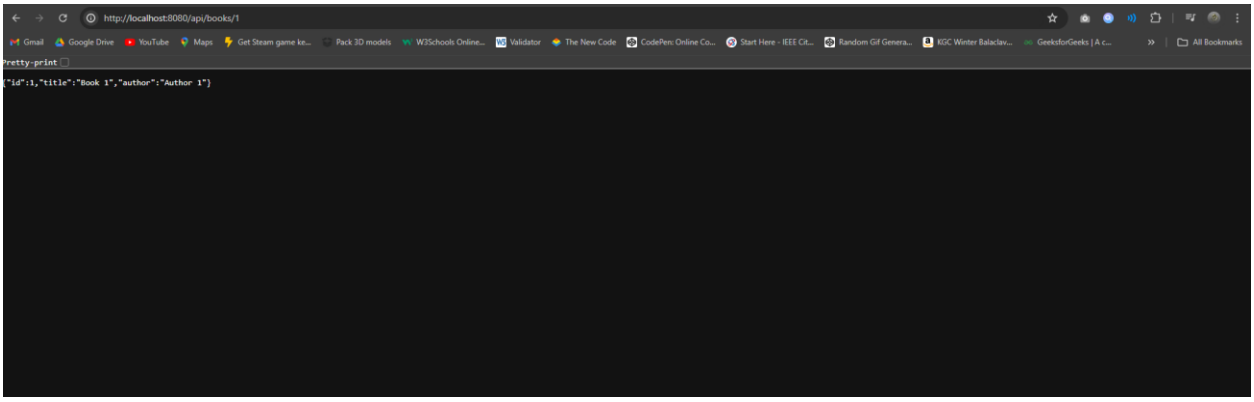
10. Now once these files are ready proceed to initiate this compose
    - docker-compose up

```
PS C:\OS\gitrepo\solution2> docker-compose up
[+] Running 8/8
 ✓ nginx Pulled                                                                                    2.2s
   ✓ f11c1adaa26e Already exists                                                                    0.0s
   ✓ c6b156574604 Already exists                                                                    0.0s
   ✓ ea5d7144c337 Already exists                                                                    0.0s
   ✓ 1bbcb9df2c93 Already exists                                                                    0.0s
   ✓ 537a6cfe3404 Already exists                                                                    0.0s
   ✓ 767bff2cc03e Already exists                                                                    0.0s
   ✓ adc73cb74f25 Already exists                                                                    0.0s
2024/07/05 09:34:56 http2: server: error reading preface from client //./pipe/docker_engine: file has already bee
n closed
[+] Building 0.8s (11/11) FINISHED                                                        docker:default
 => [api internal] load build definition from Dockerfile                                           0.0s
 => => transferring dockerfile: 152B                                                               0.0s
 => [api internal] load metadata for docker.io/library/node:latest                                 0.7s
 => [api auth] library/node:pull token for registry-1.docker.io                                    0.0s
 => [api internal] load .dockerignore                                                              0.0s
 => => transferring context: 2B                                                                    0.0s
 => [api 1/5] FROM docker.io/library/node:latest@sha256:2558f19e787cb0baed81a8068adf7509023b43dedce24ed606  0.0s
 => [api internal] load build context                                                              0.0s
 => => transferring context: 198B                                                                  0.0s
 => CACHED [api 2/5] WORKDIR /app                                                                  0.0s
 => CACHED [api 3/5] COPY package*.json ./                                                         0.0s
```

This command builds the images and starts the containers

11. Once the containers are up and running proceed to open a browser on host machine and go to http://localhost:8080/api/books to test the containers. This shows the JSON of all the books and then again go to http://localhost:8080/api/books/1 to check if the server responds with the JSON of only one book.





