

### Challenge-3:

Please find below the detailed steps to have a full stack application, running a node server, a database for storing the data and a nginx web server, running on docker containers using docker compose:






1. Go to the following link and download the latest version of Docker desktop for your OS:  
<https://www.docker.com/products/docker-desktop/>
2. Follow the instructions in the docker installer and complete the installation with the default configurations provided.
3. Run the docker desktop application and verify that it is installed correctly by opening a command prompt and typing the following command:  
docker --version

```
C:\Users\jawad>docker --version
Docker version 25.0.3, build 4defb41

C:\Users\jawad>
```

Seeing the docker version as output verifies that our installation was successful.

4. Create a folder for this challenge 3. Inside this folder create three subfolders, called api, db and nginx. These folders will contain the code for the servers, any configuration required and the docker files for each of these services.
5. Inside the folder for challenge 3, we will also have two more files, a docker compose file to run all the three services together and a .env file to hold the environment variables for the application as well as the database service.
6. Our folder structure will look something like the picture shown below:

 docker-compose.yml	2024-04-19 7:37 PM	Yaml Source File
 .env	2024-04-19 5:44 PM	ENV File
 api	2024-04-19 7:56 PM	File folder
 db	2024-04-19 7:56 PM	File folder
 nginx	2024-04-19 7:56 PM	File folder

7. Inside the api folder, we will have three files, one server.js file that contains the server code, exposes our APIs and connects to the database, one package.json file that contains the dependencies that are needed to be installed for our server code to run, and one Docker file that contains the instructions that are used to build the docker image.
8. Inside the db folder, we will have two files. One init.sql file to create and set up our database and insert the values into the database that will be fetched by our APIs, and one Docker file to build the image for the database service.

9. Inside the nginx folder we will have two files. One nginx.conf file that will configure our nginx web server to forward all the request it receives to our node API server, and one Docker file to build the image for our nginx server. Our docker file also changes the default configuration of the nginx server to use the configuration file that we have created while building the image.
10. Our .env file which holds the environment variables for the application and the database servers, looks like this:

```
# Application Config

DB_ROOT_PASSWORD=myrootpassword
DB_DATABASE=mydatabase
DB_USERNAME=myuser
DB_PASSWORD=mypassword
DB_HOST=db


# Database Config
MYSQL_ROOT_PASSWORD=myrootpassword
MYSQL_DATABASE=mydatabase
MYSQL_USER=myuser
MYSQL_PASSWORD=mypassword
MYSQL_HOST=localhost
```

We have separate environment variables for the two servers. They are consumed by each of the services respectively while building the images for them, which is done in the docker compose file.

11. The code for our Docker compose file is shown below:

```
version: "3.8"
```

```
services:
```

```
  api-server:
```

```
    build:
```

```
      context: ./api
```

```
      dockerfile: Dockerfile
```

```
    ports:
```

```
      - "3000:3000"
```

```
    environment:
```

```
      DB_DATABASE: ${DB_DATABASE}
```

```
      DB_USERNAME: ${DB_USERNAME}
```

```
      DB_PASSWORD: ${DB_PASSWORD}
```

```
      DB_HOST: ${DB_HOST}
```

```
    depends_on:
```

```
      - db
```

```
  reverse-proxy:
```

```
    build:
```

```
      context: ./nginx
```

```
      dockerfile: Dockerfile
```

```
    ports:
```

```
      - "8080:80"
```

```
    depends_on:
```

```
      - api-server
```

```
  db:
```

```
    build:
```

```
      context: ./db
```

```
      dockerfile: Dockerfile
```

```
    environment:
```

```
      MYSQL_ROOT_PASSWORD: ${MYSQL_ROOT_PASSWORD}
```

```
      MYSQL_DATABASE: ${MYSQL_DATABASE}
```

```
      MYSQL_USER: ${MYSQL_USER}
```

```
      MYSQL_PASSWORD: ${MYSQL_PASSWORD}
```

```
    ports:
```

```
      - "3306:3306"
```

The API server runs on port 3000. It consumes the application environment variables, and it depends on the db service, which means Docker starts the db service first before starting the API server.

