POLITECHNIKA
LUBELSKA
WYDZIAŁ ELEKTROTECHNIKI
I INFORMATYKI

Programowanie aplikacji w chmurze obliczeniowej

laboratorium 5

Igor Pąśko

grupa dziekańska: 6.10
numer albumu: 99655

Prowadzący laboratorium mgr. inż. K. Łazaruk

Lublin rok 2025

Dockerfile – budowanie aplikacji

Zawartość utworzonego pliku Dockerfile:

```
FROM node:18-alpine AS builder
WORKDIR /app
ARG VERSION=1.0.0
ENV VERSION=$VERSION
COPY package.json .
COPY index.js .
RUN npm install
```

Wybieramy katalog roboczy a następnie przekazujemy wersję jako zmienną. Kopiujemy pliki json i js które pojawią się w naszym projekcie.

Kolejna część pliku Dockerfile:

```
FROM alpine
RUN apk add --no-cache nodejs npm curl
WORKDIR /app
COPY --from=builder /app .
EXPOSE 3000
HEALTHCHECK --interval=30s --timeout=5s \
CMD curl -f http://localhost:3000 || exit 1
CMD ["node", "index.js"]
```

Ta część odpowiada za tworzenie obrazu. Instalujemy curl i nodejs. Ustawiamy tu katalog docelowy dla nginx, kopiujemy wcześniejszy plik html i ustawiamy healthcheck.

Tworzymy pliki index.js oraz package.json

Zawartość pliku json:

```
{
  "name": "lab5",
  "version": "1.0.0",
  "main": "index.js",
  "dependencies": {}
}
```

Zawartość pliku js:

```
const http = require('http');
const os = require('os');
const VERSION = process.env.VERSION || '1.0.0';
const server = http.createServer((req, res) => {
  const hostname = os.hostname();
  const ip = Object.values(os.networkInterfaces())
    .flat()
    .find((i) => i.family === 'IPv4' && !i.internal)?.address || 'unknown';
  const html = `
    <html>
      <head><title>LAB 5</title></head>
      <body>
        <h1>Adres IP: ${ip}</h1>
        <h2>Nazwa hosta: ${hostname}</h2>
        <h3>Wersja aplikacji: ${VERSION}</h3>
      </body>
    </html>
  `;
  res.writeHead(200, { 'Content-Type': 'text/html' });
  res.end(html);
});
const PORT = process.env.PORT || 3000;
server.listen(PORT, () => console.log(`Serwer działa na porcie ${PORT}`));
```

Plik js tworzy dynamiczny plik html którego zadaniem jest wyświetlenie adresu ip serwera oraz nazwy hosta.
Const ip pobiera zewnętrzny adres serwera pomijając „localhost”.
Const port uruchamia serwer na porcie zdefiniowanym w zmiennej PORT czyli na porcie 3000.

Budowanie obrazu za pomocą komendy:

docker build -t lab5app:v1 --build-arg VERSION=2.5.1 .

```
PS C:\Users\pasko\Downloads\LAB5> docker build -t lab5app:v1 --build-arg VERSION=2.5.1 .
[+] Building 6.8s (17/17) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 496B
 => [internal] load metadata for docker.io/library/node:18-alpine
 => [internal] load metadata for docker.io/library/alpine:3.18
 => [auth] library/node:pull token for registry-1.docker.io
 => [auth] library/alpine:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [builder 1/5] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e
 => => resolve docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e
 => [internal] load build context
 => => transferring context: 966B
 => [stage-1 1/4] FROM docker.io/library/alpine:3.18@sha256:de0eb0b3f2a47ba1eb89389859a9bd88b28e82f5826b6969ad604979713c2d4f
 => => resolve docker.io/library/alpine:3.18@sha256:de0eb0b3f2a47ba1eb89389859a9bd88b28e82f5826b6969ad604979713c2d4f
 => => sha256:44cf07d57ee4424189f012074a59110ee2065adfdde9c7d9826bebdffce0a885 3.42MB / 3.42MB
 => => extracting sha256:44cf07d57ee4424189f012074a59110ee2065adfdde9c7d9826bebdffce0a885
 => CACHED [builder 2/5] WORKDIR /app
 => [builder 3/5] COPY package.json .
 => [builder 4/5] COPY index.js .
 => [builder 5/5] RUN npm install
 => [stage-1 2/4] RUN apk add --no-cache nodejs npm curl
 => [stage-1 3/4] WORKDIR /app
 => [stage-1 4/4] COPY --from=builder /app .
 => exporting to image
 => => exporting layers
 => => exporting manifest sha256:0a05996ada4da37136439da8d042eb08c3bbbb3447feee9c0992f7145516ecbd
 => => exporting config sha256:7471e188fbb0f0f1c735430f0b30acbe3a9f19ed089c5ef5401531ff7131ee6f
 => => exporting attestation manifest sha256:2182ee56b1df3d1e2e8e6e146a609d8734089f126dc9b19e03cc3d7642495277
 => => exporting manifest list sha256:70c60a20c33e4a613d5043538218fa44e39add04a29abd46bb1868b1160782d3
 => => naming to docker.io/library/lab5app:v1
 => => unpacking to docker.io/library/lab5app:v1
```

