Politechnika Krakowska

Wydział Informatyki i Telekomunikacji

Studia Stacjonarne

Sprawozdanie z przedmiotu:

Zaawansowane Techniki Programowania

Laboratorium nr 5

Wykonał

Jan Kopeć

Wprowadzenie do Dockera

Cel zadania:

nauka podstawowych funkcji **Dockera**, takich jak tworzenie obrazu, uruchamianie kontenerów, zarządzanie nimi, oraz komunikacja między kontenerami.

1. Podstawowe komendy Dockera

Docker jest zainstalowany w wersji 20.10.8 i jest uruchomiony

2. Tworzenie i uruchamianie kontenera

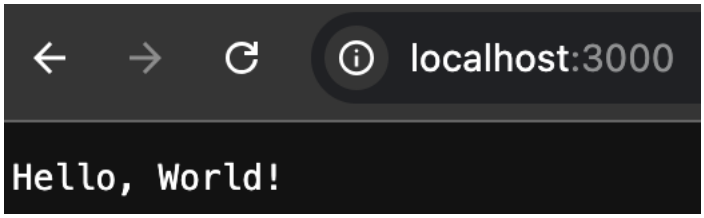
```
jankopec@pc3781 lab5 % docker pull alpine
Using default tag: latest
latest: Pulling from library/alpine
f18232174bc9: Pull complete
Digest: sha256:a8560b36e8b8210634f77d9f7f9efd7ffa463e380b75e2e74aff4511df3ef88c
Status: Downloaded newer image for alpine:latest
docker.io/library/alpine:latest
jankopec@pc3781 lab5 % docker run -it alpine /bin/sh
/ # echo "Hello, Docker!"
Hello, Docker!
/ # exit
jankopec@pc3781 lab5 % docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
jankopec@pc3781 lab5 % docker ps -a
CONTAINER ID   IMAGE       COMMAND                  CREATED          STATUS                      PORTS                NAMES
5b7e6b97b100   alpine      "/bin/sh"                37 seconds ago   Exited (0) 14 seconds ago                        blissful_jackson
```

3. Tworzenie własnego obrazu Dockera

```
jankopec@pc3781 lab5 % docker build -t my-node-app .

[+] Building 4.2s (9/9) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 128B
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [internal] load metadata for docker.io/library/node:14
 => [1/4] FROM docker.io/library/node:14@sha256:a158d3b9b4e3fa813fa6c8c590b8f0a860e015ad4e59bbce5744d2f6fd8461aa
 => [internal] load build context
 => => transferring context: 1.68kB
 => CACHED [2/4] WORKDIR /app
 => [3/4] COPY . .
 => [4/4] RUN npm install
 => exporting to image
 => => exporting layers
 => => writing image sha256:307eca8f9a80ebeb202bcac19a08cb2a9b959461da0223e6c2a0233a22e63e21
 => => naming to docker.io/library/my-node-app
```

```
jankopec@pc3781 lab5 % docker run -p 3000:3000 my-node-app
Server running on port 3000
```



4. Zarządzanie kontenerami

```
jankopec@pc3781 lab5 % docker ps
CONTAINER ID   IMAGE         COMMAND                  CREATED         STATUS         PORTS                    NAMES
e1854eb4c842   my-node-app   "docker-entrypoint.s…"   3 minutes ago   Up 3 minutes   0.0.0.0:3000->3000/tcp   festive_liskov
jankopec@pc3781 lab5 % docker stop e1854eb4c842
e1854eb4c842
jankopec@pc3781 lab5 % docker rm e1854eb4c842
e1854eb4c842
jankopec@pc3781 lab5 % docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
```

5. Tworzenie i zarządzanie wolumenami

```
jankopec@pc3781 lab5 % docker volume create my_volume
my_volume
jankopec@pc3781 lab5 % docker run -d -v my_volume:/data alpine sleep 1000
6bd1e4f9afe92d326d622c86e181645b5f6e41791fe5ae4053b91138af09a54c
jankopec@pc3781 lab5 % docker exec -it 6bd1e4f9afe9 /bin/sh
/ # echo "Persistent data" > /data/persistent.txt
/ # %
jankopec@pc3781 lab5 % docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
jankopec@pc3781 lab5 % docker run -d -v my_volume:/data alpine sleep 1000
8c73635bd71761734d93a9ce2bbc45cf0dce6b1425a90337a4212f0ad664388a
jankopec@pc3781 lab5 % docker exec -it 8c73635bd717 /bin/sh
/ # cat /data/persistent.txt
Persistent data
/ #
```

```
jankopec@pc3781 lab5 % docker ps
CONTAINER ID   IMAGE     COMMAND        CREATED         STATUS         PORTS     NAMES
6bd1e4f9afe9   alpine    "sleep 1000"   7 minutes ago   Up 7 minutes             serene_robinson
jankopec@pc3781 lab5 % docker stop 6bd1e4f9afe9
6bd1e4f9afe9
jankopec@pc3781 lab5 % docker rm 6bd1e4f9afe9
6bd1e4f9afe9
jankopec@pc3781 lab5 % docker ps
CONTAINER ID   IMAGE     COMMAND        CREATED          STATUS          PORTS     NAMES
8c73635bd717   alpine    "sleep 1000"   24 seconds ago   Up 23 seconds             quizzical_blackburn
```