The nginx web server called as reverse-proxy here runs on port 8080 and depends on the api server to run first.

Our db server consumes the environment variables for the database and runs on port 3306.

12. Open command prompt and navigate to the challenge 3 directory that has your 3 services folders and your .env and docker compose files.
13. Run the command 'docker-compose up -d' to run our three services. We get the output shown below:

```
C:\Users\javad\Downloads\OS Docker Challenge Final\Challenge 3>docker-compose up -d
2024/04/22 02:58:23 http2: server: error reading preface from client //./pipe/docker_engine: file has already been closed
[+] Building 1.0s (3/3)                                docker:default
=> [db internal] load build definition from Dockerfile                                0.0s
[+] Building 1.2s (8/8)                                docker:default
=> [db internal] load build definition from Dockerfile                                0.0s
[+] Building 1.9s (11/11)                              docker:default
=> [db internal] load build definition from Dockerfile                                0.0sp
[+] Building 2.9s (28/28) FINISHED                docker:default
=> [db internal] load build definition from Dockerfile                                0.0s
=> => transferring dockerfile: 93B                                                    0.0s
=> [db internal] load metadata for docker.io/library/mariadb:latest                  1.0s
=> [db auth] library/mariadb:pull token for registry-1.docker.io                    0.0s
=> [db internal] load .dockerignore                                                  0.0s
=> => transferring context: 2B                                                         0.0s
=> [db internal] load build context                                                  0.0s
=> => transferring context: 30B                                                         0.0s
=> [db 1/2] FROM docker.io/library/mariadb:latest@sha256:78d0c3b8c39                0.0s
=> CACHED [db 2/2] COPY init.sql /docker-entrypoint-initdb.d                      0.0s
=> [db] exporting to image                                                            0.0sp
=> => exporting layers                                                                0.0s
=> => writing image sha256:a7aee433d55ee2fabda997439487eda2bf8c1ba6a                0.0s
=> => naming to docker.io/library/challenge3-db                                    0.0s
=> [api-server internal] load build definition from Dockerfile                      0.0s
=> => transferring dockerfile: 449B                                                    0.0s
=> [api-server internal] load metadata for docker.io/library/node:al                0.6sp
=> [api-server auth] library/node:pull token for registry-1.docker.i                0.0s
=> [api-server internal] load .dockerignore                                          0.0s
=> => transferring context: 2B                                                         0.0s
=> [api-server 1/5] FROM docker.io/library/node:alpine@sha256:6d0f18                0.0s
=> [api-server internal] load build context                                          0.0s
=> => transferring context: 93B                                                         0.0s
=> CACHED [api-server 2/5] WORKDIR /app                                              0.0s
=> CACHED [api-server 3/5] COPY package*.json ./                                  0.0s
=> CACHED [api-server 4/5] RUN npm install                                           0.0s
=> CACHED [api-server 5/5] COPY * .                                                  0.0s
=> [api-server] exporting to image                                                    0.0s
=> => exporting layers                                                                0.0s
=> => writing image sha256:94e878c3cc76993fd03b7c636b2f504ce4370586c                0.0s
=> => naming to docker.io/library/challenge3-api-server                            0.0s
```

```

=> [reverse-proxy internal] load build definition from Dockerfile      0.0s
=> => transferring dockerfile: 132B                                   0.0s
=> [reverse-proxy internal] load metadata for docker.io/library/nginx 0.6s
=> [reverse-proxy auth] library/nginx:pull token for registry-1.dock 0.0s
=> [reverse-proxy internal] load .dockerignore                        0.0s
=> => transferring context: 2B                                         0.0s
=> [reverse-proxy 1/3] FROM docker.io/library/nginx:latest@sha256:04 0.0s
=> [reverse-proxy internal] load build context                        0.0s
=> => transferring context: 31B                                         0.0s
=> CACHED [reverse-proxy 2/3] RUN rm /etc/nginx/conf.d/default.conf  0.0s
=> CACHED [reverse-proxy 3/3] COPY nginx.conf /etc/nginx/conf.d/defa 0.0s
=> [reverse-proxy] exporting to image                                 0.0s
=> => exporting layers                                                 0.0s
=> => writing image sha256:ac4e8cc2f615c32842ba765bd57e742dbb9eed46a 0.0s
=> => naming to docker.io/library/challenge3-reverse-proxy            0.0s
[+] Running 3/4
- Network challenge3_default          Created                  1.2s
✔ Container challenge3-db-1           Started                 0.6s
✔ Container challenge3-api-server-1    Started                 0.8s
✔ Container challenge3-reverse-proxy-1 Started                 1.1s