12. With this we have completed the challenge 2.

Solution to Challenge-3

1. First, verify that Docker is installed on your host machine by running the following command in the terminal:
   - docker –version

```
Client:
 Cloud integration: v1.0.35+desktop.13
 Version:           26.1.1
 API version:       1.45
 Go version:        go1.21.9
 Git commit:        4cf5afa
 Built:             Tue Apr 30 11:48:43 2024
 OS/Arch:           windows/amd64
 Context:           default

Server: Docker Desktop 4.30.0 (149282)
 Engine:
  Version:          26.1.1
  API version:      1.45 (minimum version 1.24)
  Go version:       go1.21.9
  Git commit:       ac2de55
  Built:            Tue Apr 30 11:48:28 2024
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.31
  GitCommit:        e377cd56a71523140ca6ae87e30244719194a521
 runc:
  Version:          1.1.12
  GitCommit:        v1.1.12-0-g51d5e94
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0
```

2. If Docker is not installed, download Docker Desktop from Docker's official website and follow the installation instructions.
3. After confirming that Docker is installed, ensure Docker Desktop is running on your machine.
4. Navigate to the directory containing the `challenge3` folder.
5. Open this directory in Visual Studio Code by going to terminal and typing **code .** .
6. Create a `.env` file in the root of the `challenge3` directory with the following contents:

```
DB_ROOT_PASSWORD=rootpassword
DB_DATABASE=booksdb
DB_USERNAME=dbuser
DB_PASSWORD=dbpassword
DB_HOST=db
```

7. Create a `docker-compose.yml` file in the root of the `challenge3` folder with the following configuration[2]:

```yaml
services:
  db:
    image: mariadb:latest
    container_name: challenge3-db
    environment:
      MYSQL_ROOT_PASSWORD: ${DB_ROOT_PASSWORD}
      MYSQL_DATABASE: ${DB_DATABASE}
      MYSQL_USER: ${DB_USERNAME}
      MYSQL_PASSWORD: ${DB_PASSWORD}
    volumes:
      - ./db/init:/docker-entrypoint-initdb.d
    networks:
      - app-network

  node-service:
    build: ./api
    container_name: challenge3-node-service
    environment:
      DB_HOST: ${DB_HOST}
      DB_DATABASE: ${DB_DATABASE}
      DB_USERNAME: ${DB_USERNAME}
      DB_PASSWORD: ${DB_PASSWORD}
    networks:
      - app-network
    depends_on:
      - db

  nginx:
    build: ./nginx
    container_name: challenge3-nginx
    ports:
      - "8080:80"
    networks:
      - app-network
    depends_on:
      - node-service

networks:
  app-network:
    driver: bridge
```

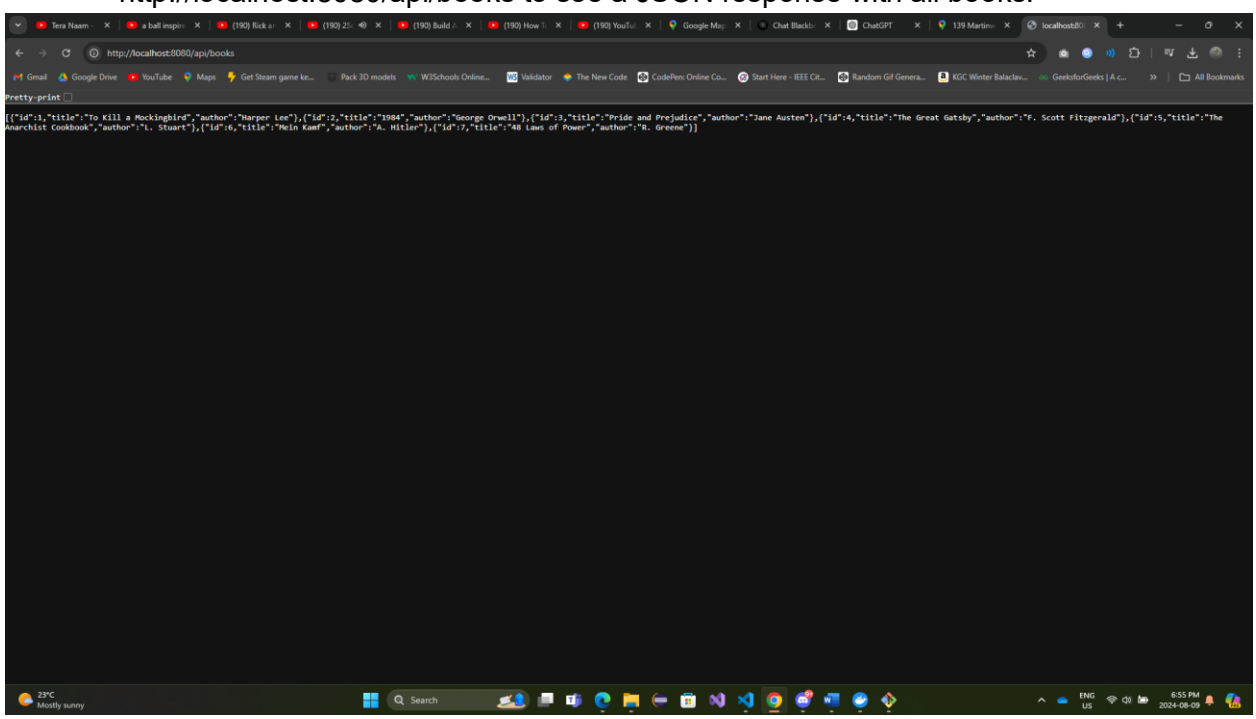
8. Open the terminal in VS Code and execute the following command to build and start the services:

