

# SPRAWOZDANIE

## Laboratorium 5

### Programowanie aplikacji w chmurze obliczeniowej

Rafał Seredowski Gr. 6.8.

#### 1. Treść utworzonego pliku Dockerfile:

```
Dockerfile > _
1 #Etap 1 budowy obrazu
2 #Użycie bazowego obrazu metodą od podstaw - from scratch
3 FROM scratch AS builder
4 #Definicja zmiennej
5 ARG VERSION
6 #Użycie obrazu bazowego z minimalną zawartością systemu Linux Alpine
7 ADD alpine-minirootfs-3.19.1-x86_64.tar /
8 #Określenie katalogu roboczego, w którym będzie tworzona aplikacja app.js
9 WORKDIR /usr/app
10 #Skopiowanie plików z lokalnego systemu plików do środowiska kontenera
11 COPY ./package.json ./
12 COPY ./app.js ./
13 #Aktualizacja pakietów APK, instalacja node.js i narzędzia npm oraz czyszczenie pamięci podręcznej cache
14 RUN apk update && apk add nodejs npm && rm -rf /var/cache/apk/*
15 #Uruchomienie instalacji zależności zdefiniowanych w pliku package.json
16 RUN npm install
17 #Informacja o porcie, na którym kontener będzie nasłuchiwał
18 EXPOSE 3000
19 #Etap 2 budowy obrazu
20 #Użycie obrazu Alpine z Nginx
21 FROM nginx:alpine
22 #Przechylenie zmiennej środowiskowej zdefiniowanej w etapie 1 budowy obrazu
23 ARG VERSION
24 ENV APP_VERSION=${VERSION:-v1}
25 #Określenie katalogu roboczego
26 WORKDIR /usr/share/nginx/html
27 #Skopiowanie plików z poprzedniego etapu budowy obrazu do obecnego
28 COPY --from=builder /usr/app /usr/share/nginx/html
29 #Skopiowanie pliku default.conf (pliku konfiguracyjnego dla serwera Nginx) z lokalnego systemu plików do kontenera
30 COPY ./default.conf /etc/nginx/conf.d
31 #Aktualizacja pakietów APK, instalacja node.js oraz czyszczenie pamięci podręcznej cache
32 RUN apk update && apk add nodejs && rm -rf /var/cache/apk/*
33 #Informacja o porcie, na którym kontener będzie nasłuchiwał
34 EXPOSE 80
35 #Użycie komendy HEALTHCHECK umożliwiającej sprawdzenie poprawności działania kontenera
36 HEALTHCHECK --interval=10s --timeout=1s --start-period=5s --retries=3 \
37   CMD curl -f http://localhost:80 || exit 1
38 #Uruchomienie serwera Nginx oraz aplikacji app.js
39 CMD nginx -g "daemon off;" & node app.js
```

#### 2. Polecenie do budowy obrazu:

docker build --build-arg VERSION=1.0.1 -t lab5:1 .

Wynik działania polecenia:

```
rafał@Rafał-Dell: ~/Pulpit/aplikacjeChmurowe/Lab5
Plik Edycja Widok Wyszukiwanie Terminal Karty Pomoc
rafał@Rafał-Dell: ~/Pulpit/aplikacjeChmurowe/Lab5 x rafał@Rafał-Dell: ~/Pulpit/aplikacjeChmurowe/Lab5 x
rafał@Rafał-Dell:~/Pulpit/aplikacjeChmurowe/Lab5$ docker build --build-arg VERSION=1.0.1 -t lab5:1 .
[+] Building 16.0s (16/16) FINISHED docker:default
=> [internal] load build definition from Dockerfile 0.0s
=> => transferring dockerfile: 1.90kB 0.0s
=> [internal] load metadata for docker.io/library/nginx:alpine 0.0s
=> [internal] load .dockerignore 0.0s
=> => transferring context: 2B 0.0s
=> [internal] load build context 0.0s
=> => transferring context: 149B 0.0s
=> [stage-1 1/5] FROM docker.io/library/nginx:alpine 0.0s
=> CACHED [builder 1/6] ADD alpine-minirootfs-3.19.1-x86_64.tar / 0.0s
=> CACHED [builder 2/6] WORKDIR /usr/app 0.0s
=> CACHED [builder 3/6] COPY ./package.json ./ 0.0s
=> CACHED [builder 4/6] COPY ./app.js ./ 0.0s
=> [builder 5/6] RUN apk update && apk add nodejs npm && rm -rf /var/cache/apk/* 5.2s
=> [builder 6/6] RUN npm install 6.8s
=> CACHED [stage-1 2/5] WORKDIR /usr/share/nginx/html 0.0s
=> CACHED [stage-1 3/5] COPY --from=builder /usr/app /usr/share/nginx/html 0.0s
=> CACHED [stage-1 4/5] COPY ./default.conf /etc/nginx/conf.d 0.0s
=> [stage-1 5/5] RUN apk update && apk add nodejs && rm -rf /var/cache/apk/* 3.5s
=> exporting to image 0.3s
=> => exporting layers 0.3s
=> => writing image sha256:a33acffa66322f6e07f7ed98a893458d37a9c2cec90f30e1eb1840a95cf6aa5a 0.0s
=> => naming to docker.io/library/lab5:1 0.0s
rafał@Rafał-Dell:~/Pulpit/aplikacjeChmurowe/Lab5$
```

