

**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

The screenshot shows a terminal window with two tabs: 'docker-compose.yaml' and 'docker'. The 'docker-compose.yaml' tab contains the following YAML code:

```
version: "3.9"
services:
  hello_world:
    image: "hello-world"
```

The 'docker' tab shows the output of the command 'docker compose up'. It starts with '[+] Running 2/1' and lists two created services: 'docker_default' and 'docker-hello_world-1'. It then attaches to the 'docker-hello_world-1' container. Inside the container, it prints 'Hello from Docker!' followed by a message stating that the installation appears to be working correctly. It then details the four steps Docker took to reach this point. Finally, it provides instructions to run an Ubuntu container, share images, and visit the Docker documentation, ending with 'docker-hello_world-1 exited with code 0'.

```
(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  | Hello from Docker!
docker-hello_world-1  | This message shows that your installation appears to be working correctly.
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 x
Scroll Tabs version: "3.9"
1
2
3 services:
4   lite_server:
5     container_name: lite_server_name
6     build:
7       context: .
8       dockerfile: Dockerfile
9     ports:
10      - "8002:3000"
11     restart: "no"
```

```
◀ ▶ Dockerfile x
1 FROM node
2
3 WORKDIR /usr/src/app
4 COPY package*.json ./
5
6 RUN npm install
7 RUN npm install --global lite-server
8 COPY . .
9
10 EXPOSE 3000
11
12 CMD npm run start
```

```
◀ ▶ index.html x
1 <html>
2   <head>
3     <title>Task_2</title>
4   </head>
5   <body>
6     
9   </body>
10  </html>
```

```
◀ ▶ package.json x
1  {
2    "main": "index.html",
3    "devDependencies": {
4      "lite-server": "^2.2.0"
5    },
6    "scripts": {
7      "start": "lite-server"
8    }
9  }
10
```

```
[(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 them
[+] Running 2/2
  # Network task_2_default     Created
  # Container lite_server_name Created
```



task_3

The screenshot shows a code editor interface with two tabs open. On the left, the file tree shows a directory structure under 'docker' containing 'task_1', 'task_2', 'task_3' (which is expanded to show 'context' and 'db.json'), and 'task_4'. The 'docker-compose.yaml' file is selected in the tree. On the right, the code editor has two tabs: 'docker-compose.yaml' and 'Dockerfile'. The 'docker-compose.yaml' tab contains the following YAML code:

```
version: "3.9"
services:
  json-server:
    container_name: "json_server_name"
    build:
      context: .
    ports:
      - "8001:3000"
    restart: "no"
    volumes:
      - "./context:/context"
```

The 'Dockerfile' tab contains the following Dockerfile code:

```
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
```

This screenshot shows the same code editor interface as the previous one, but the 'Dockerfile' tab is the active one. The code editor shows the Dockerfile content from the previous screenshot.

This screenshot shows the code editor with three tabs open. The left sidebar shows the file tree with 'docker-compose.yaml' selected. The tabs at the top are 'docker-compose.yaml', 'db.json', and 'Dockerfile'. The 'db.json' tab is active and contains the following JSON data:

```
{
  "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"
  }
}
```

The 'docker-compose.yaml' tab shows the same configuration as the previous screenshots. The 'Dockerfile' tab shows the same Dockerfile content.

```

[(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 |          \{^_^}/ hi!
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

```

① <http://localhost:8000> ⌂ ⭐

JSON Server

❤ GitHub Sponsors

⌚ My JSON Server

Congrats!

You're successfully running JSON Server
✧*。✧(੭੮੯)*✧*

Resources

[/products](#) 2x
[/profile](#) object

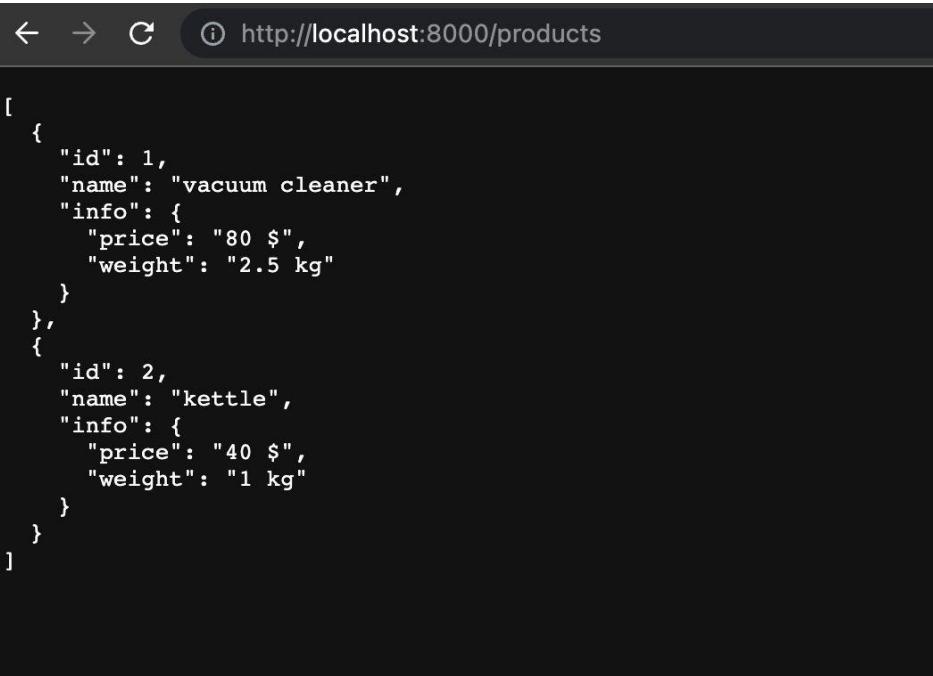
To access and modify resources, you can use any HTTP method:

