Національний Технічний Університет України
“КПІ ім. Ігоря Сікорського”
Інститут прикладного системного аналізу

ЛАБОРАТОРНА РОБОТА № 1

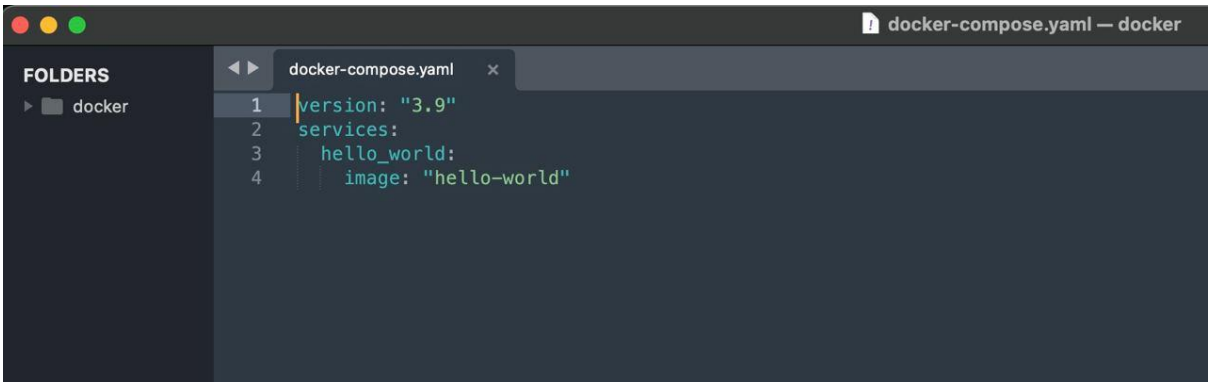
Тема: Docker

Виконали:

студенти гр. КІ-21мп
Безимянний Олексій
Овчаренко Олександр
Кудрєв Денис

Київ 2022

task_1




```
(base) macbookair@MacBook-Air-MacBook docker % docker compose up
[+] Running 2/1
 ⠿ Network docker_default          Created
 ⠿ Container docker-hello_world-1  Created
Attaching to docker-hello_world-1
docker-hello_world-1  |
docker-hello_world-1  | Hello from Docker!
docker-hello_world-1  | This message shows that your installation appears to be working correctly.
docker-hello_world-1  |
docker-hello_world-1  | To generate this message, Docker took the following steps:
docker-hello_world-1  |  1. The Docker client contacted the Docker daemon.
docker-hello_world-1  |  2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
docker-hello_world-1  |     (arm64v8)
docker-hello_world-1  |  3. The Docker daemon created a new container from that image which runs the
docker-hello_world-1  |     executable that produces the output you are currently reading.
docker-hello_world-1  |  4. The Docker daemon streamed that output to the Docker client, which sent it
docker-hello_world-1  |     to your terminal.
docker-hello_world-1  |
docker-hello_world-1  | To try something more ambitious, you can run an Ubuntu container with:
docker-hello_world-1  |  $ docker run -it ubuntu bash
docker-hello_world-1  |
docker-hello_world-1  | Share images, automate workflows, and more with a free Docker ID:
docker-hello_world-1  |  https://hub.docker.com/
docker-hello_world-1  |
docker-hello_world-1  | For more examples and ideas, visit:
docker-hello_world-1  |  https://docs.docker.com/get-started/
docker-hello_world-1  |
docker-hello_world-1 exited with code 0
```

task_2

docker-compose.yaml

Scroll Tabs

```yaml
version: "3.9"

services:
  lite_server:
    container_name: lite_server_name
    build:
      context: .
      dockerfile: Dockerfile
    ports:
      - "8002:3000"
    restart: "no"
```

