Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Лабораторна робота №3

з дисципліни «Проектування розподілених систем»

за темою «Розгортання веб-порталу як Docker Container»

Виконав:

Студент групи ІА-11мн

Новиков Данило Михайлович

Перевірив:

доц. Волокина Артем Миколайович

Київ 2022

Тема: Розгортання веб-порталу як Docker Container. Розгортання застосунок за допомогою docker-compose на віртуальному сервері (новому або з Лаб. Работи 1)

Завдання:

- Застосунок має бути зібраний в docker image і розгорнутий локально;
- Опціонально: Розгорнути застосунок локально за допомогою docker-compose;
- Опублікувати docker image з застосунком в публічному Container Registry (Azure Container Registry, Digital Ocean Container Registry або GitLab Container Registry);
- Розгорнути віртуальний сервер і встановити docker engine на ньому;
- Розгорнути застосунок і всі допоміжні сервіси через docker-compose використовуючи публічний Container Registry.

Хід роботи

Репозиторій: https://github.com/JokerFunny/PRZ**.**

1. Створимо Dockerfile для запуску нашого додатку:

Dockerfile > ...

```
1   FROM mcr.microsoft.com/dotnet/sdk:6.0-alpine as build
2   WORKDIR /src
3   COPY JotterAPI/ .
4   RUN dotnet restore JotterAPI/JotterAPI.csproj
5   RUN dotnet publish JotterAPI/JotterAPI.csproj -c Release -o output
6
7   FROM mcr.microsoft.com/dotnet/aspnet:6.0-alpine as runtime
8   WORKDIR /output
9   COPY --from=build /src/output .
10  ENTRYPOINT [ "dotnet", "JotterAPI.dll" ]
```

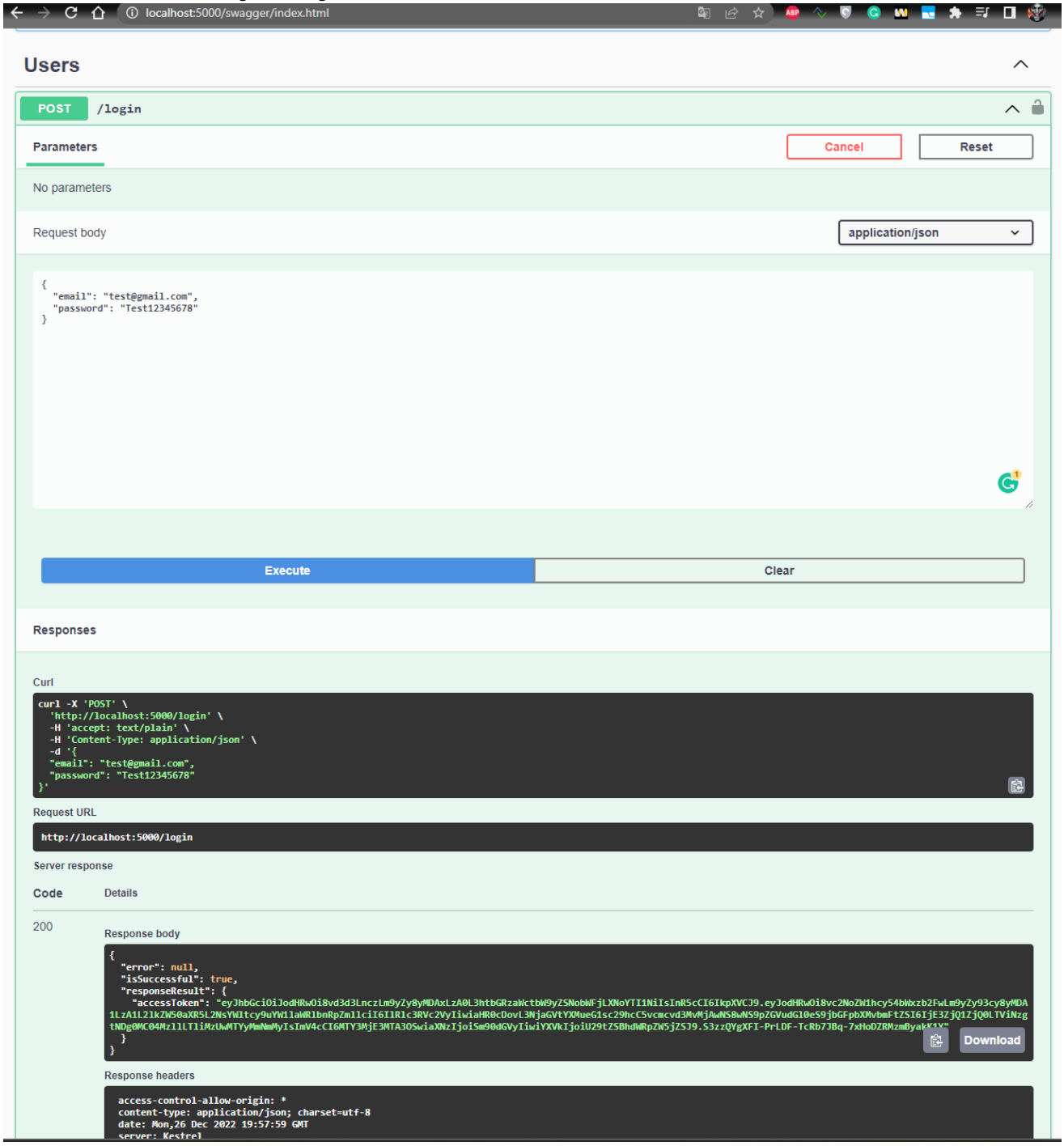
```
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> docker build -t jotter .
[+] Building 0.2s (2/2) FINISHED
 => [internal] load build definition from Dockerfile                                                                                                0.2s
 => => transferring dockerfile: 2B                                                                                                                  0.0s
[+] Building 57.0s (14/14) FINISHED
 => [internal] load build definition from Dockerfile                                                                                                0.1s
 => => transferring dockerfile: 395B                                                                                                                0.0s
 => [internal] load .dockerignore                                                                                                                   0.1s
 => => transferring context: 2B                                                                                                                     0.0s
 => [internal] load metadata for mcr.microsoft.com/dotnet/aspnet:6.0-alpine                                                                         1.3s
 => [internal] load metadata for mcr.microsoft.com/dotnet/sdk:6.0-alpine                                                                            1.2s
 => [build 1/5] FROM mcr.microsoft.com/dotnet/sdk:6.0-alpine@sha256:b02b08b6ae0c2054a856ba5f3c0db38bbe0971c433f059b962dc81734                     32.0s
 => => resolve mcr.microsoft.com/dotnet/sdk:6.0-alpine@sha256:b02b08b6ae0c2054a856ba5f3c0db38bbe0971c433f059b962dc8173493b9053                      0.1s
 => => sha256:13050f9ab7043edf3fc9c9779eca0299d4791d5cd98946741195a3db5a3f5450 1.80kB / 1.80kB                                                      0.0s
 => => sha256:6b34f490fdab37e4a7886ad068578739227d686dad528846ed11c459835c36b1 31.18MB / 31.18MB                                                   12.4s
 => => sha256:b02b08b6ae0c2054a856ba5f3c0db38bbe0971c433f059b962dc8173493b9053 1.01kB / 1.01kB                                                      0.0s
 => => sha256:112ce6d82e454a32f944ede46161efb8947d292579505ebb217fa8a16d886faf 7.97kB / 7.97kB                                                      0.0s
 => => sha256:8429b38ed18866a42ac4014d84b642235cf14109ca9f4fe2d19e3df0d82308f4 1.85MB / 1.85MB                                                      2.1s
 => => sha256:ca7dd9ec2225f2385955c43b2379305acd51543c28cf1d4e94522b3d94cce3ce 2.81MB / 2.81MB                                                      1.5s
 => => extracting sha256:ca7dd9ec2225f2385955c43b2379305acd51543c28cf1d4e94522b3d94cce3ce                                                           0.1s
 => => sha256:7f5885fc72db5577edb28c7d2b5e426e721f95cd97ab6066939581e3bb80f08e 9.46MB / 9.46MB                                                      4.2s
 => => extracting sha256:8429b38ed18866a42ac4014d84b642235cf14109ca9f4fe2d19e3df0d82308f4                                                           0.1s
 => => sha256:f2bf015585b1b6146ad5a54f8824ebd2f50d545c7d767c98b913e83ea1145065 21.52MB / 21.52MB                                                   14.6s
 => => sha256:1c283e3069c4959597f1967bf37ddad84edf96eec12626110b2f76fccc4bec1e 148.01MB / 148.01MB                                                 29.7s
 => => sha256:f90a04da14675d019b42564e8a4c7390c7afb7888525e2edd46a7f9052c7a8ec 12.93MB / 12.93MB                                                   17.5s
 => => extracting sha256:6b34f490fdab37e4a7886ad068578739227d686dad528846ed11c459835c36b1                                                          0.2s
 => => extracting sha256:7f5885fc72db5577edb28c7d2b5e426e721f95cd97ab6066939581e3bb80f08e                                                           0.1s
 => => extracting sha256:f2bf015585b1b6146ad5a54f8824ebd2f50d545c7d767c98b913e83ea1145065                                                           0.2s
 => => extracting sha256:1c283e3069c4959597f1967bf37ddad84edf96eec12626110b2f76fccc4bec1e                                                           1.7s
 => => extracting sha256:f90a04da14675d019b42564e8a4c7390c7afb7888525e2edd46a7f9052c7a8ec                                                           0.2s
 => [internal] load build context                                                                                                                   7.9s
 => => transferring context: 549.71MB                                                                                                               7.8s
 => [runtime 1/3] FROM mcr.microsoft.com/dotnet/aspnet:6.0-alpine@sha256:6eb8fd422895c3d2a9fcb9d94d81eb4833426f39eb2ded7034bc                      13.3s
 => => resolve mcr.microsoft.com/dotnet/aspnet:6.0-alpine@sha256:6eb8fd422895c3d2a9fcb9d94d81eb4833426f39eb2ded7034bce16d5a2da                      0.1s
 => => sha256:6eb8fd422895c3d2a9fcb9d94d81eb4833426f39eb2ded7034bce16d5a2da902 1.01kB / 1.01kB                                                      0.0s
 => => sha256:2d6f9c287a02046220cd56ed14f1aabf204d6a53c589803e49f8e48d83542563 1.16kB / 1.16kB                                                      0.0s
 => => sha256:12a9b25d94506dfb9e524a61e2b3ae730da6d52d5ad2f16f225e4aec24162d8e 4.45kB / 4.45kB                                                      0.0s
 => => sha256:ca7dd9ec2225f2385955c43b2379305acd51543c28cf1d4e94522b3d94cce3ce 2.81MB / 2.81MB                                                      1.5s
 => => sha256:6b34f490fdab37e4a7886ad068578739227d686dad528846ed11c459835c36b1 31.18MB / 31.18MB                                                   12.4s
 => => sha256:8429b38ed18866a42ac4014d84b642235cf14109ca9f4fe2d19e3df0d82308f4 1.85MB / 1.85MB                                                      2.1s
 => => sha256:7f5885fc72db5577edb28c7d2b5e426e721f95cd97ab6066939581e3bb80f08e 9.46MB / 9.46MB                                                      4.2s
 => => extracting sha256:ca7dd9ec2225f2385955c43b2379305acd51543c28cf1d4e94522b3d94cce3ce                                                           0.1s
 => => extracting sha256:8429b38ed18866a42ac4014d84b642235cf14109ca9f4fe2d19e3df0d82308f4                                                          53.3s
 => => extracting sha256:6b34f490fdab37e4a7886ad068578739227d686dad528846ed11c459835c36b1                                                          0.2s
 => => extracting sha256:7f5885fc72db5577edb28c7d2b5e426e721f95cd97ab6066939581e3bb80f08e                                                           0.1s
 => [runtime 2/3] WORKDIR /output                                                                                                                   0.8s
 => [build 2/5] WORKDIR /src                                                                                                                        0.7s
 => [build 3/5] COPY JotterAPI/ .                                                                                                                   0.9s
 => [build 4/5] RUN dotnet restore JotterAPI/JotterAPI.csproj                                                                                      16.8s
 => [build 5/5] RUN dotnet publish JotterAPI/JotterAPI.csproj -c Release -o output                                                                  4.3s
 => [runtime 3/3] COPY --from=build /src/output .                                                                                                   0.2s
 => exporting to image                                                                                                                              0.3s
 => => exporting layers                                                                                                                             0.2s
 => => writing image sha256:05d489b72631bd2b5ff9306df0d682ea434cb91e5ab70950cf378364a4a138c1                                                        0.0s
 => => naming to docker.io/library/jotter                                                                                                           0.0s

Use 'docker scan' to run Snyk tests against images to find vulnerabilities and learn how to fix them
```