   docker-compose up[2]

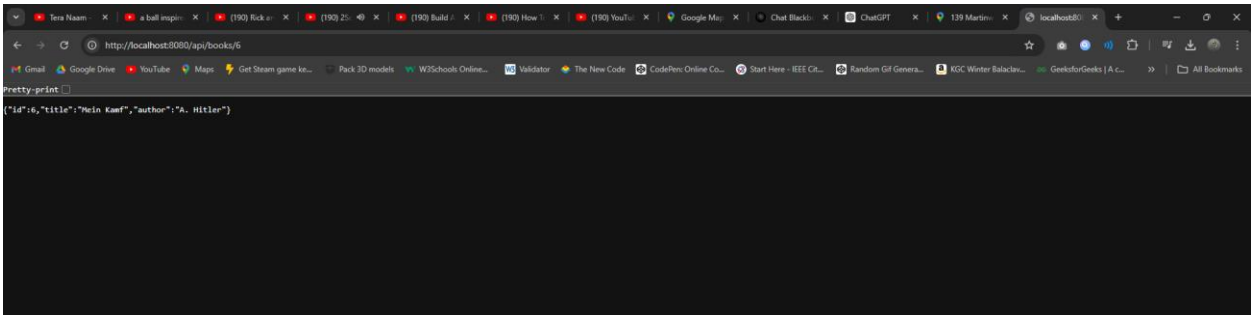
```
challenge3-nginx         | 2024/08/10 00:55:08 [notice] 1#1: start worker process 37
challenge3-node-service  | Server running on port 3000
challenge3-nginx         | 2024/08/10 00:55:08 [notice] 1#1: start worker process 38
challenge3-nginx         | 2024/08/10 00:55:08 [notice] 1#1: start worker process 39
challenge3-db            | 2024-08-10 00:55:08+00:00 [Note] [Entrypoint]: MariaDB upgrade not required
challenge3-db            | 2024-08-10  0:55:08 0 [Note] Starting MariaDB 11.4.2-MariaDB-ubu2404 source revision 3fca5ed772fb75e3e57c507edef2985f8eba5b12 as process 1
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: Compressed tables use zlib 1.3
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: Number of transaction pools: 1
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: Using crc32 + pclmulqdq instructions
challenge3-db            | 2024-08-10  0:55:08 0 [Note] mariadbd: O_TMPFILE is not supported on /tmp (disabling future attempts)
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: Using liburing
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: Initializing buffer pool, total size = 128.000MiB, chunk size = 2.000MiB
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: Completed initialization of buffer pool
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: File system buffers for log disabled (block size=4096 bytes)
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: End of log at LSN=54100
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: Opened 3 undo tablespaces
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: 128 rollback segments in 3 undo tablespaces are active.
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: Setting file './ibtmp1' size to 12.000MiB. Physically writing the file full; Please wait ...
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: File './ibtmp1' size is now 12.000MiB.
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: log sequence number 54100; transaction id 37
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
challenge3-db            | 2024-08-10  0:55:08 0 [Note] Plugin 'FEEDBACK' is disabled.
challenge3-db            | 2024-08-10  0:55:08 0 [Note] Plugin 'wsrep-provider' is disabled.
challenge3-db            | 2024-08-10  0:55:08 0 [Note] InnoDB: Buffer pool(s) load completed at 240810  0:55:08
challenge3-db            | 2024-08-10  0:55:10 0 [Note] Server socket created on IP: '0.0.0.0'.
challenge3-db            | 2024-08-10  0:55:10 0 [Note] Server socket created on IP: '::'.
challenge3-db            | 2024-08-10  0:55:10 0 [Note] mariadbd: Event Scheduler: Loaded 0 events
challenge3-db            | 2024-08-10  0:55:10 0 [Note] mariadbd: ready for connections.
challenge3-db            | Version: '11.4.2-MariaDB-ubu2404'  socket: '/run/mysqld/mysqld.sock'  port: 3306  mariadb.org binary distribution
challenge3-nginx         | 172.18.0.1 - - [10/Aug/2024:00:55:47 +0000] "GET /api/books HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safari
537.36" "-"
challenge3-nginx         | 172.18.0.1 - - [10/Aug/2024:00:56:03 +0000] "GET /api/books/6 HTTP/1.1" 304 0 "-" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/127.0.0.0 Safa
i/537.36" "-"
```