Dockerfile

```
FROM node

WORKDIR /usr/src/app
COPY package*.json ./

RUN npm install
RUN npm install --global lite-server
COPY . .

EXPOSE 3000

CMD npm run start
```

index.html

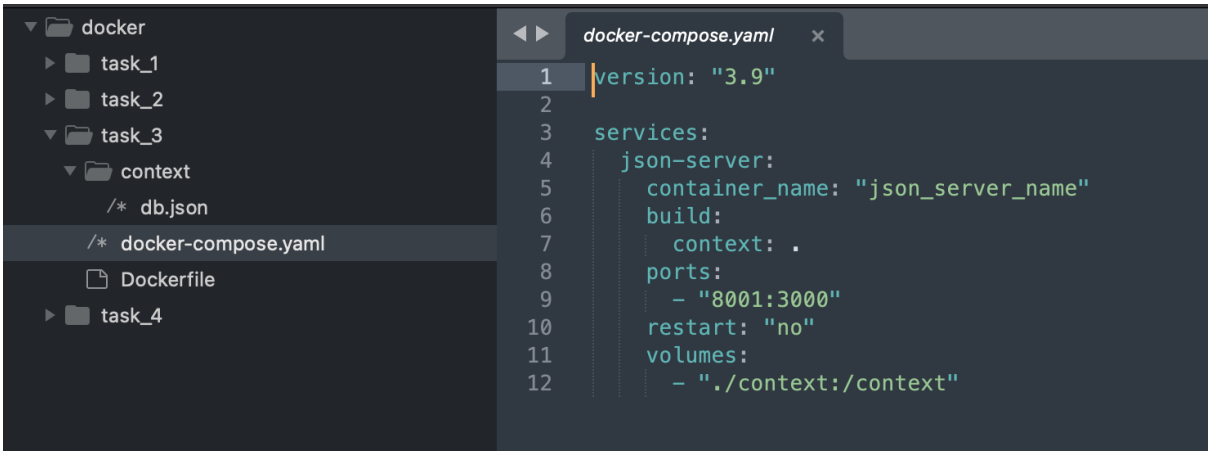
```html
<html>
<head>
    <title>Task_2</title>
</head>
<body>
<img class="picture"
     src="image.jpg"
     alt="Charon">
</body>
</html>
```