2. docker build -t jotter .

```
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> docker run -e 'ConnectionStrings__JotterDbContext=User ID=pgadmin;Password=12345678;Host=lab1-db.cl5mp5qdgoug.eu-central-1.rds.amazonaws.com;Port=5432;Database=jotter;Pooling=true;' -p 5000:5000 --name jotter jotter
warn: Microsoft.AspNetCore.DataProtection.Repositories.FileSystemXmlRepository[60]
      Storing keys in a directory '/root/.aspnet/DataProtection-Keys' that may not be persisted outside of the container. Protected data will be unavailable when container is destroyed.
warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
      No XML encryptor configured. Key {c6fcbe53-1998-41b5-9033-637787736802} may be persisted to storage in unencrypted form.
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://0.0.0.0:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /output
```

3. Перевіримо працездатність:

docker run -e 'ConnectionStrings__JotterDbContext=User ID=pgadmin;Password=12345678;Host=lab1-db.cl5mp5qdgoug.eu-central-1.rds.amazonaws.com;Port=5432;Database=jotter;Pooling=true;' -p 5000:5000 --name jotter jotter



4. Створимо dockecr-compose

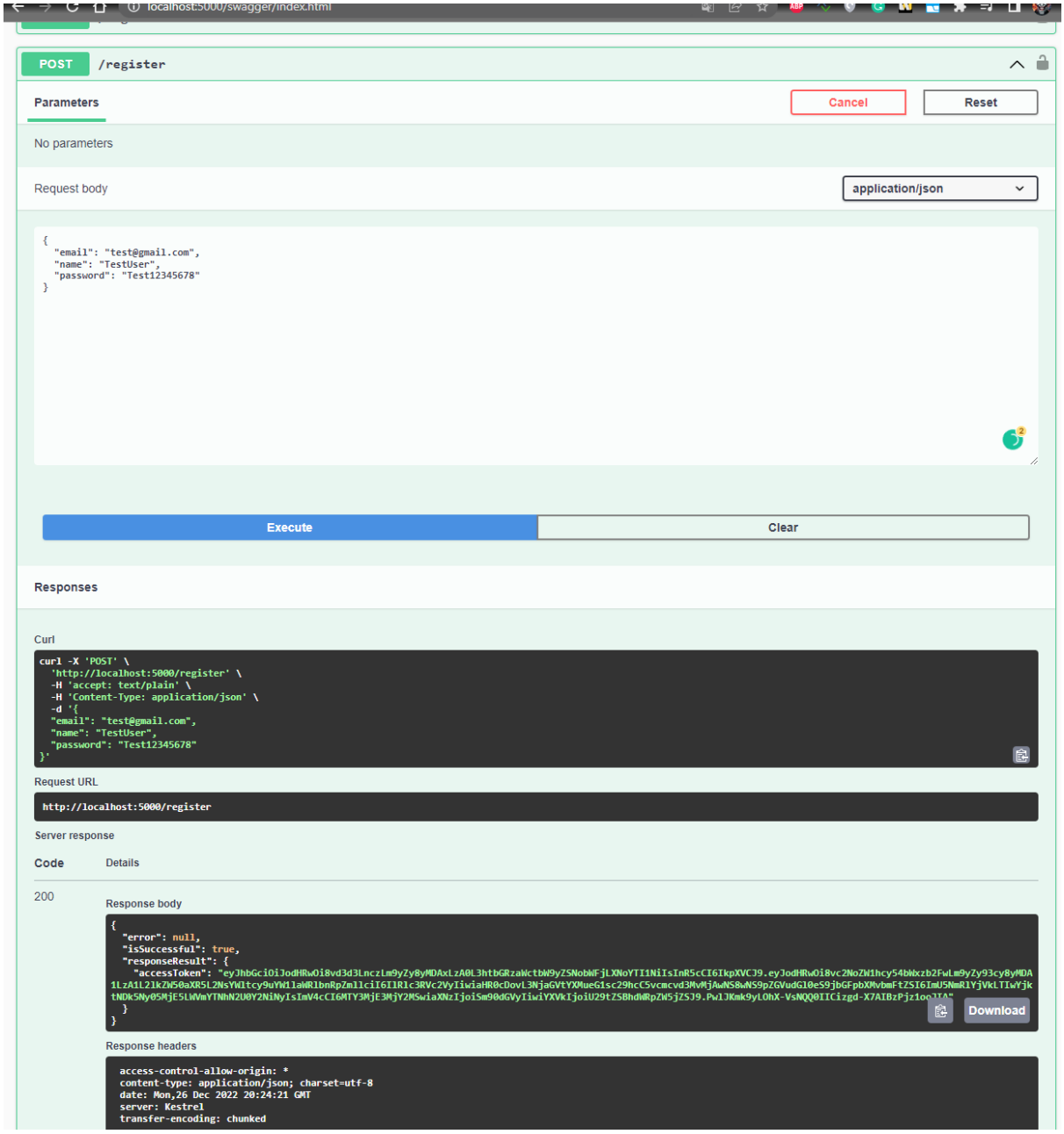
docker-compose.yml > {} services > {} jotterapi > [ ] env_file

```yaml
version: '3.4'

services:
  postgres:
    image: postgres:13.7
    environment:
      POSTGRES_USER: "pgadmin"
      POSTGRES_PASSWORD: "pgadmin"
    volumes:
      - dbdata:/var/lib/postgresql/data
    ports:
      - "5432:5432"

  jotterapi:
    image: jotterapi
    container_name: jotterapi
    build: .
    depends_on:
      - postgres
    ports:
      - '5000:5000'
    env_file:
      - ./jotterapi.env
    environment:
      WAIT_HOSTS: postgres:5432

volumes:
  dbdata:
    driver: local
```