Uruchamianie kontenera za pomocą koemdu:

docker run -d -p 8080:3000 --name lab5app lab5app:v1

```
View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/1in0fu4uzuhsdxugsv9huy6ot
PS C:\Users\pasko\Downloads\LAB5> docker run -d -p 8080:3000 --name lab5app lab5app:v1
c82c79b868716a888e9977049b9655e2f1c5f15c16ab3cb3139c5344c54deffa
```

Sprawdzenie działania kontenera za pomocą curl:

```
PS C:\Users\pasko\Downloads\LAB5> curl http://localhost:8080


StatusCode        : 200
StatusDescription : OK
Content           :
                    <html>
                      <head><title>LAB 5</title></head>
                      <body>
                        <h1>Adres IP: 172.17.0.2</h1>
                        <h2>Nazwa hosta: c82c79b86871</h2>
                        <h3>Wersja aplikacji: 1.0.0</h3>
                      </body>...
RawContent        : HTTP/1.1 200 OK
                    Connection: keep-alive
                    Keep-Alive: timeout=5
                    Transfer-Encoding: chunked
                    Content-Type: text/html
                    Date: Fri, 04 Apr 2025 10:04:56 GMT


                    <html>
                      <head><title>LAB 5</title>...
Forms             : {}
Headers           : {[Connection, keep-alive], [Keep-Alive, timeout=5], [Transfer-Encoding, chunked], [Content-Type, text/html]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : System.__ComObject
RawContentLength  : 215


PS C:\Users\pasko\Downloads\LAB5>
```

Aby spełnić wszystkie wymagania należy użyć obrazu nginx jako finalnego, sprawić by plik js generował plik index.html. Do tego będą potrzebne zmiany w plikach js i dockerfile.

Oto zmodyfikowany plik dockerfile:

```
FROM node:18-alpine AS builder
WORKDIR /app
ARG VERSION=1.0.0
ENV VERSION=$VERSION
COPY index.js .
COPY package.json .
RUN npm install
RUN node index.js > index.html
FROM nginx:alpine
WORKDIR /usr/share/nginx/html
COPY --from=builder /app/index.html .
HEALTHCHECK --interval=30s --timeout=5s CMD wget --spider http://localhost || exit 1
```

Komentarz do zmian:
Budujemy stronę html przez node.js. Następnie uruchamiamy node by wygenerowany został plik index.html. Używamy serwera nginx i zmieniamy katalog roboczy na domyślny nginx i kopiujemy wygenerowaną stonę html.

Modyfikacje pliku js:

```
const os = require('os');
const VERSION = process.env.VERSION || '1.0.0';
const hostname = os.hostname();
const ip = Object.values(os.networkInterfaces())
  .flat()
  .find((i) => i.family === 'IPv4' && !i.internal)?.address || 'unknown';
const html = `
<html>
  <head><title>LAB 5</title></head>
  <body>
    <h1>Adres IP: ${ip}</h1>
    <h2>Host: ${hostname}</h2>
    <h3>Wersja: ${VERSION}</h3>
  </body>
</html>
`;
console.log(html);
```

Teraz plik js. nie uruchamia serwera a generuje plik html.