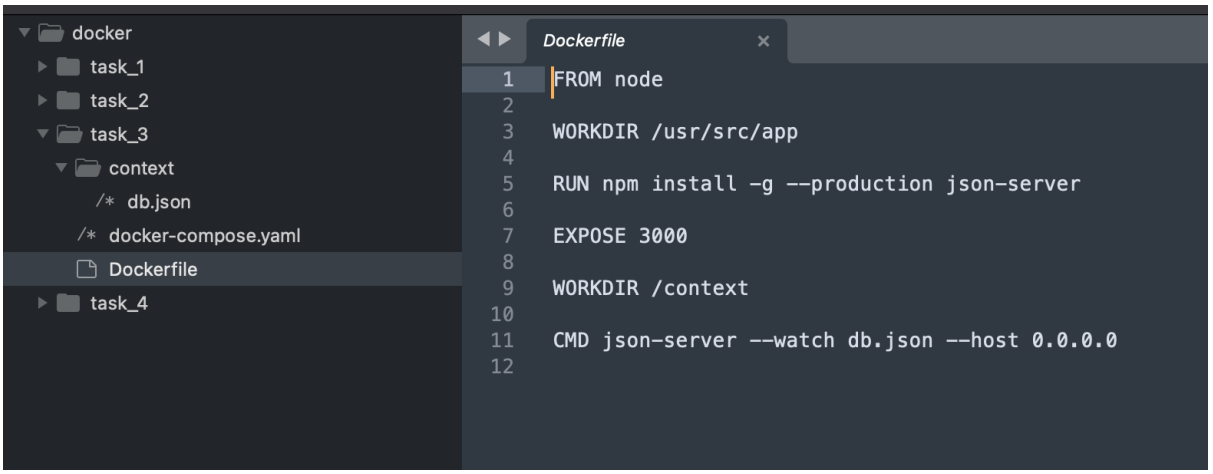
package.json

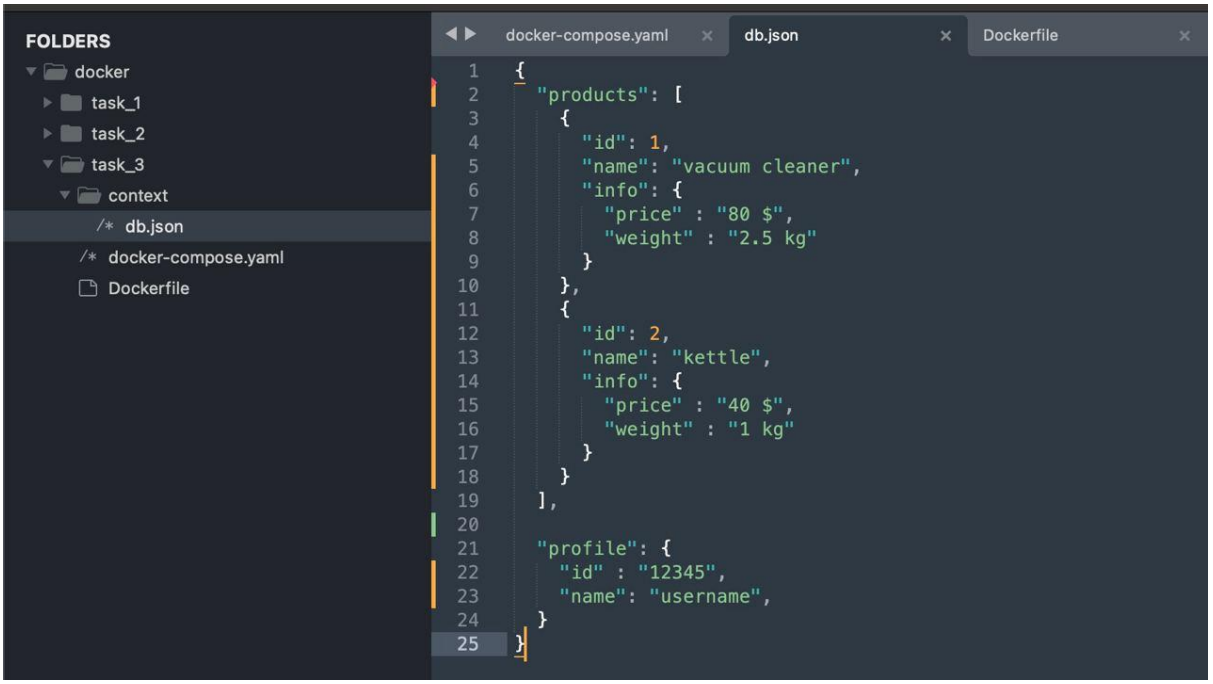
```
{
  "main": "index.html",
  "devDependencies": {
    "lite-server": "^2.2.0"
  },
  "scripts": {
    "start": "lite-server"
  }
}
```

```
(base) macbookair@MacBook-Air-MacBook task_2 % docker compose up
[+] Building 2.1s (11/11) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 168B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/node:latest
 => [1/6] FROM docker.io/library/node
 => [internal] load build context
 => => transferring context: 542.37kB
 => CACHED [2/6] WORKDIR /usr/src/app
 => CACHED [3/6] COPY package*.json ./
 => CACHED [4/6] RUN npm install
 => CACHED [5/6] RUN npm install --global lite-server
 => [6/6] COPY . .
 => exporting to image
 => => exporting layers
 => => writing image sha256:5a58de9496a00a4e55d76f2cc9aa228d297fe765e36e3e6a0c3f19ca2db05b93
 => => naming to docker.io/library/task_2-lite_server

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix the
[+] Running 2/2
 ⠿ Network task_2_default      Created
 ⠿ Container lite_server_name  Created
```