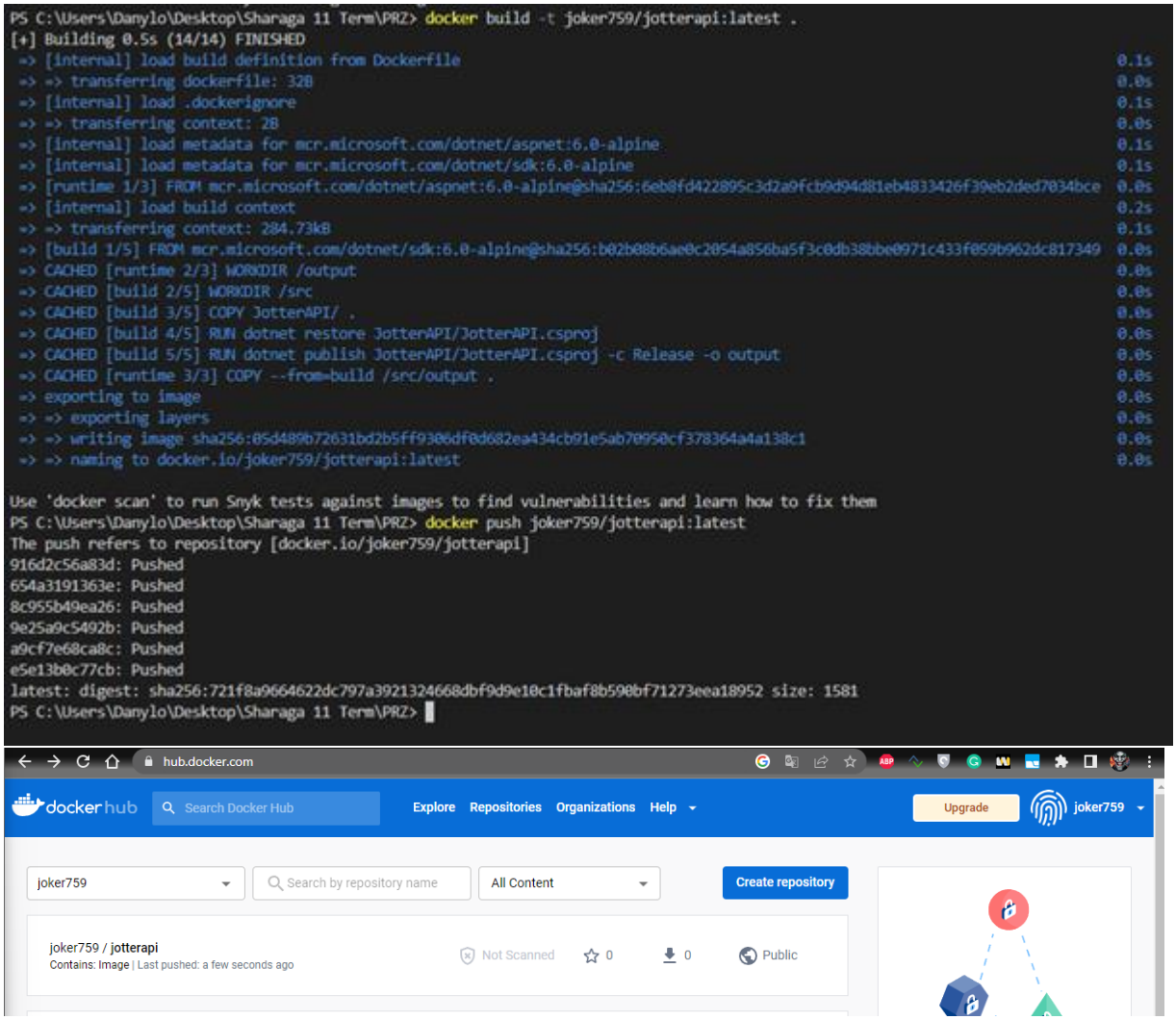
5. Перевіримо працездатність додатку:

docker-compose up

```
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> docker-compose up
[+] Running 2/0
 - Container prz-postgres-1  Created                                                                                0.0s
 - Container jotterapi       Created                                                                                0.0s
Attaching to jotterapi, prz-postgres-1
prz-postgres-1  |
prz-postgres-1  | PostgreSQL Database directory appears to contain a database; Skipping initialization
prz-postgres-1  |
prz-postgres-1  | 2022-12-26 20:29:31.588 UTC [1] LOG:  starting PostgreSQL 13.7 (Debian 13.7-1.pgdg110+1) on x86_64-pc-linux-gnu, co
mpiled by gcc (Debian 10.2.1-6) 10.2.1 20210110, 64-bit
prz-postgres-1  | 2022-12-26 20:29:31.588 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
prz-postgres-1  | 2022-12-26 20:29:31.588 UTC [1] LOG:  listening on IPv6 address "::", port 5432
prz-postgres-1  | 2022-12-26 20:29:31.614 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
prz-postgres-1  | 2022-12-26 20:29:31.640 UTC [26] LOG:  database system was shut down at 2022-12-26 20:29:28 UTC
prz-postgres-1  | 2022-12-26 20:29:31.648 UTC [1] LOG:  database system is ready to accept connections
jotterapi       | warn: Microsoft.AspNetCore.DataProtection.Repositories.FileSystemXmlRepository[60]
jotterapi       |       Storing keys in a directory '/root/.aspnet/DataProtection-Keys' that may not be persisted outside of the cont
ainer. Protected data will be unavailable when container is destroyed.
jotterapi       | info: Microsoft.Hosting.Lifetime[14]
jotterapi       |       Now listening on: http://0.0.0.0:5000
jotterapi       | info: Microsoft.Hosting.Lifetime[0]
jotterapi       |       Application started. Press Ctrl+C to shut down.
jotterapi       | info: Microsoft.Hosting.Lifetime[0]
jotterapi       |       Hosting environment: Production
jotterapi       | info: Microsoft.Hosting.Lifetime[0]
jotterapi       |       Content root path: /output
```