9. Once all the services are up, open your browser and navigate to:

   - http://localhost:8080/api/books to see a JSON response with all books.



   - http://localhost:8080/api/books/1 to see a JSON response with the details of the first book.

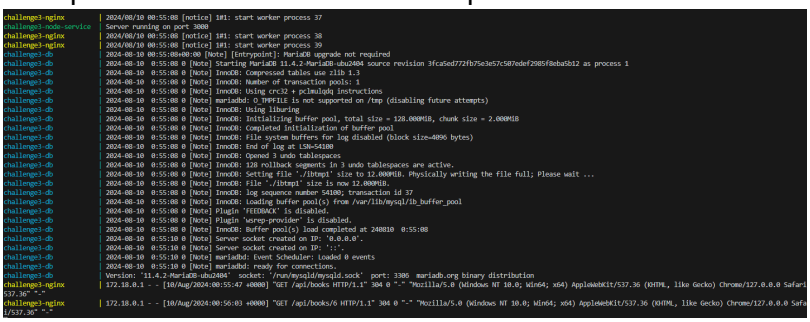


10. To confirm all services are running, execute the following command:
    - docker-compose ps

```
PS C:\OS\Solutions\docker-challenge-template\challenge3> docker-compose ps
NAME                      IMAGE                     COMMAND                  SERVICE        CREATED          STATUS         PORTS
challenge3-db             mariadb:latest            "docker-entrypoint.s…"   db             29 minutes ago   Up 9 seconds   3306/tcp
challenge3-nginx          challenge3-nginx          "/docker-entrypoint.…"   nginx          29 minutes ago   Up 9 seconds   0.0.0.0:8080->80/tcp
challenge3-node-service   challenge3-node-service   "docker-entrypoint.s…"   node-service   29 minutes ago   Up 9 seconds   3000/tcp
PS C:\OS\Solutions\docker-challenge-template\challenge3>
```

Solution to Challenge-4

1. Check if the docker desktop is running.
    - docker version
2. Now proceed to run the docker compose on the root of the challenge 4.