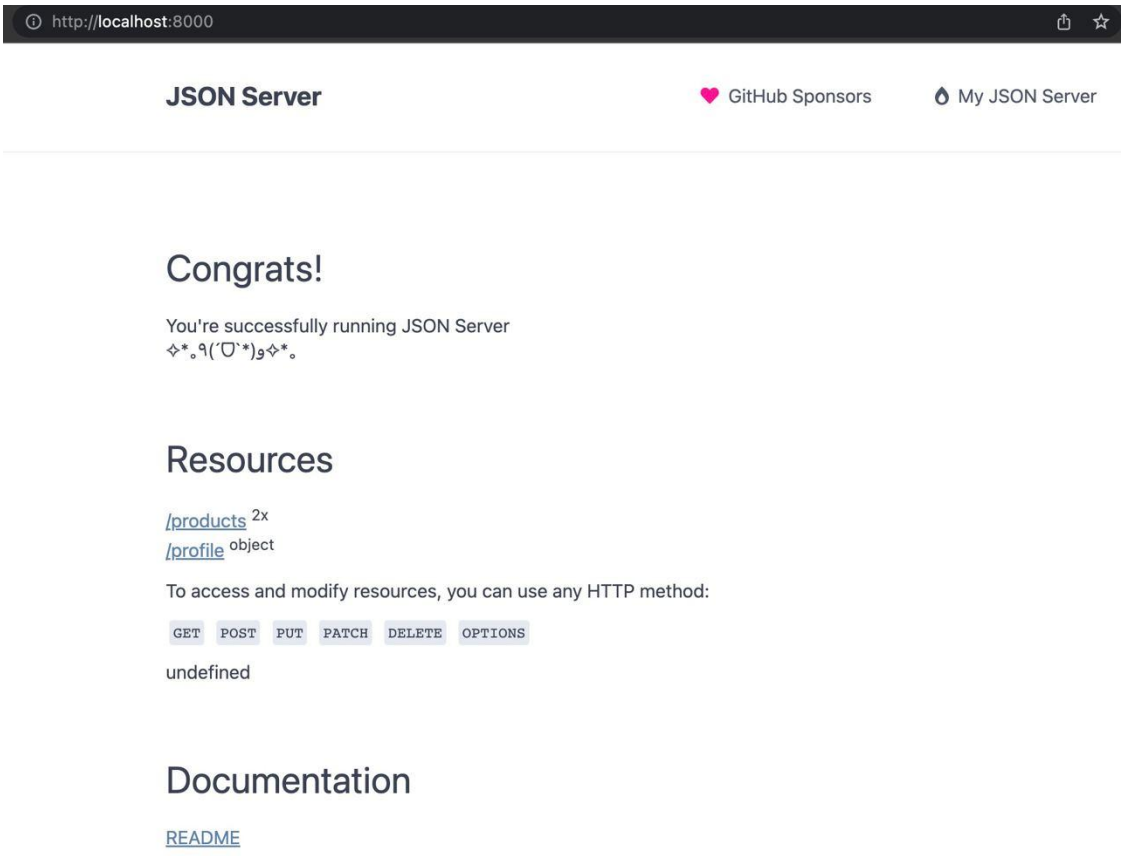
task_3

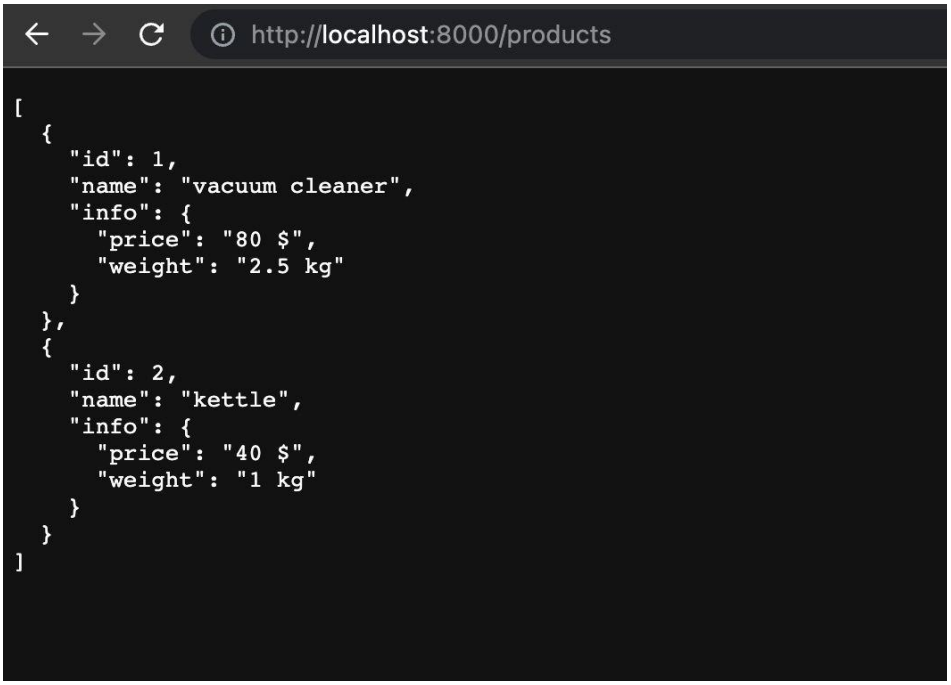

docker
task_1
task_2
task_3
context
db.json
docker-compose.yaml
Dockerfile
task_4
docker-compose.yaml
version: "3.9"
services:
json-server:
container_name: "json_server_name"
build:
context: .
ports:
- "8001:3000"
restart: "no"
volumes:
- "./context:/context"


docker
task_1
task_2
task_3
context
db.json
docker-compose.yaml
Dockerfile
task_4
Dockerfile
FROM node
WORKDIR /usr/src/app
RUN npm install -g --production json-server
EXPOSE 3000
WORKDIR /context
CMD json-server --watch db.json --host 0.0.0.0