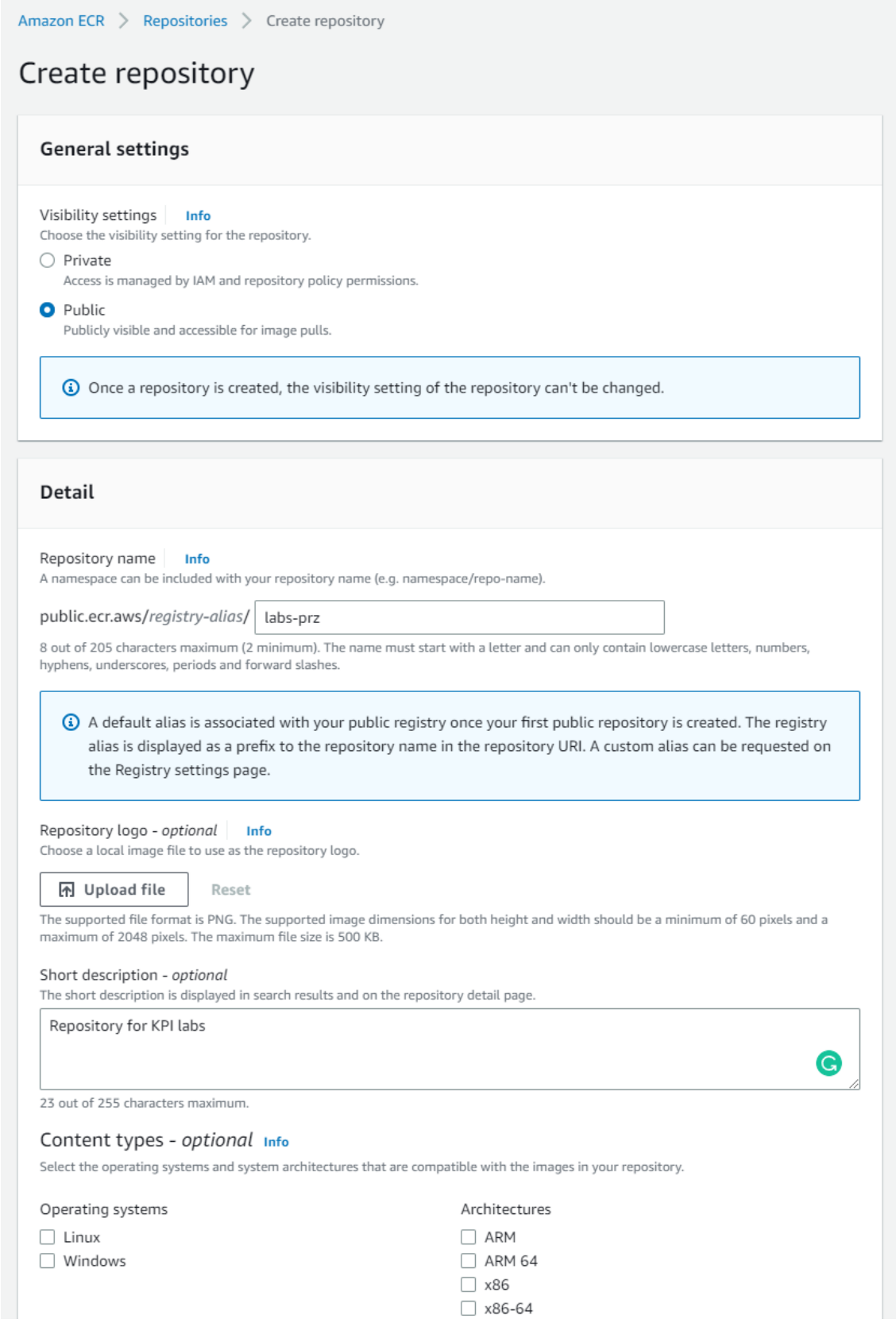


6. Опублікуємо docker image на «DockerHub»:



Але, так як цього варіанту немає, розглянемо через AWS.

7. Створимо публічний репозиторій використовуючи «Amazon Elastic Container Registry»:



8. Створимо юзера для роботи з «Amazon Elastic Container Registry»:

Add user

1 2 3 4 5

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* ECS-FullAccess

Add another user

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. Learn more

Select AWS credential type* ☑ **Access key - Programmatic access**
Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.

☐ **Password - AWS Management Console access**
Enables a **password** that allows users to sign-in to the AWS Management Console.

Add user

1 2 3 4 5

Set permissions

Add user to group | Copy permissions from existing user | Attach existing policies directly

Create policy

Filter policies: AmazonECS_Full — Showing 1 result

	Policy name	Type	Used as
☑	AmazonECS_FullAccess	AWS managed	None

Set permissions boundary

Add user

1 2 3 4 5

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	ECS-FullAccess
AWS access type	Programmatic access - with an access key
Permissions boundary	Permissions boundary is not set

Permissions summary

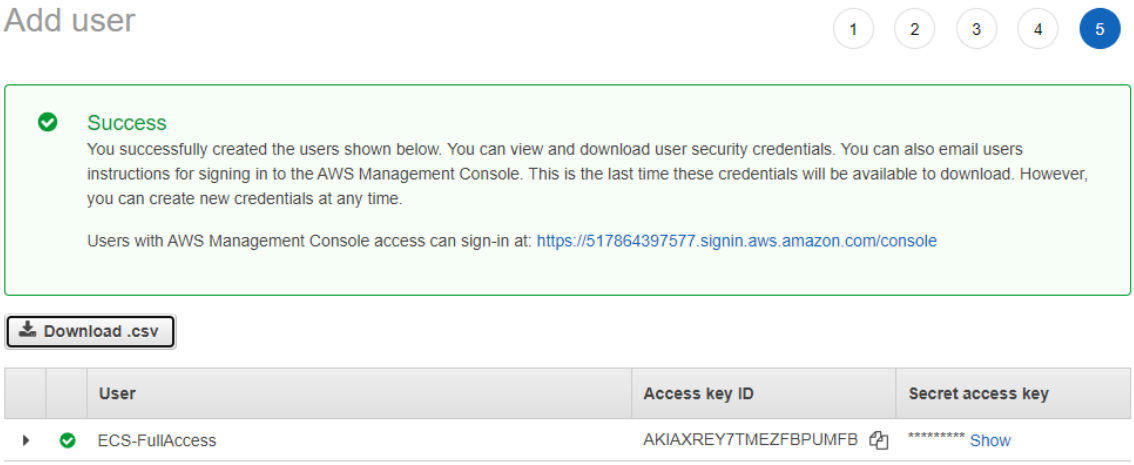
The following policies will be attached to the user shown above.