```

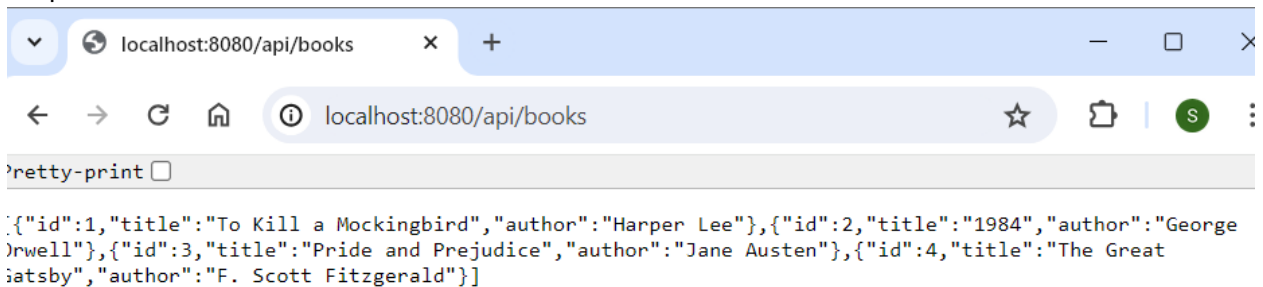
14. Run the command ‘docker-compose ps’ to check that all three of our services are running as expected. The output is shown below:

```

C:\Users\javad\Downloads\OS Docker Challenge Final\Challenge 3>docker-compose ps
NAME                IMAGE                COMMAND                  SERVICE    CREATED   STATUS    PORTS
challenge3-api-server-1 challenge3-api-server "docker-entrypoint.s..." api-server 5 minutes ago Up 5 minutes 0.0.0.0:3000->3000/tcp
challenge3-db-1      challenge3-db        "docker-entrypoint.s..." db          5 minutes ago Up 5 minutes 0.0.0.0:3306->3306/tcp
challenge3-reverse-proxy-1 challenge3-reverse-proxy "/docker-entrypoint..." reverse-proxy 5 minutes ago Up 5 minutes 0.0.0.0:8080->80/tcp