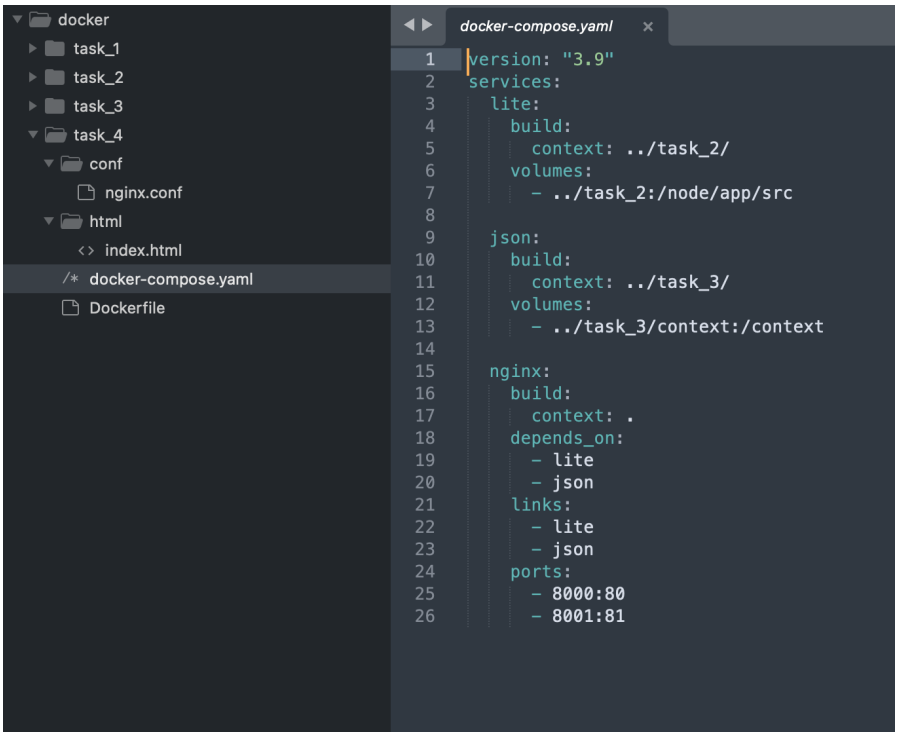

FOLDERS
docker
task_1
task_2
task_3
context
db.json
docker-compose.yaml
Dockerfile
docker-compose.yaml
db.json
Dockerfile
{
"products": [
{
"id": 1,
"name": "vacuum cleaner",
"info": {
"price" : "80 $",
"weight" : "2.5 kg"
}
},
{
"id": 2,
"name": "kettle",
"info": {
"price" : "40 $",
"weight" : "1 kg"
}
}
],
"profile": {
"id" : "12345",
"name": "username",
}
}