Type	Name
Managed policy	AmazonECS_FullAccess

Tags

The new user will receive the following tag

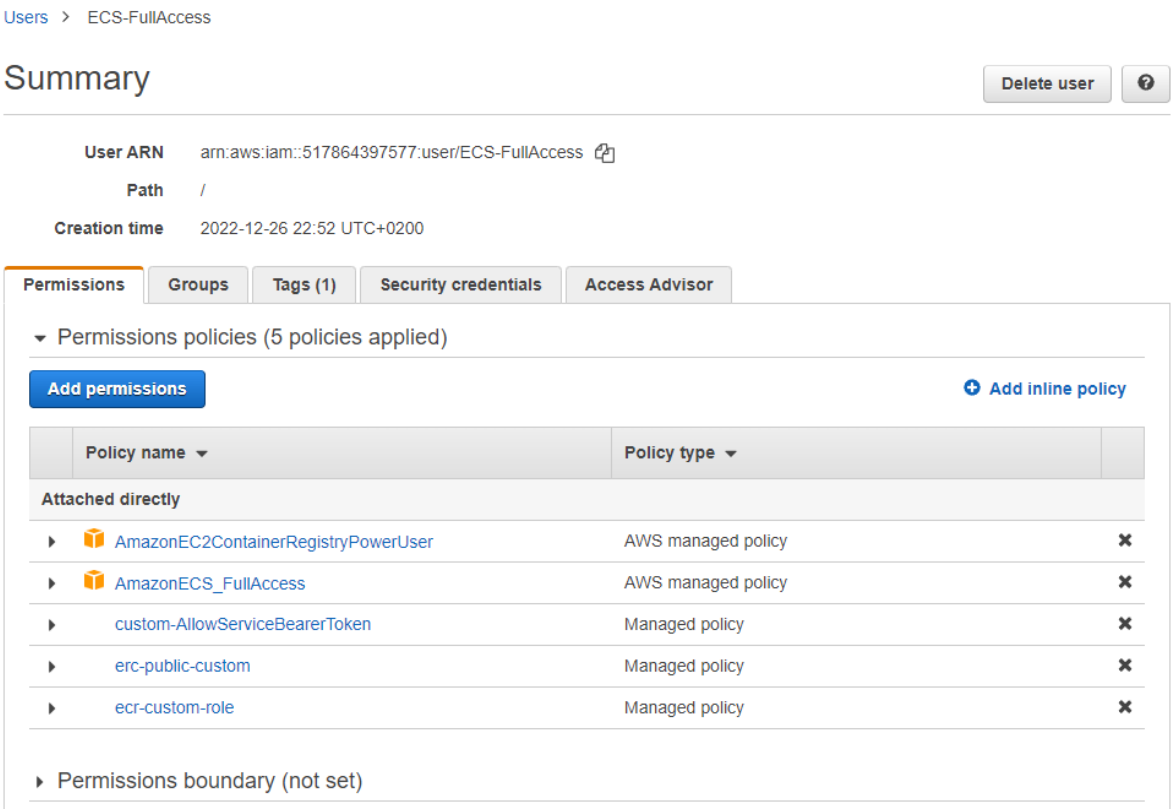
Key	Value
purpose	labs



9. Встановимо і налаштуємо **AWSCLI** для роботи з AWS:

```
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> aws --version
aws-cli/2.9.10 Python/3.9.11 Windows/10 exe/AMD64 prompt/off
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> aws configure
AWS Access Key ID [None]: AKIAXREY7TMEZFBPUMFB
AWS Secret Access Key [None]: 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
Default region name [None]:
Default output format [None]:
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> aws configure list
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                <not set>             None    None
access_key     ****************UMFB shared-credentials-file
secret_key     ****************N49B shared-credentials-file
    region                <not set>             None    None
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> aws ecr get-login-password

You must specify a region. You can also configure your region by running "aws configure".
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> aws configure
AWS Access Key ID [****************UMFB]:
AWS Secret Access Key [****************N49B]:
Default region name [None]: eu-central-1
Default output format [None]:
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> aws configure list
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                <not set>             None    None
access_key     ****************UMFB shared-credentials-file
secret_key     ****************N49B shared-credentials-file
    region                eu-central-1      config-file    ~/.aws/config
```

10.Модифікуємо створеного юзера, додаймо додаткові політики безпеки:

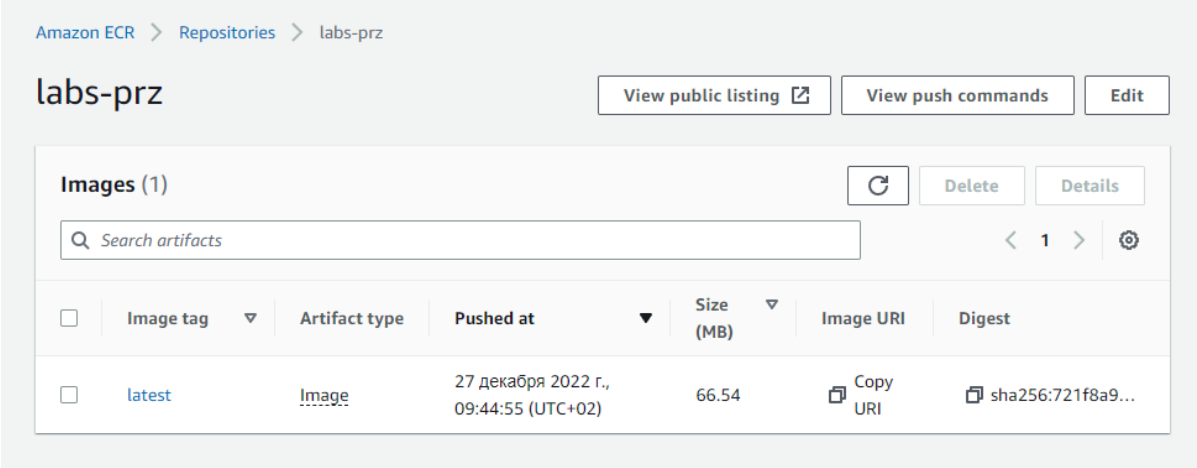


11. Логінемося в AWS ECR використовуючи AWSCLI, пушимо докер образ:

```
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> aws ecr-public get-login-password --region us-east-1 | docker login
--username AWS --password-stdin public.ecr.aws
Login Succeeded

Logging in with your password grants your terminal complete access to your account.
For better security, log in with a limited-privilege personal access token. Learn more at https://docs.docker.com/go
/access-tokens/
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> docker images --filter reference=jotterapi
REPOSITORY   TAG       IMAGE ID       CREATED        SIZE
jotterapi    latest    05d489b72631   12 hours ago   162MB
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> docker tag jotterapi:latest public.ecr.aws/s9e9c2b3/labs-prz
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> docker push public.ecr.aws/s9e9c2b3/labs-prz
Using default tag: latest
The push refers to repository [public.ecr.aws/s9e9c2b3/labs-prz]
916d2c56a83d: Pushed
654a3191363e: Pushed
8c955b49ea26: Pushed
9e25a9c5492b: Pushed
a9cf7e68ca8c: Pushed
e5e13b0c77cb: Pushed
latest: digest: sha256:721f8a9664622dc797a3921324668dbf9d9e10c1fbaf8b590bf71273eea18952 size: 1581
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ>
```

```
PS C:\Users\Danylo\Desktop\Sharaga 11 Term\PRZ> docker pull public.ecr.aws/s9e9c2b3/labs-prz:latest
latest: Pulling from s9e9c2b3/labs-prz
Digest: sha256:721f8a9664622dc797a3921324668dbf9d9e10c1fbaf8b590bf71273eea18952
Status: Image is up to date for public.ecr.aws/s9e9c2b3/labs-prz:latest
public.ecr.aws/s9e9c2b3/labs-prz:latest
```

12. Перевіряємо, що образ запушено:



13. Перейдемо на віртуальний сервер, створений в ЛР1. Нам треба встановити та налаштувати docker:

```
ubuntu@ip-172-31-22-246:~$ curl -fsSL https://get.docker.com -o get-docker.sh
ubuntu@ip-172-31-22-246:~$ sudo sh get-docker.sh
# Executing docker install script, commit: 4f282167c425347a931ccfd95cc91fab041d4
14f
+ sh -c apt-get update -qq >/dev/null
+ sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq apt-transport-http
s ca-certificates curl >/dev/null
+ sh -c mkdir -p /etc/apt/keyrings && chmod -R 0755 /etc/apt/keyrings
+ sh -c curl -fsSL "https://download.docker.com/linux/ubuntu/gpg" | gpg --dearmo
r --yes -o /etc/apt/keyrings/docker.gpg
+ sh -c chmod a+r /etc/apt/keyrings/docker.gpg
+ sh -c echo "deb [arch=amd64 signed-by=/etc/apt/keyrings/docker.gpg] https://do
wnload.docker.com/linux/ubuntu jammy stable" > /etc/apt/sources.list.d/docker.li
st
+ sh -c apt-get update -qq >/dev/null
+ sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq --no-install-recom
mends docker-ce docker-ce-cli containerd.io docker-compose-plugin docker-scan-pl
ugin >/dev/null
+ version_gte 20.10
+ [ -z  ]
+ return 0
+ sh -c DEBIAN_FRONTEND=noninteractive apt-get install -y -qq docker-ce-rootless
-extras >/dev/null
+ sh -c docker version
Client: Docker Engine - Community
 Version:           20.10.22
 API version:       1.41
 Go version:        go1.18.9
 Git commit:        3a2c30b
 Built:             Thu Dec 15 22:28:04 2022
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true

Server: Docker Engine - Community
 Engine:
  Version:          20.10.22
  API version:      1.41 (minimum version 1.12)
  Go version:       go1.18.9
  Git commit:       42c8b31
  Built:            Thu Dec 15 22:25:49 2022
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.6.14
  GitCommit:        9ba4b250366a5ddde94bb7c9d1def331423aa323
 runc:
  Version:          1.1.4
  GitCommit:        v1.1.4-0-g5fd4c4d
 docker-init:
  Version:          0.19.0
  GitCommit:        de40ad0

================================================================================

To run Docker as a non-privileged user, consider setting up the
Docker daemon in rootless mode for your user:

    dockerd-rootless-setuptool.sh install

Visit https://docs.docker.com/go/rootless/ to learn about rootless mode.


To run the Docker daemon as a fully privileged service, but granting non-root
users access, refer to https://docs.docker.com/go/daemon-access/

WARNING: Access to the remote API on a privileged Docker daemon is equivalent
         to root access on the host. Refer to the 'Docker daemon attack surface'
         documentation for details: https://docs.docker.com/go/attack-surface/

================================================================================

ubuntu@ip-172-31-22-246:~$
```

```
ubuntu@ip-172-31-22-246:~$ docker version
Client: Docker Engine - Community
 Version:           20.10.22
 API version:       1.41
 Go version:        go1.18.9
 Git commit:        3a2c30b
 Built:             Thu Dec 15 22:28:04 2022
 OS/Arch:           linux/amd64
 Context:           default
 Experimental:      true
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Get "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/versio
n": dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-22-246:~$
```