[GET](#) [POST](#) [PUT](#) [PATCH](#) [DELETE](#) [OPTIONS](#)

undefined

Documentation

[README](#)



A screenshot of a web browser window displaying a JSON array of product data. The URL in the address bar is `http://localhost:8000/products`. The JSON response is as follows:

```
[  
  {  
    "id": 1,  
    "name": "vacuum cleaner",  
    "info": {  
      "price": "80 $",  
      "weight": "2.5 kg"  
    }  
  },  
  {  
    "id": 2,  
    "name": "kettle",  
    "info": {  
      "price": "40 $",  
      "weight": "1 kg"  
    }  
  }]
```

task_4

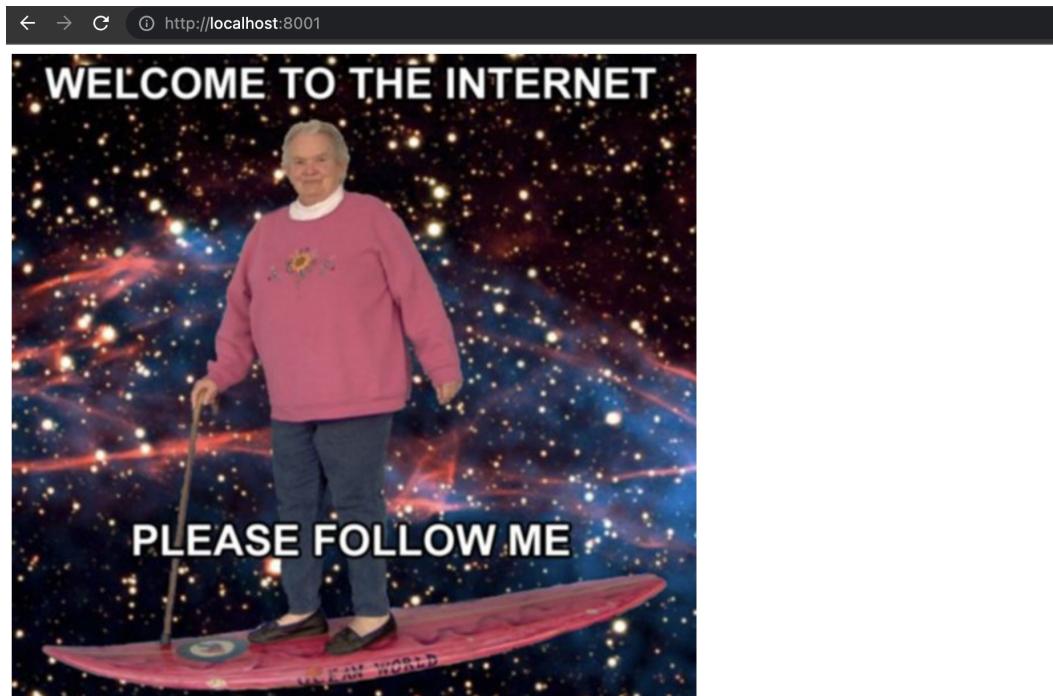
```
version: "3.9"
services:
  lite:
    build:
      context: ../task_2/
      volumes:
        - ../task_2:/node/app/src
    json:
      build:
        context: ../task_3/
        volumes:
          - ../task_3/context:/context
    nginx:
      build:
        context: .
      depends_on:
        - lite
      links:
        - lite
      ports:
        - 8000:80
        - 8001:81
```

```
FROM nginx:latest
COPY ./conf/nginx.conf ./etc/nginx/nginx.conf
```

```
worker_processes 1;
events {};
http {
  default_type application/octet-stream;
  server {
    # this server listens on port 80
    listen 80 default_server;
    listen ::1:80 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 ::1:81 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;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
      }
    }
}
```

① http://localhost:8000

JSON Server



```
(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

alexa@macbookair:~/Documents$ 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 different 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 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk
Body	Cookies	Headers (13)	Test Results	Status: 200 OK Time: 194 ms Size: 735 B Save Response

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 1,
4     "name": "vacuum cleaner",
5     "info": {
6       "price": "80 $",
7       "weight": "2.5 kg"
8     }
9   },
10  {
11    "id": 2,
12    "name": "kettle",
13    "info": {
14      "price": "40 $",
15      "weight": "1 kg"
16    }
17  },
18  {
19    "id": 10,
20    "name": "TV",
21  }
```

Cookies Desktop Agent Runner Trash

GET 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

```
1 <html>
2 |
3 <head>
4 |   <title>Task_2</title>
5 </head>
6
7 <body>
8 |   
11 </body>
12
13 </html>
```