6. Sieciowanie kontenerów

```
jankopec@pc3781 lab5 % docker network ls
NETWORK ID     NAME              DRIVER    SCOPE
54a8ecda7d1a   backend           bridge    local
df44969f3a7b   backend_default   bridge    local
0ae4472b98b5   bridge            bridge    local
275a3e93618b   host              host      local
d72c6e1cc33e   minikube          bridge    local
a77b00eb61b4   my_network        bridge    local
```

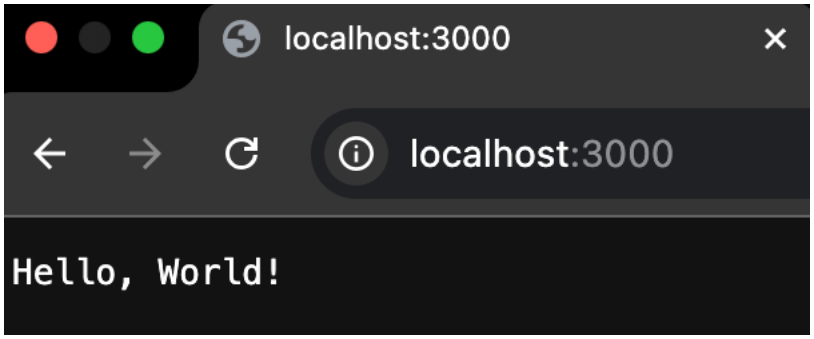
```
jankopec@pc3781 lab5 % docker run -d --name redis --network my_network --privileged redis

15556904d5ebf2c8a3ef3743452a6e22f1bb63b4e9cd3730c95b0191563018af
jankopec@pc3781 lab5 % docker ps
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS         PORTS      NAMES
15556904d5eb   redis     "docker-entrypoint.s…"   10 seconds ago   Up 9 seconds   6379/tcp   redis
jankopec@pc3781 lab5 % docker run -it --network my_network alpine /bin/sh
/ # apk add redis
fetch https://dl-cdn.alpinelinux.org/alpine/v3.21/main/x86_64/APKINDEX.tar.gz
fetch https://dl-cdn.alpinelinux.org/alpine/v3.21/community/x86_64/APKINDEX.tar.gz
(1/1) Installing redis (7.2.7-r0)
Executing redis-7.2.7-r0.pre-install
Executing redis-7.2.7-r0.post-install
Executing busybox-1.37.0-r12.trigger
OK: 10 MiB in 16 packages
/ # redis-cli -h redis ping
PONG
/ #
```

7. Docker Compose – Aplikacja wielokontenerowa

```yaml
version: '3'
▷Run All Services
services:
  ▷Run Service
  web:
    build: .
    ports:
      - "3000:3000"
  ▷Run Service
  redis:
    image: "redis"
    command: ["redis-server", "--protected-mode", "no", "--daemonize", "no"]
    privileged: true
```

```
jankopec@pc3781 lab5 % docker-compose up
Creating lab5_redis_1 ... done
Creating lab5_web_1   ... done
Attaching to lab5_redis_1, lab5_web_1
redis_1  | 1:C 25 Mar 2025 12:53:00.410 * oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
redis_1  | 1:C 25 Mar 2025 12:53:00.410 * Redis version=7.4.2, bits=64, commit=00000000, modified=0, pid=1, just started
redis_1  | 1:C 25 Mar 2025 12:53:00.410 * Configuration loaded
redis_1  | 1:M 25 Mar 2025 12:53:00.411 * monotonic clock: POSIX clock_gettime
redis_1  | 1:M 25 Mar 2025 12:53:00.412 * Running mode=standalone, port=6379.
redis_1  | 1:M 25 Mar 2025 12:53:00.413 * Server initialized
redis_1  | 1:M 25 Mar 2025 12:53:00.413 * Ready to accept connections tcp
web_1    | Server running on port 3000
```



8. Wnioski

Podczas laboratorium udało się przetestować podstawowe funkcje **Dockera**, takie jak tworzenie obrazu, uruchamianie kontenerów czy zarządzanie oraz komunikacja między kontenerami. Wykonano wszystkie zadania. W trakcie wykonywania zadań 6 i 7 pojawiły się problemy. Wynikały one z ograniczeń systemowych Dockera, które blokowały Redisowi tworzenie procesów w tle.

Rozwiązaniem było uruchomienie kontenera z flagą –privileged w zadaniu 6, co pozwoliło Redisowi działać poprawnie. Natomiast analogiczny problem w zadaniu 7 rozwiązano używając flagi privileged: true w pliku docker-compose.yml, aby nadać Redisowi wymagane uprawnienia. Po tych zmianach Redis uruchomił się poprawnie.

To laboratorium było wartościowym ćwiczeniem, które pozwoliło na praktyczne opanowanie podstawowych funkcji Dockera.