14. Створимо папку для роботи з ЛР3:

```
ubuntu@ip-172-31-22-246:~$ ls
get-docker.sh  labs  test  test.1  test.2
ubuntu@ip-172-31-22-246:~$ cd labs
ubuntu@ip-172-31-22-246:~/labs$ ls
lab1
ubuntu@ip-172-31-22-246:~/labs$ mkdir lab3
ubuntu@ip-172-31-22-246:~/labs$ ls
lab1  lab3
ubuntu@ip-172-31-22-246:~/labs$ cd lab3
ubuntu@ip-172-31-22-246:~/labs/lab3$
```

15. Встановимо AWSCLI на віртуальну машину:

```
 52  curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
 53  unzip awscliv2.zip
 54  sudo apt install unzip
 55  unzip awscliv2.zip
 56  sudo ./aws/install~
 57  ls
 58  sudo ./aws/install
 59  /usr/local/bin/aws --version
```

```
ubuntu@ip-172-31-22-246:~$ ^C
ubuntu@ip-172-31-22-246:~$ /usr/local/bin/aws --version
aws-cli/2.9.10 Python/3.9.11 Linux/5.15.0-1026-aws exe/x86_64.ubuntu.22 prompt/off
```

16. Налаштовуємо локального користувача, логінемося:

```
ubuntu@ip-172-31-22-246:~/labs/lab3$ aws configure
AWS Access Key ID [None]: AKIAXREY7TMEZFBPUMFB
AWS Secret Access Key [None]: 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
Default region name [None]: eu-central-1
Default output format [None]:
ubuntu@ip-172-31-22-246:~/labs/lab3$ aws configure list
      Name                    Value             Type    Location
      ----                    -----             ----    --------
   profile                <not set>             None    None
access_key     ****************UMFB shared-credentials-file
secret_key     ****************N49B shared-credentials-file
    region                eu-central-1      config-file    ~/.aws/config
ubuntu@ip-172-31-22-246:~/labs/lab3$ aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post "http://%2Fvar%2Frun%2Fdocker.sock/v1.24/auth"
: dial unix /var/run/docker.sock: connect: permission denied
ubuntu@ip-172-31-22-246:~/labs/lab3$ ^C
ubuntu@ip-172-31-22-246:~/labs/lab3$ sudo chmod 666 /var/run/docker.sock
ubuntu@ip-172-31-22-246:~/labs/lab3$ aws ecr-public get-login-password --region us-east-1 | docker login --username AWS --password-stdin public.ecr.aws
WARNING! Your password will be stored unencrypted in /home/ubuntu/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
ubuntu@ip-172-31-22-246:~/labs/lab3$
```

17. Завантажемо наш docker image з AWS ECR на віртуальну машину:

```
ubuntu@ip-172-31-22-246:~/labs/lab3$ docker images
REPOSITORY   TAG       IMAGE ID   CREATED   SIZE
ubuntu@ip-172-31-22-246:~/labs/lab3$ docker pull public.ecr.aws/s9e9c2b3/labs-prz:latest
latest: Pulling from s9e9c2b3/labs-prz
ca7dd9ec2225: Pull complete
8429b38ed188: Pull complete
6b34f490fdab: Pull complete
7f5885fc72db: Pull complete
5bc1b6913b24: Pull complete
31550759ed2f: Pull complete
Digest: sha256:721f8a9664622dc797a3921324668dbf9d9e10c1fbaf8b590bf71273eea18952
Status: Downloaded newer image for public.ecr.aws/s9e9c2b3/labs-prz:latest
public.ecr.aws/s9e9c2b3/labs-prz:latest
ubuntu@ip-172-31-22-246:~/labs/lab3$ docker images
REPOSITORY                          TAG       IMAGE ID       CREATED        SIZE
public.ecr.aws/s9e9c2b3/labs-prz    latest    05d489b72631   13 hours ago   162MB
ubuntu@ip-172-31-22-246:~/labs/lab3$
```

18. Скопіюємо docker-compose, зробимо відповідні зміни:

```
ubuntu@ip-172-31-22-246:~/labs/lab3$ ls
docker-compose.yml  jotterapi.env
```

docker-compose-vm.yml > {} services > {} jotterapi

```yaml
version: '3.4'

services:
  postgres:
    image: postgres:13.7
    environment:
      POSTGRES_USER: "pgadmin"
      POSTGRES_PASSWORD: "pgadmin"
    volumes:
      - dbdata:/var/lib/postgresql/data
    ports:
      - "5432:5432"

  jotterapi:
    image: public.ecr.aws/s9e9c2b3/labs-prz:latest
    container_name: jotterapi
    depends_on:
      - postgres
    ports:
    - '5000:5000'
    env_file:
      - ./jotterapi.env
    environment:
      WAIT_HOSTS: postgres:5432

volumes:
  dbdata:
    driver: local
```

jotterapi.env

```
ConnectionStrings__JotterDbContext=User ID=pgadmin;Password=pgadmin;Host=postgres;Port=5432;Database=jotter;Pooling=true;
```

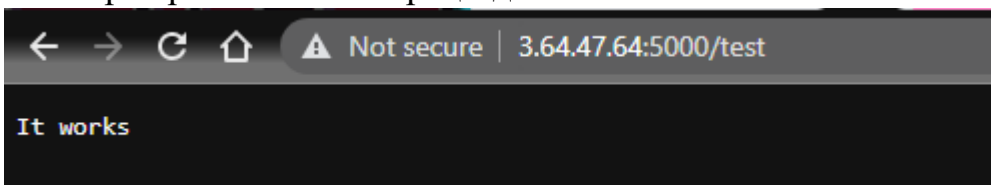
19. Встановимо docker-compose:

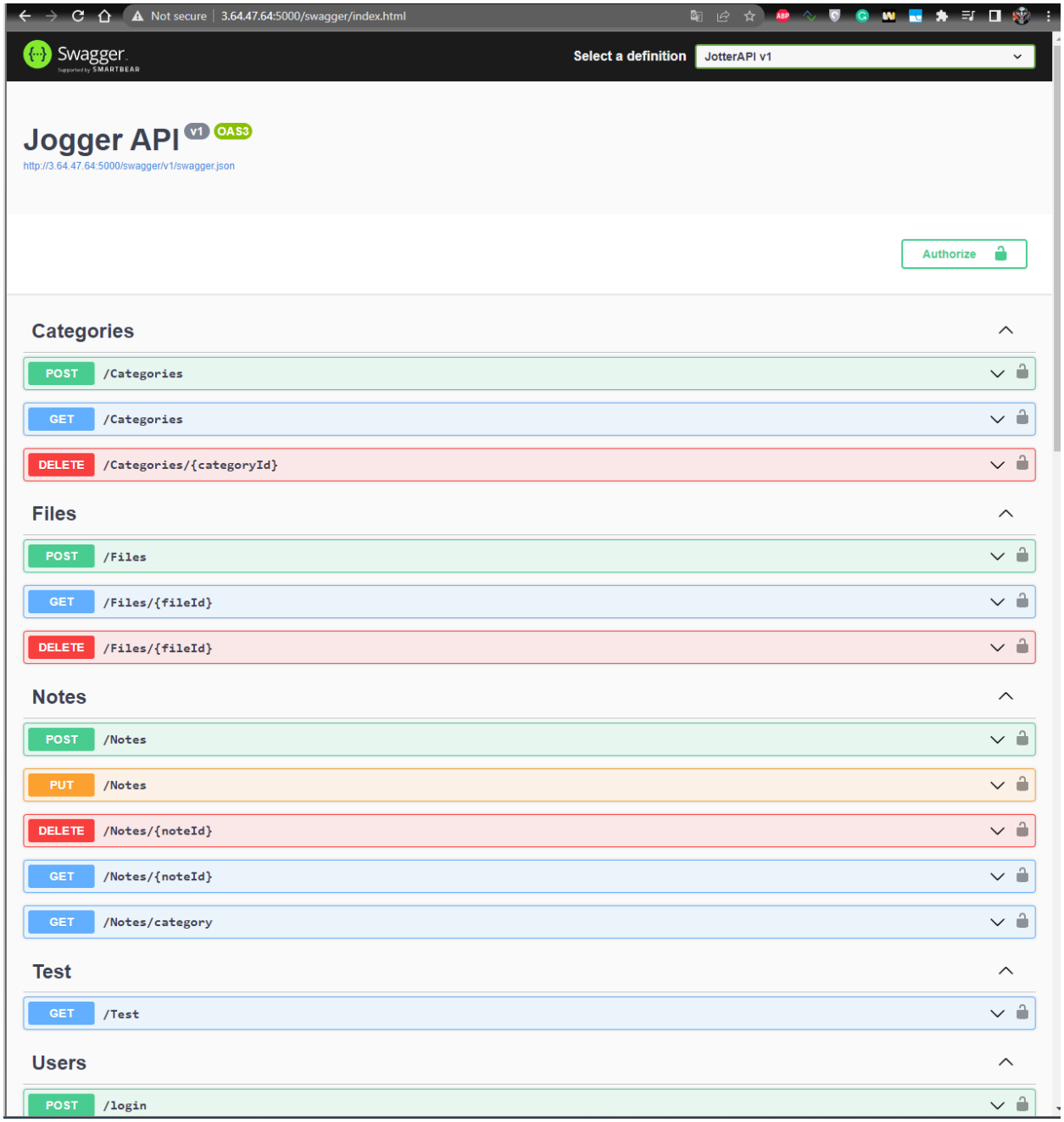
```
ubuntu@ip-172-31-22-246:~/labs/lab3$ docker-compose up
Command 'docker-compose' not found, but can be installed with:
sudo snap install docker          # version 20.10.17, or
sudo apt  install docker-compose  # version 1.29.2-1
See 'snap info docker' for additional versions.
ubuntu@ip-172-31-22-246:~/labs/lab3$ ^C
ubuntu@ip-172-31-22-246:~/labs/lab3$ sudo apt  install docker-compose
Reading package lists... Done
```