```

15. Open your web browser and go to the url ‘http://localhost:8080/api/books’. You should the output as below:



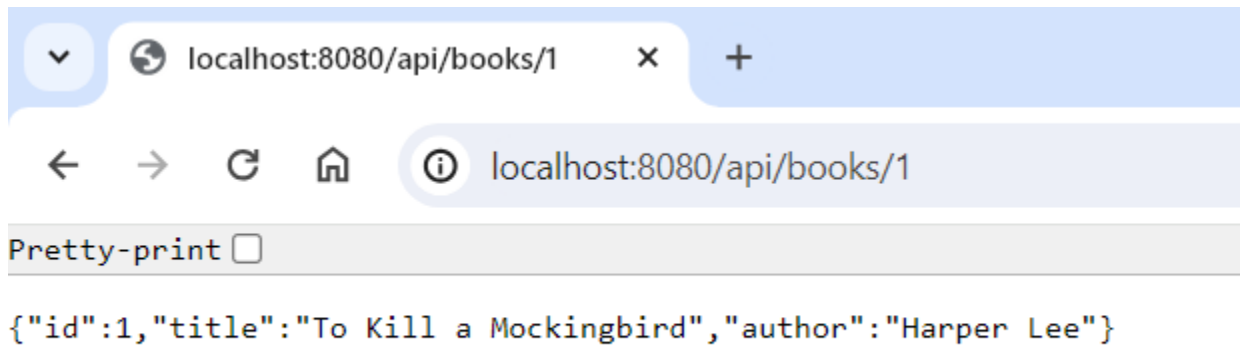
```

[{"id":1,"title":"To Kill a Mockingbird","author":"Harper Lee"}, {"id":2,"title":"1984","author":"George Orwell"}, {"id":3,"title":"Pride and Prejudice","author":"Jane Austen"}, {"id":4,"title":"The Great Gatsby","author":"F. Scott Fitzgerald"}]

```

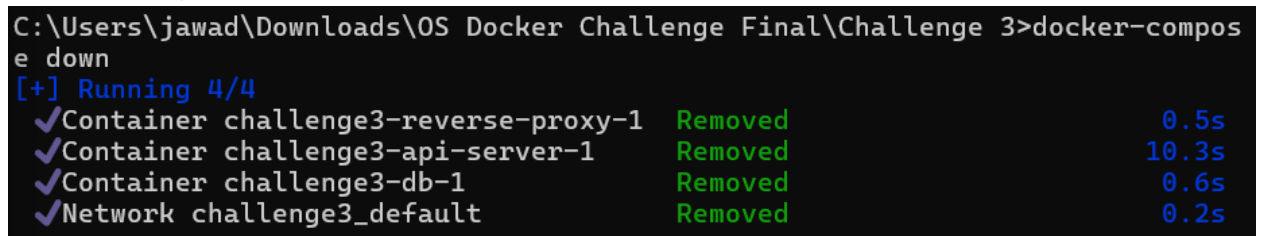
It shows all the books stored in our database.

16. If you want to see a particular book, just enter it’s ID at the end of the url. For example if you go to ‘http://localhost:8080/api/books/1’, you should see the output as below:



```
localhost:8080/api/books/1
{
  "id": 1,
  "title": "To Kill a Mockingbird",
  "author": "Harper Lee"
}
```

17. Run the command 'docker-compose down' to remove the services and free the ports being used. The output is shown below:

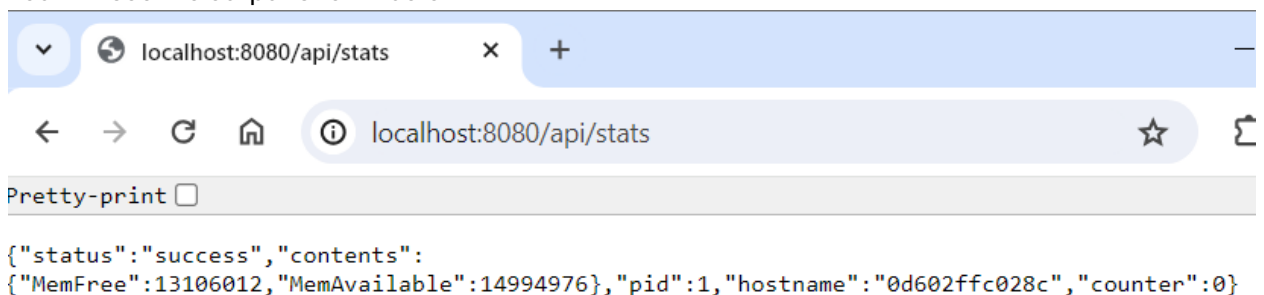


```
C:\Users\jawad\Downloads\OS Docker Challenge Final\Challenge 3>docker-compose down
[+] Running 4/4
✓Container challenge3-reverse-proxy-1    Removed      0.5s
✓Container challenge3-api-server-1      Removed     10.3s
✓Container challenge3-db-1              Removed      0.6s
✓Network challenge3_default             Removed      0.2s
```