### 3. Polecenie uruchamiające serwer:

`docker run -it -p 8888:80 --name aplikacja_lab5 lab5:1`

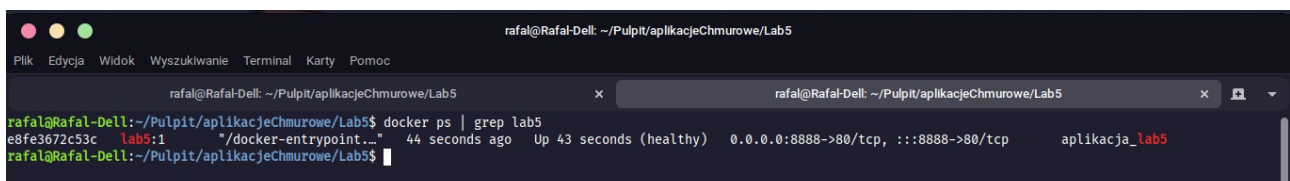
#### Wynik działania polecenia:

```
rafal@Rafal-Dell: ~/Pulpit/aplikacjeChmurowe/Lab5$ docker run -it -p 8888:80 --name aplikacja_lab5 lab5:1
2024/04/07 13:15:23 [notice] 6#6: using the "epoll" event method
2024/04/07 13:15:23 [notice] 6#6: nginx/1.25.4
2024/04/07 13:15:23 [notice] 6#6: built by gcc 12.2.1 20220924 (Alpine 12.2.1_git20220924-r10)
2024/04/07 13:15:23 [notice] 6#6: OS: Linux 5.15.0-97-generic
2024/04/07 13:15:23 [notice] 6#6: getrlimit(RLIMIT_NOFILE): 1048576:1048576
2024/04/07 13:15:23 [notice] 6#6: start worker processes
2024/04/07 13:15:23 [notice] 6#6: start worker process 7
2024/04/07 13:15:23 [notice] 6#6: start worker process 8
2024/04/07 13:15:23 [notice] 6#6: start worker process 9
2024/04/07 13:15:23 [notice] 6#6: start worker process 10
Aplikacja działa na porcie: 3000
```

### 4. Polecenie potwierdzające działanie kontenera i poprawne funkcjonowanie opracowanej aplikacji:

`docker ps | grep lab5`

#### Wynik działania polecenia:



```
rafal@Rafal-Dell: ~/Pulpit/aplikacjeChmurowe/Lab5$ docker ps | grep lab5
e8fe3672c53c lab5:1 "/docker-entrypoint..." 44 seconds ago Up 43 seconds (healthy) 0.0.0.0:8888->80/tcp, :::8888->80/tcp aplikacja_lab5
```

#### Wynik działania aplikacji w przeglądarce:

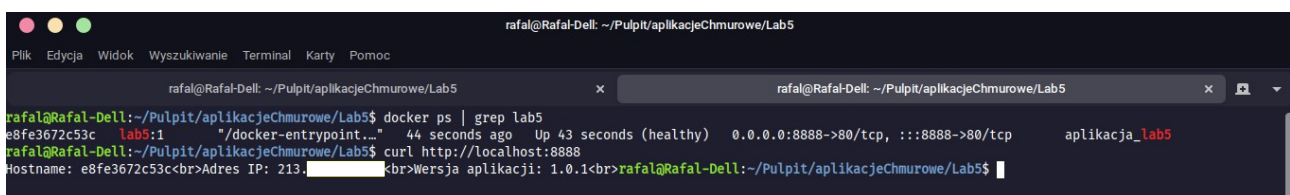


Mozilla Firefox

localhost:8888

Hostname: e8fe3672c53c  
Adres IP: 213.  
Wersja aplikacji: 1.0.1

#### Wynik działania aplikacji za pomocą polecenia curl:



```
rafal@Rafal-Dell: ~/Pulpit/aplikacjeChmurowe/Lab5$ docker ps | grep lab5
e8fe3672c53c lab5:1 "/docker-entrypoint..." 44 seconds ago Up 43 seconds (healthy) 0.0.0.0:8888->80/tcp, :::8888->80/tcp aplikacja_lab5
rafal@Rafal-Dell: ~/Pulpit/aplikacjeChmurowe/Lab5$ curl http://localhost:8888
Hostname: e8fe3672c53c<br>Adres IP: 213.<br>Wersja aplikacji: 1.0.1<br>rafal@Rafal-Dell: ~/Pulpit/aplikacjeChmurowe/Lab5$
```