Przechodzimy do budowania.

Używamy komendy:
docker build -t lab5-nginx:v1 --build-arg VERSION=3.2.1 .

```
PS C:\Users\pasko\Downloads\LAB5> docker build -t lab5-nginx:v1 --build-arg VERSION=3.2.1 .
[+] Building 2.8s (17/17) FINISHED
 => [internal] load build definition from Dockerfile
 => => transferring dockerfile: 388B
 => [internal] load metadata for docker.io/library/node:18-alpine
 => [internal] load metadata for docker.io/library/nginx:alpine
 => [auth] library/nginx:pull token for registry-1.docker.io
 => [auth] library/node:pull token for registry-1.docker.io
 => [internal] load .dockerignore
 => => transferring context: 2B
 => [builder 1/6] FROM docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e
 => => resolve docker.io/library/node:18-alpine@sha256:8d6421d663b4c28fd3ebc498332f249011d118945588d0a35cb9bc4b8ca09d9e
 => [internal] load build context
 => => transferring context: 535B
 => [stage-1 1/3] FROM docker.io/library/nginx:alpine@sha256:4ff102c5d78d254a6f0da062b3cf39eaf07f01eec0927fd21e219d0af8bc0591
 => => resolve docker.io/library/nginx:alpine@sha256:4ff102c5d78d254a6f0da062b3cf39eaf07f01eec0927fd21e219d0af8bc0591
 => CACHED [builder 2/6] WORKDIR /app
 => CACHED [stage-1 2/3] WORKDIR /usr/share/nginx/html
 => [builder 3/6] COPY index.js .
 => [builder 4/6] COPY package.json .
 => [builder 5/6] RUN npm install
 => [builder 6/6] RUN node index.js > index.html
 => [stage-1 3/3] COPY --from=builder /app/index.html .
 => exporting to image
 => => exporting layers
 => => exporting manifest sha256:e72ff3ef2c32c8b515c6c6135badbe7a699001df29add3728ec02400e0ceea3a
 => => exporting config sha256:e6c623f02f1448ebd1d6f3ad73fc4ec2c25fbef4e83501b57d0e0dc596326551
 => => exporting attestation manifest sha256:1372f69ab9d358862c4d53b9329b956f1e4f595ea0123398efe02297d1d0346b
 => => exporting manifest list sha256:d7b7ba42635d5843a4d38d7517bb5dc09c9db0056c2e490ef03e2e638ea4a3e8
 => => naming to docker.io/library/lab5-nginx:v1
 => => unpacking to docker.io/library/lab5-nginx:v1

View build details: docker-desktop://dashboard/build/desktop-linux/desktop-linux/8n250i8ndmjbyei7e4p6ecmgu
PS C:\Users\pasko\Downloads\LAB5>
```

Uruchamiamy za pomocą komendy:
docker run -d -p 8080:80 --name lab5-nginx lab5-nginx:v1

```
PS C:\Users\pasko\Downloads\LAB5> docker run -d -p 8080:80 --name lab5-nginx lab5-nginx:v1
5ae8b8eabdd85c89aec49e9eeee100fb844c2e61b203b0ab3b54e8d5f029f139
```

Test za pomocą polecenia:
curl http://localhost:8080

```
PS C:\Users\pasko\Downloads\LAB5> curl http://localhost:8080


StatusCode        : 200
StatusDescription : OK
Content           :
                    <html>
                      <head><title>Static Page</title></head>
                      <body>
                        <h1>Adres IP: 172.17.0.4</h1>
                        <h2>Host: buildkitsandbox</h2>
                        <h3>Wersja: 3.2.1</h3>
                      </body>
                    </html>


RawContent        : HTTP/1.1 200 OK
                    Connection: keep-alive
                    Accept-Ranges: bytes
                    Content-Length: 174
                    Content-Type: text/html
                    Date: Fri, 04 Apr 2025 10:32:11 GMT
                    ETag: "67efb450-ae"
                    Last-Modified: Fri, 04 Apr 2025 1...
Forms             : {}
Headers           : {[Connection, keep-alive], [Accept-Ranges, bytes], [Content-Length, 174], [Content-Type, text/html]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : System.__ComObject
RawContentLength  : 174
```