#### Challenge-4:

Please find below, the detailed steps to scale up a service in docker:

1. Copy all the five files and folders from challenge 3 directory to a new directory called challenge 4.
2. Run the services using the same code as used in challenge 3, using the docker compose commands. Open your web browser and navigate to the url 'http://localhost:8080/api/stats'. You will see the output shown below:



```
localhost:8080/api/stats
{
  "status": "success",
  "contents": {
    "MemFree": 13106012,
    "MemAvailable": 14994976,
    "pid": 1,
    "hostname": "0d602ffc028c",
    "counter": 0
  }
}
```

You will note that the hostname value remains the same no matter how many times you reload this page. It is because we are using only one instance of our API server.

3. Change the api-server configuration in your docker compose file to deploy three instances of the node server. The changed code is shown below:

```

api-server:
  build:
    context: ./api
    dockerfile: Dockerfile
  ports:
    - "3000"
  environment:
    DB_DATABASE: ${DB_DATABASE}
    DB_USERNAME: ${DB_USERNAME}
    DB_PASSWORD: ${DB_PASSWORD}
    DB_HOST: ${DB_HOST}
  depends_on:
    - db
  deploy:
    replicas: 3

```

As you can see that we have added a deploy field in the file, under which it says that we want to create three replicas of our node server.

Also, you can see that in the value of ports we only mention the host machine's port which is 3000 but we do not map it to any port for our container. It is because there will be three separate containers running our node server, so we can just let docker handle the mapping. No other change in code is required.

4. Open command prompt and navigate to the directory challenge 4.
5. Run the command 'docker-compose up -d'. You will the output as below:

```

C:\Users\jawaad\Downloads\OS Docker Challenge Final\Challenge 4>docker-compose up -d
[+] Running 5/5
✓Container challenge4-db-1           Started           0.0s
✓Container challenge4-api-server-2   Started           0.0s
✓Container challenge4-api-server-3   Started           0.0s
✓Container challenge4-api-server-1   Started           0.0s
✓Container challenge4-reverse-proxy-1 Started           0.0s

```

You can see from the output that we have 3 instances of our API server running.

6. Run the command 'docker-compose ps' to see that all our services are running as expected. The output is shown below:

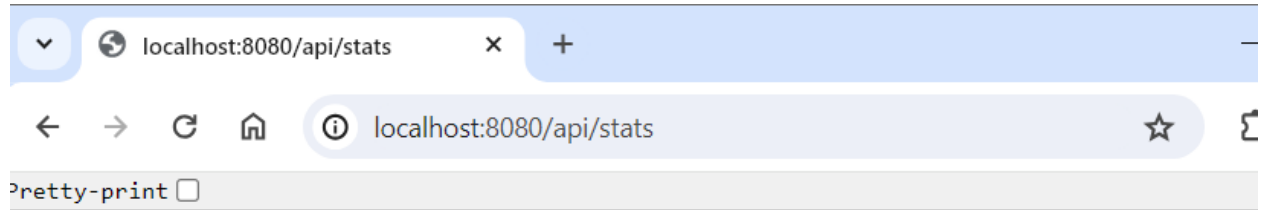
```

C:\Users\jawaad\Downloads\OS Docker Challenge Final\Challenge 4>docker-compose ps
NAME                                IMAGE                COMMAND              SERVICE    CREATED    STATUS      PORTS
challenge4-api-server-1             challenge4-api-server "docker-entrypoint.s" api-server  2 days ago Up 39 seconds 0.0.0.0:51056->3000/tcp
challenge4-api-server-2             challenge4-api-server "docker-entrypoint.s" api-server  2 days ago Up 40 seconds 0.0.0.0:51054->3000/tcp
challenge4-api-server-3             challenge4-api-server "docker-entrypoint.s" api-server  2 days ago Up 40 seconds 0.0.0.0:51055->3000/tcp
challenge4-db-1                     challenge4-db         "docker-entrypoint.s" db          2 days ago Up 40 seconds 0.0.0.0:3306->3306/tcp
challenge4-reverse-proxy-1           challenge4-reverse-proxy "/docker-entrypoint..." reverse-proxy 2 days ago Up 39 seconds 0.0.0.0:8080->80/tcp

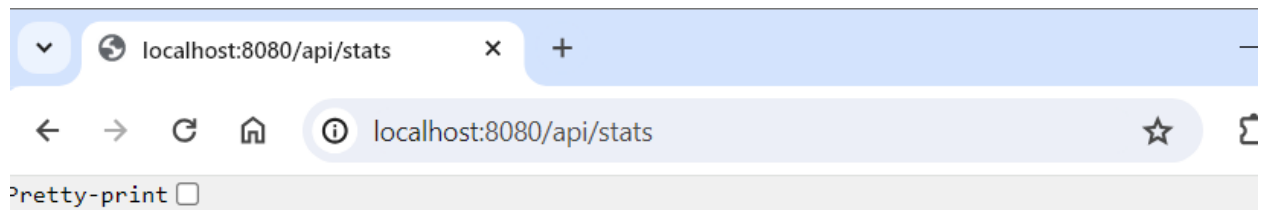
```