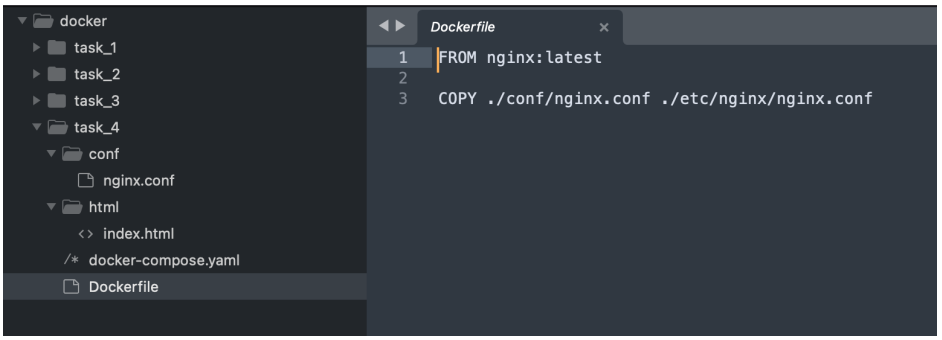
```
(base) macbookair@MacBook-Air-MacBook task_3 % docker compose up
[+] Building 0.2s (8/8) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 32B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/node:latest
 => [1/4] FROM docker.io/library/node
 => CACHED [2/4] WORKDIR /usr/src/app
 => CACHED [3/4] RUN npm install -g --production json-server
 => CACHED [4/4] WORKDIR /context
 => exporting to image
 => => exporting layers
 => => writing image sha256:3227966eba78a08ded68423a8c464dff84d148c269a642343c3709716a50064b
 => => naming to docker.io/library/task_3-json-server

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
[+] Running 2/1
 ⠿ Network task_3_default  Created
 ⠿ Container json_server   Created
Attaching to json_server
json_server  |
json_server  |   \{^_^}/ hi!
json_server  |
json_server  |   Loading db.json
json_server  |   Done
json_server  |
json_server  |   Resources
json_server  |   http://0.0.0.0:3000/products
json_server  |   http://0.0.0.0:3000/profile
json_server  |
json_server  |   Home
json_server  |   http://0.0.0.0:3000
json_server  |
json_server  |   Type s + enter at any time to create a snapshot of the database
json_server  |   Watching...
json_server  |
json_server  | GET /db 200 6.623 ms - 346
json_server  | GET /__rules 404 14.904 ms - 2
json_server  | GET /products 200 8.218 ms - 232
json_server  | GET /profile 200 4.919 ms - 41
```

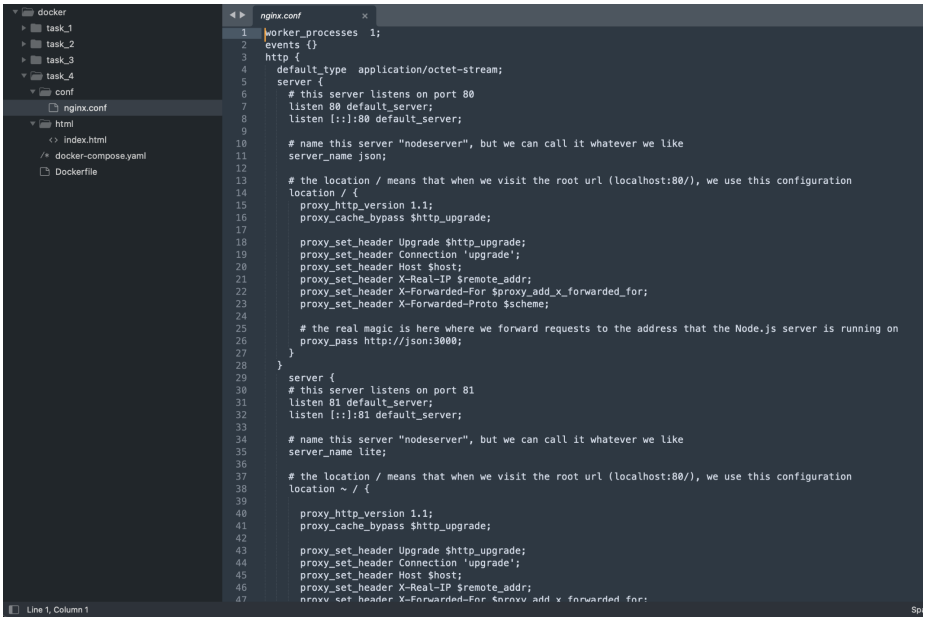



http://localhost:8000/products
[
{
"id": 1,
"name": "vacuum cleaner",
"info": {
"price": "80 $",
"weight": "2.5 kg"
}
},
{
"id": 2,
"name": "kettle",
"info": {
"price": "40 $",
"weight": "1 kg"
}
}
]

task_4


docker
task_1
task_2
task_3
task_4
conf
nginx.conf
html
index.html
docker-compose.yaml
Dockerfile
docker-compose.yaml
version: "3.9"
services:
lite:
build:
context: ../task_2/
volumes:
- ../task_2:/node/app/src
json:
build:
context: ../task_3/
volumes:
- ../task_3/context:/context
nginx:
build:
context: .
depends_on:
- lite
- json
links:
- lite
- json
ports:
- 8000:80
- 8001:81