3. Now once the docker compose is running, proceed with executing another command as
    - docker-compose up --scale node-service=3 -d

    This command is going to instantiate 3 node-services that can handle multiple requests and provide with the results and it is good for load balancing and scalability of a project.

```
PS C:\OS\Solutions\docker-challenge-template\challenge4> docker-compose up --scale node-service=3 -d
[+] Running 5/5
 ✓ Container challenge4-db-1            Running
 ✓ Container challenge4-node-service-1  Running
 ✓ Container challenge4-node-service-3  Started
 ✓ Container challenge4-node-service-2  Started
 ✓ Container challenge4-nginx-1         Running
PS C:\OS\Solutions\docker-challenge-template\challenge4>
```

4. Now confirm that using docker-compose ps
5. As now multiple node services are running you can make multiple requests to http://localhost:8080/api/stats from browser or can do the following steps as I did.
6. Open Bash at challenge 4 root and run the following command

```
for i in {1..10}; do curl http://localhost:8080/api/stats; echo "";done
```

7. Now this command is going to do 10 requests to the url and you can see different hostnames responded by the server as:

```
amrit@OhillonPC MINGW64 /c/OS/Solutions/docker-challenge-template/challenge4
$ for i in {1..10}; do curl http://localhost:8080/api/stats; echo "";done
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   120  100   120    0     0  28070      0 --:--:-- --:--:-- --:--:-- 30000{"status":"success","contents":{"MemFree":7189240,"MemAvailable":8533860},"pid":1,"hostname":"3a7d104c195f","counter":0}

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   120  100   120    0     0  31088      0 --:--:-- --:--:-- --:--:-- 40000{"status":"success","contents":{"MemFree":7189240,"MemAvailable":8534152},"pid":1,"hostname":"3a7d104c195f","counter":0}

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   120  100   120    0     0   8741      0 --:--:-- --:--:-- --:--:--  9230{"status":"success","contents":{"MemFree":7188736,"MemAvailable":8533648},"pid":1,"hostname":"ad0924b4037e","counter":0}

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   120  100   120    0     0   7310      0 --:--:-- --:--:-- --:--:--  7500{"status":"success","contents":{"MemFree":7188484,"MemAvailable":8533396},"pid":1,"hostname":"cc8fafc14056","counter":0}

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   120  100   120    0     0   5325      0 --:--:-- --:--:-- --:--:--  5454{"status":"success","contents":{"MemFree":7187980,"MemAvailable":8532892},"pid":1,"hostname":"cc8fafc14056","counter":0}

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   120  100   120    0     0   5566      0 --:--:-- --:--:-- --:--:--  5714{"status":"success","contents":{"MemFree":7187728,"MemAvailable":8532640},"pid":1,"hostname":"ad0924b4037e","counter":0}

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   120  100   120    0     0  32266      0 --:--:-- --:--:-- --:--:-- 40000{"status":"success","contents":{"MemFree":7187728,"MemAvailable":8532640},"pid":1,"hostname":"3a7d104c195f","counter":0}

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   120  100   120    0     0  32867      0 --:--:-- --:--:-- --:--:-- 40000{"status":"success","contents":{"MemFree":7187476,"MemAvailable":8532388},"pid":1,"hostname":"3a7d104c195f","counter":0}

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   120  100   120    0     0  26905      0 --:--:-- --:--:-- --:--:-- 30000{"status":"success","contents":{"MemFree":7187476,"MemAvailable":8532388},"pid":1,"hostname":"ad0924b4037e","counter":0}

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   120  100   120    0     0  28328      0 --:--:-- --:--:-- --:--:-- 30000{"status":"success","contents":{"MemFree":7187224,"MemAvailable":8532144},"pid":1,"hostname":"cc8fafc14056","counter":0}
```

And there you have it and you have completed challenge 4.

References

[1] P. McKee, "How to Use the NGINX Docker Official Image ," docker. Accessed: Abbreviated July 3,2024.. [Online]. Available: https://www.docker.com/blog/how-to-use-the-official-nginx-docker-image/

[2] "Docker compose up," docker.docs . Accessed: Abbreviated August 8,2024.. [Online]. Available: https://docs.docker.com/reference/cli/docker/compose/up/