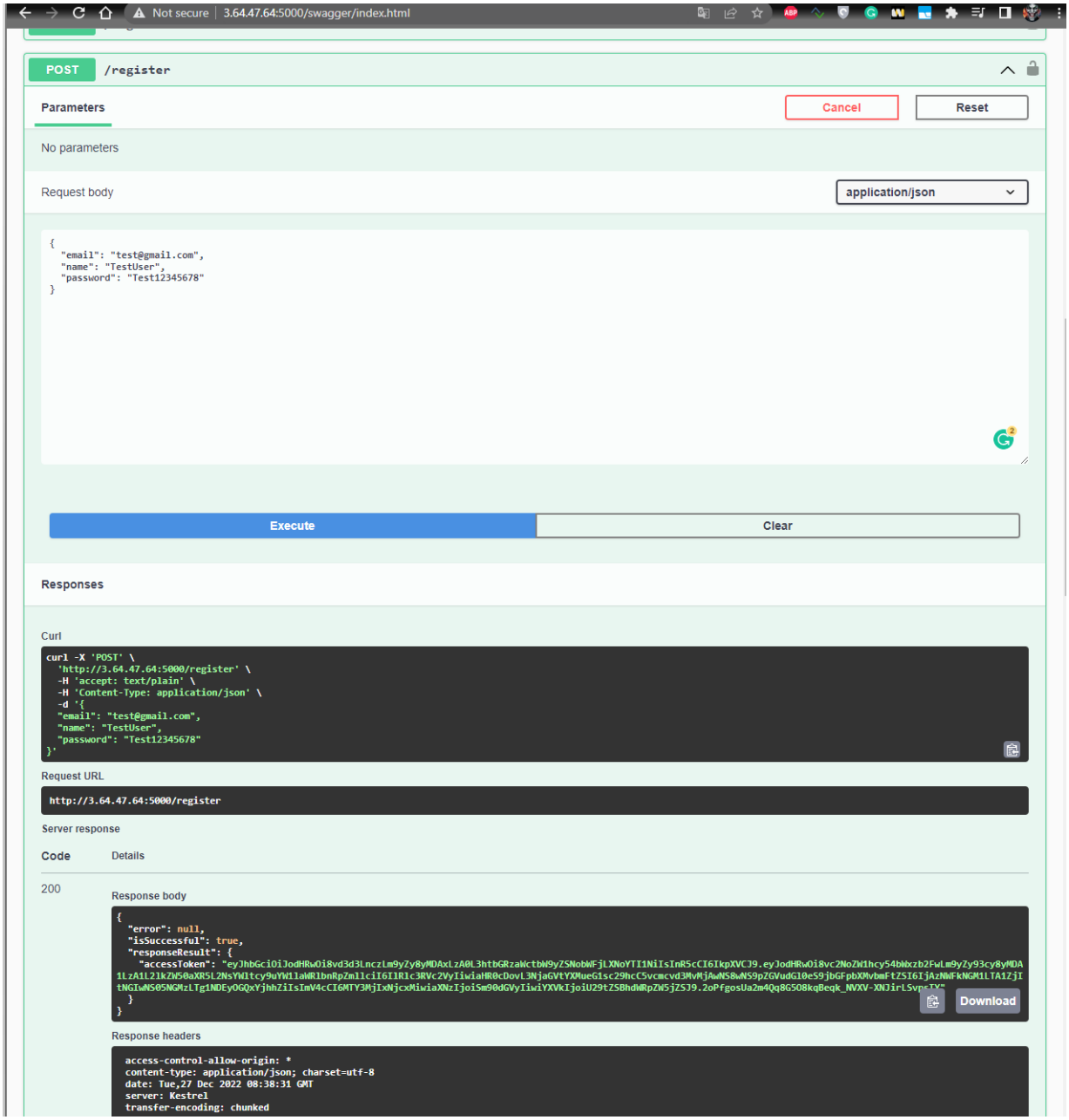
20. Запустимо додаток:

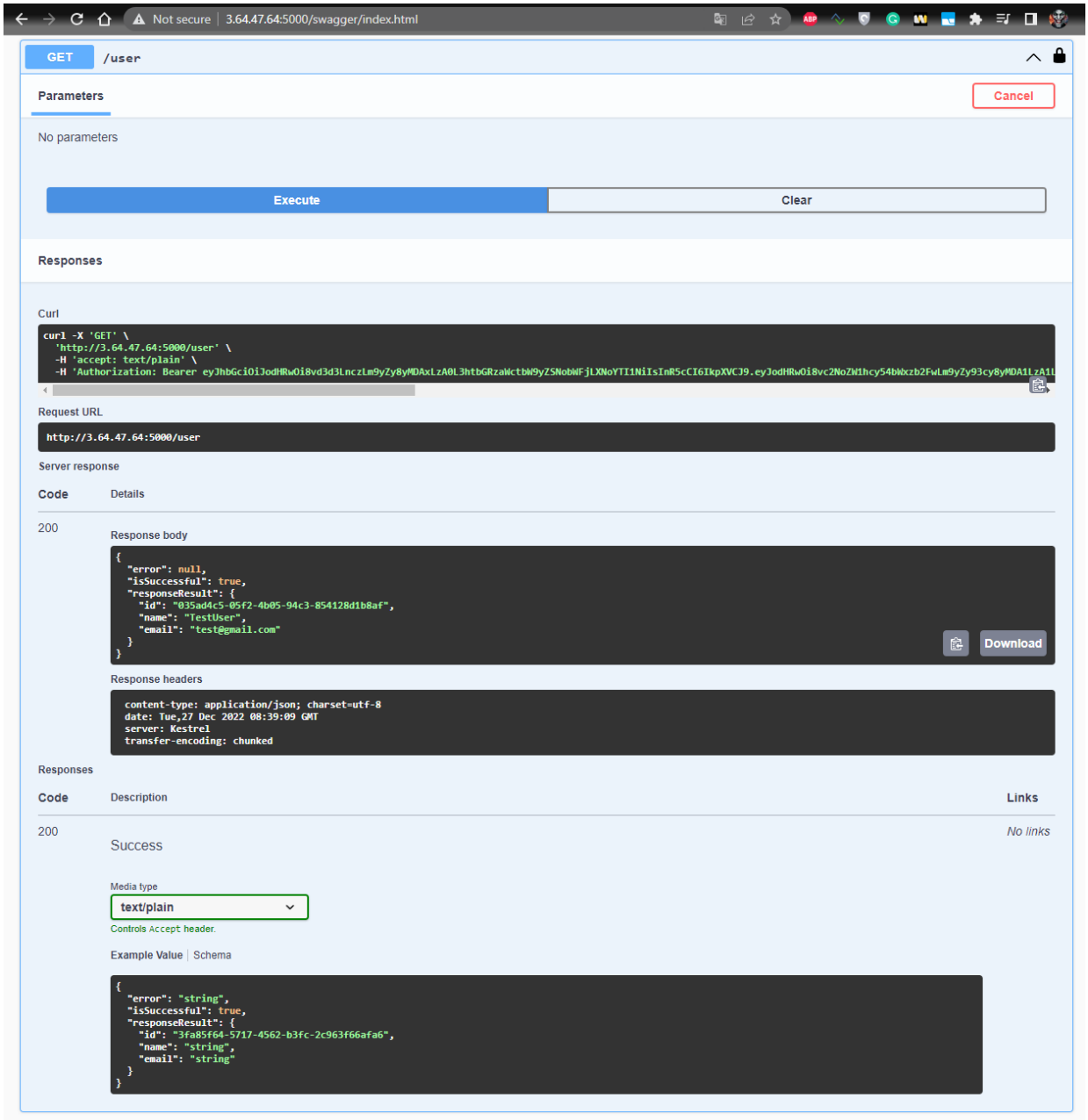
```
ubuntu@ip-172-31-22-246:~/labs/lab3$ docker-compose up
Creating network "lab3_default" with the default driver
Creating volume "lab3_dbdata" with local driver
Pulling postgres (postgres:13.7)...
13.7: Pulling from library/postgres
1efc276f4ff9: Pull complete
66c520df917d: Pull complete
5b124e6748c9: Pull complete
1a06bb042d01: Pull complete
e3849c675ec5: Pull complete
d3b2eaf1435b: Pull complete
074399829dc9: Pull complete
6feb085525d8: Pull complete
4153d17924d6: Pull complete
bc311b90edd7: Pull complete
9dab89a024b4: Pull complete
e60b3f3ab3f2: Pull complete
0091f9daf172: Pull complete
Digest: sha256:03652c675ae177af98ddd50f9f4b4b2cf8ad38d0e116aa68fe670fbc2cf250fc
Status: Downloaded newer image for postgres:13.7
Creating lab3_postgres_1 ... done
Creating jotterapi       ... done
Attaching to lab3_postgres_1, jotterapi
postgres_1   | The files belonging to this database system will be owned by user "postgres".
postgres_1   | This user must also own the server process.
postgres_1   |
postgres_1   | The database cluster will be initialized with locale "en_US.utf8".
postgres_1   | The default database encoding has accordingly been set to "UTF8".
postgres_1   | The default text search configuration will be set to "english".
postgres_1   |
postgres_1   | Data page checksums are disabled.
postgres_1   |
postgres_1   | fixing permissions on existing directory /var/lib/postgresql/data ... ok
postgres_1   | creating subdirectories ... ok
postgres_1   | selecting dynamic shared memory implementation ... posix
postgres_1   | selecting default max_connections ... 100
postgres_1   | selecting default shared_buffers ... 128MB
postgres_1   | 2022-12-27 08:33:39.105 UTC [47] LOG:  database system was shut down at 2022-12-27 08:33:38 UTC
postgres_1   | 2022-12-27 08:33:39.110 UTC [46] LOG:  database system is ready to accept connections
postgres_1   |  done
postgres_1   | server started
jotterapi    | warn: Microsoft.AspNetCore.DataProtection.KeyManagement.XmlKeyManager[35]
jotterapi    |       No XML encryptor configured. Key {a1b3c701-8f53-4434-ab7e-e90f4bcbe7ed} may be persisted to storage in unencrypted form.
postgres_1   | CREATE DATABASE
postgres_1   |
postgres_1   |
postgres_1   | /usr/local/bin/docker-entrypoint.sh: ignoring /docker-entrypoint-initdb.d/*
postgres_1   |
postgres_1   | 2022-12-27 08:33:39.432 UTC [46] LOG:  received fast shutdown request
postgres_1   | waiting for server to shut down....2022-12-27 08:33:39.434 UTC [46] LOG:  aborting any active transactions
postgres_1   | 2022-12-27 08:33:39.444 UTC [46] LOG:  background worker "logical replication launcher" (PID 53) exited with exit code 1
postgres_1   | 2022-12-27 08:33:39.446 UTC [48] LOG:  shutting down
postgres_1   | 2022-12-27 08:33:39.463 UTC [46] LOG:  database system is shut down
postgres_1   |  done
postgres_1   | server stopped
postgres_1   |
postgres_1   | PostgreSQL init process complete; ready for start up.
postgres_1   |
postgres_1   | 2022-12-27 08:33:39.578 UTC [1] LOG:  starting PostgreSQL 13.7 (Debian 13.7-1.pgdg110+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 10.2.
1-6) 10.2.1 20210110, 64-bit
postgres_1   | 2022-12-27 08:33:39.579 UTC [1] LOG:  listening on IPv4 address "0.0.0.0", port 5432
postgres_1   | 2022-12-27 08:33:39.579 UTC [1] LOG:  listening on IPv6 address "::", port 5432
postgres_1   | 2022-12-27 08:33:39.584 UTC [1] LOG:  listening on Unix socket "/var/run/postgresql/.s.PGSQL.5432"
postgres_1   | 2022-12-27 08:33:39.590 UTC [60] LOG:  database system was shut down at 2022-12-27 08:33:39 UTC
postgres_1   | 2022-12-27 08:33:39.595 UTC [1] LOG:  database system is ready to accept connections
postgres_1   | 2022-12-27 08:33:40.933 UTC [67] FATAL:  database "jotter" does not exist
postgres_1   | 2022-12-27 08:33:40.958 UTC [68] FATAL:  database "jotter" does not exist
postgres_1   | 2022-12-27 08:33:40.963 UTC [69] FATAL:  database "jotter" does not exist
jotterapi    | info: Microsoft.Hosting.Lifetime[14]
jotterapi    |       Now listening on: http://0.0.0.0:5000
jotterapi    | info: Microsoft.Hosting.Lifetime[0]
jotterapi    |       Application started. Press Ctrl+C to shut down.
jotterapi    | info: Microsoft.Hosting.Lifetime[0]
jotterapi    |       Hosting environment: Production
jotterapi    | info: Microsoft.Hosting.Lifetime[0]
jotterapi    |       Content root path: /output
```

21. Перевіримо його на працездатність:




Not secure | 3.64.47.64:5000/swagger/index.html
Swagger
Supported by SMARTBEAR
Select a definition
JotterAPI v1
Jogger API
v1
OAS3
http://3.64.47.64:5000/swagger/v1/swagger.json
Authorize
Categories
POST /Categories
GET /Categories
DELETE /Categories/{categoryId}
Files
POST /Files
GET /Files/{fileId}
DELETE /Files/{fileId}
Notes
POST /Notes
PUT /Notes
DELETE /Notes/{noteId}
GET /Notes/{noteId}
GET /Notes/category
Test
GET /Test
Users
POST /login


Not secure | 3.64.47.64:5000/swagger/index.html
POST
/register
Parameters
Cancel
Reset
No parameters
Request body
application/json
{
"email": "test@gmail.com",
"name": "TestUser",
"password": "Test12345678"
}
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://3.64.47.64:5000/register' \
-H 'accept: text/plain' \
-H 'Content-Type: application/json' \
-d '{
"email": "test@gmail.com",
"name": "TestUser",
"password": "Test12345678"
}'
Request URL
http://3.64.47.64:5000/register
Server response
Code
Details
200
Response body
{
"error": null,
"isSuccessful": true,
"responseResult": {
"accessToken": "eyJhbGciOiJodHRwOi8vd3d3LnczLm9yZy8yMDAxLzA0L3htbGRzaWctbW9yZSNobWFjLXNoYTI1NiIsInR5cCI6IkpXVCJ9.eyJodHRwOi8vc2NoZW1hcy54bWxzb2FwLm9yZy93cy8yMDA1LzA1L2lkZW50aXR5L2NsYW1tcy9uYW11aWRlbnRpZm11ciI6Il1c3RVc2VyIiwiaHR0cDovL3NjaGVtYXMueG1sc29hcC5vcmcvd3MvMjAwNS8wNS9pZGVudG10eS9jbGFpbXMvbmFtZSI6IjAzNWFkNGM1LTA1ZjItNGIwNS05NGMzLTg1NDEyOGQxYjhhZiIsImV4cCI6MTY3MjIxNjcxMiwiaXNzIjoiSm90dGVyIiwiYXVkIjoiU29tZSBhdWRpZW5jZSJ9.2oPfgosUa2m4Qq8G5O8kqBeqk_NVXV-XNJirLSvnsTY"
}
}
Download
Response headers
access-control-allow-origin: *
content-type: application/json; charset=utf-8
date: Tue,27 Dec 2022 08:38:31 GMT
server: Kestrel
transfer-encoding: chunked



Висновок:

В результаті виконання лабораторної роботи було розгорнуто .NET застосунок за допомогою Docker (як локально, так і на сервері). docker image було опубліковано в публічному Container registry – Amazon Elastic Container Registry.