docker
task_1
task_2
task_3
task_4
conf
nginx.conf
html
index.html
docker-compose.yaml
Dockerfile
Dockerfile
FROM nginx:latest
COPY ./conf/nginx.conf ./etc/nginx/nginx.conf


nginx.conf
worker_processes 1;
events {}
http {
default_type application/octet-stream;
server {
# this server listens on port 80
listen 80 default_server;
listen [::]:80 default_server;
name this server "nodeserver", but we can call it whatever we like
server_name json;
# the location / means that when we visit the root url (localhost:80/), we use this configuration
location / {
proxy_http_version 1.1;
proxy_cache_bypass $http_upgrade;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection 'upgrade';
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;
the real magic is here where we forward requests to the address that the Node.js server is running on
proxy_pass http://json:3000;
}
}
server {
# this server listens on port 81
listen 81 default_server;
listen [::]:81 default_server;
name this server "nodeserver", but we can call it whatever we like
server_name lite;
# the location / means that when we visit the root url (localhost:80/), we use this configuration
location ~ / {
proxy_http_version 1.1;
proxy_cache_bypass $http_upgrade;
proxy_set_header Upgrade $http_upgrade;
proxy_set_header Connection 'upgrade';
proxy_set_header Host $host;
proxy_set_header X-Real-IP $remote_addr;