The output shows a total of 5 services running.

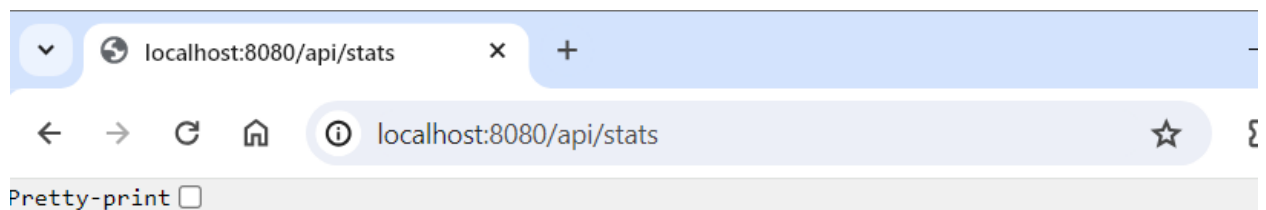
- Open a web browser and navigate to the url 'http://localhost:8080/api/stats'. Now you will see that we have a total of three different hostname values and we see one of three values each time we reload the page.



```
{"status": "success", "contents": {"MemFree": 12918656, "MemAvailable": 14808840, "pid": 1, "hostname": "6c5c6a7d2ef0", "counter": 0}}
```



```
{"status": "success", "contents": {"MemFree": 12919600, "MemAvailable": 14809768, "pid": 1, "hostname": "3dd9426b91f7", "counter": 0}}
```



```
{"status": "success", "contents": {"MemFree": 12915880, "MemAvailable": 14806056, "pid": 1, "hostname": "0a8fdef1ef1c", "counter": 0}}
```

This proves that we have successfully scaled our application to run three instances of our api server.

- Run the command 'docker-compose down' to stop all our services and free up the ports used. The output is shown below:

```
C:\Users\jawad\Downloads\OS Docker Challenge Final\Challenge 4>docker-compose down
[+] Running 6/6
✓Container challenge4-reverse-proxy-1   Removed
✓Container challenge4-api-server-1      Removed
✓Container challenge4-api-server-2      Removed
✓Container challenge4-api-server-3      Removed
✓Container challenge4-db-1              Removed
✓Network challenge4_default             Removed
```



#### References:

[1]: <https://docs.docker.com/compose/environment-variables/>

[2]: <https://www.appsdeveloperblog.com/scaling-with-docker-compose-a-beginners-guide/>

[3]: [https://hub.docker.com/\\_/mariadb](https://hub.docker.com/_/mariadb)