http://localhost:8000

JSON Server

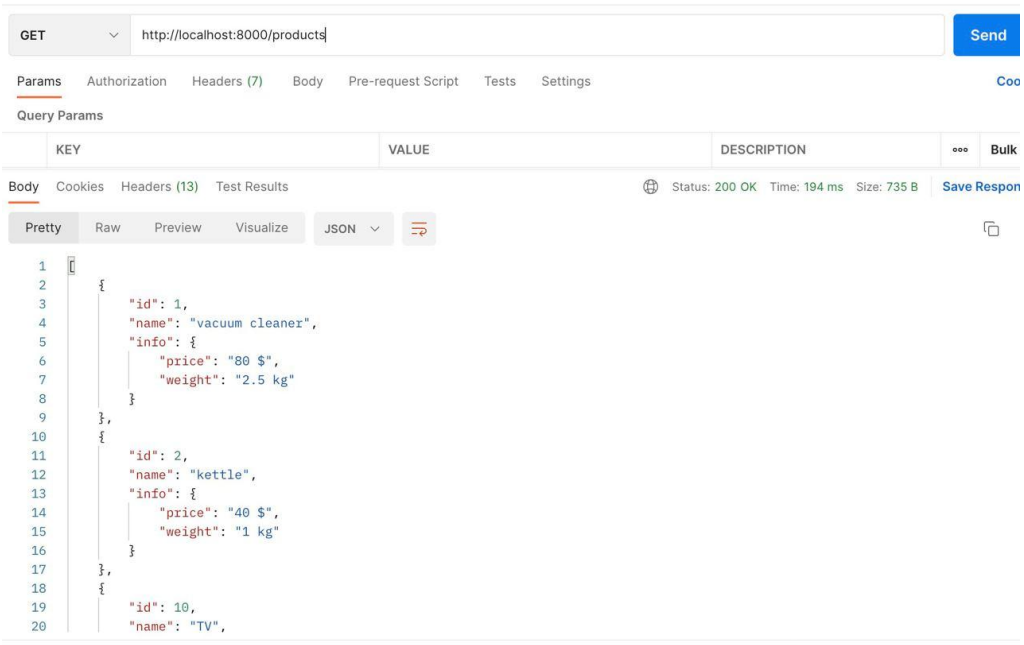
http://localhost:8001

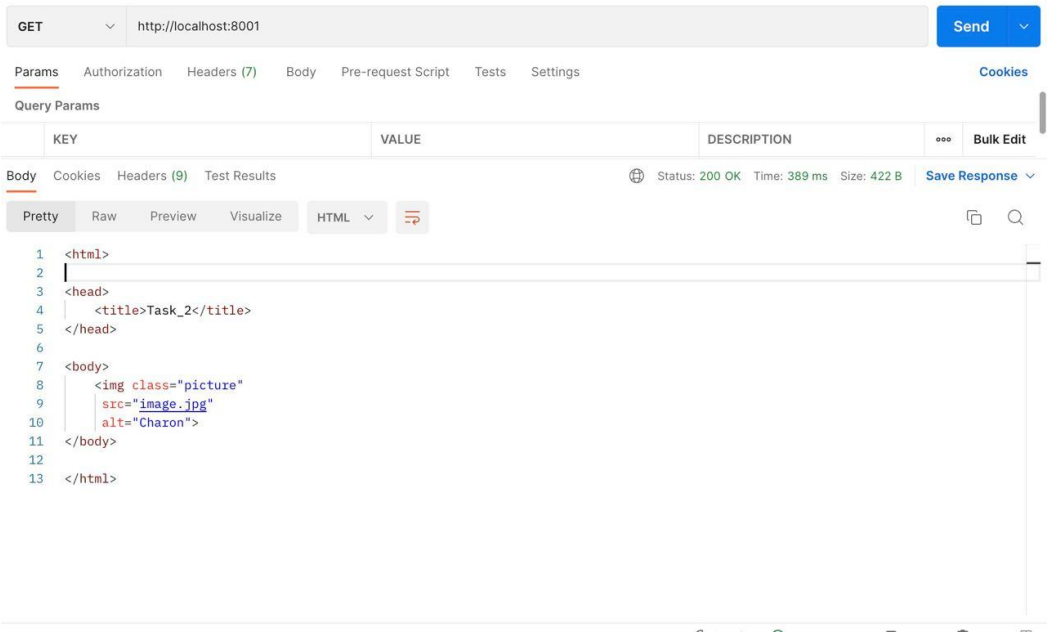

```
(base) macbookair@MacBook-Air-MacBook task_4 % docker compose up --scale lite=3 --scale json=3
[+] Running 7/7
 ⠿ Container task_4-json-1   Created
 ⠿ Container task_4-lite-3   Created
 ⠿ Container task_4-json-3   Created
 ⠿ Container task_4-lite-1   Created
 ⠿ Container task_4-lite-2   Created
 ⠿ Container task_4-json-2   Created
 ⠿ Container task_4-nginx-1  Created
```

ale more than one service, you need to mention them individually with the scale flag
er parameter to ensure that the desired number of instances are created. For exam
have two lite and three json services, you need to do something like this:

task_5

```
(base) macbookair@MacBook-Air-MacBook task_4 % curl -X POST -H "Content-Type: application/json" -d '{"id": 10,"name" :"TV","info": {"price": "70 $","weight": "2 kg"}}' "
http://localhost:8000/products"
{
  "id": 10,
  "name": "TV",
  "info": {
    "price": "70 $",
    "weight": "2 kg"
  }
}(base) macbookair@MacBook-Air-MacBook task_4 % curl -X GET http://localhost:8000/products
[
  {
    "id": 1,
    "name": "vacuum cleaner",
    "info": {
      "price": "80 $",
      "weight": "2.5 kg"
    }
  },
  {
    "id": 2,
    "name": "kettle",
    "info": {
      "price": "40 $",
      "weight": "1 kg"
    }
  },
  {
    "id": 10,
    "name": "TV",
    "info": {
      "price": "70 $",
      "weight": "2 kg"
    }
  }
]
```


GET
http://localhost:8000/products
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Query Params
KEY
VALUE
DESCRIPTION
Bulk
Body
Cookies
Headers (13)
Test Results
Status: 200 OK
Time: 194 ms
Size: 735 B
Pretty
Raw
Preview
Visualize
JSON
[
{
"id": 1,
"name": "vacuum cleaner",
"info": {
"price": "80 $",
"weight": "2.5 kg"
}
},
{
"id": 2,
"name": "kettle",
"info": {
"price": "40 $",
"weight": "1 kg"
}
},
{
"id": 10,
"name": "TV",
Cookies
Desktop Agent
Runner
Trash


GET
http://localhost:8001
Send
Params
Authorization
Headers (7)
Body
Pre-request Script
Tests
Settings
Cookies
Query Params
KEY
VALUE
DESCRIPTION
Bulk Edit
Body
Cookies
Headers (9)
Test Results
Status: 200 OK
Time: 389 ms
Size: 422 B
Save Response
Pretty
Raw
Preview
Visualize
HTML
<html>
<head>
<title>Task_2</title>
</head>
<body>
<img class="picture"
src="image.jpg"
alt="Charon">
</body>
</html>