Etap 1:
Najpierw tworzymy 2 pliki z następującą zawartością:
Dockerfile, który będzie wykorzystywał wieloetapowe budowanie obrazów (multi-stage build).

```
# Etap pośredni: Budowanie aplikacji w Node.js
FROM node:alpine AS builder

WORKDIR /usr/app
COPY ./package.json ./
RUN npm install
COPY ./index.js ./

# Etap 1: Przeniesienie aplikacji do scratch
FROM scratch

# Kopiowanie binarki Node.js
COPY --from=builder /usr/local/bin/node /usr/local/bin/node

# Kopiowanie bibliotek systemowych wymaganych przez Node.js
COPY --from=builder /lib/ld-musl-x86_64.so.1 /lib/ld-musl-x86_64.so.1
COPY --from=builder /usr/lib/libstdc++.so.6 /usr/lib/libstdc++.so.6
COPY --from=builder /usr/lib/libgcc_s.so.1 /usr/lib/libgcc_s.so.1

# Kopiowanie aplikacji
COPY --from=builder /usr/app /usr/app

# Ustawienie katalogu roboczego
WORKDIR /usr/app

# Definicja zmiennej VERSION
ARG VERSION=1.0.0
ENV VERSION=$VERSION

# Uruchomienie aplikacji
CMD ["/usr/local/bin/node", "index.js"]
```

index.js (plik z kodem aplikacji)

```
const http = require('http');
const os = require('os');
const version = process.env.VERSION || '1.0.0';

const server = http.createServer((req, res) => {
  const ip = req.socket.localAddress || 'unknown';
  const hostname = os.hostname();

  res.writeHead(200, { 'Content-Type': 'text/html' });
  res.end(`
    <h1>Aplikacja Webowa</h1>
    <p>Adres IP serwera: ${ip}</p>
    <p>Nazwa serwera (hostname): ${hostname}</p>
    <p>Wersja aplikacji: ${version}</p>
  `);
});

server.listen(8080, () => {
  console.log('Serwer działa na porcie 8080');
});
```

package.json (konfiguracja projektu)

```json
{
  "name": "simpleweb",
  "version": "1.0.0",
  "description": "Prosta aplikacja webowa dla laboratorium 5",
  "main": "index.js",
  "scripts": {
    "start": "node index.js"
  },
  "dependencies": {}
}
```

Następnie wpisujemy komendę, aby stworzyć obraz Dockera:

```
PS C:\Users\Who\OneDrive\Рабочий стол\Docker> docker run -d -p 8080:8080 --name my-app-container app-builder
>>
f5e41b31fa4b9f0ecd3d9403d8c4e10260a1b8a1a4d459b2bb76f9e9705d502a
PS C:\Users\Who\OneDrive\Рабочий стол\Docker>
```

Name	Tag	Image ID	Created	Size	Actions
webapp-scratch	1.0	e60a8ebc75b2	3 minutes ago	178.96 MB	

Następnie uruchomimy kontener:

```
PS C:\Users\Who\OneDrive\Рабочий стол\Docker\Lab5> docker run -d -p 8080:8080 --name webapp-scratch-test webapp-scratch:1.0
5f0c38238c3f98e819f7dab9337369149d353452d1581fd0a9dc80c686c712c9
PS C:\Users\Who\OneDrive\Рабочий стол\Docker\Lab5>
```

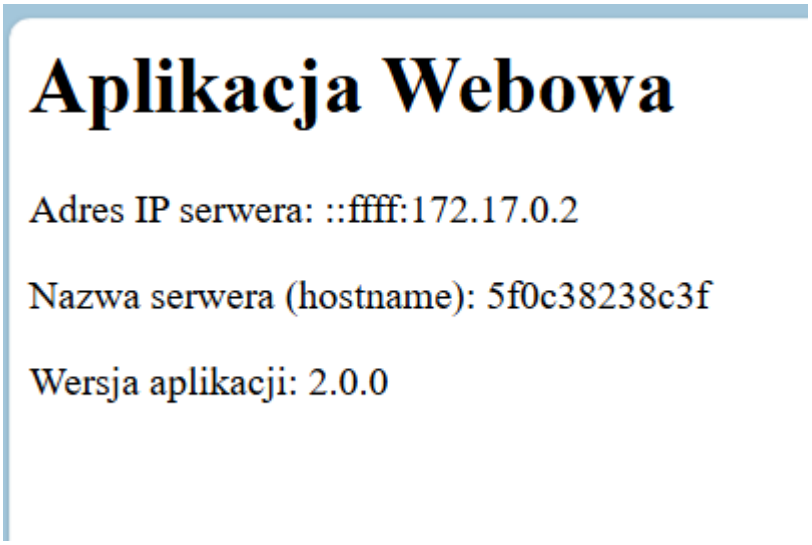
Name	Container ID	Image	Port(s)	Last start...	Actions
webapp-scratch-test	5f0c38238c3f	webapp-scratch:1.0	8080:8080	17 seconds ago	

Na końcu przetestowałam działanie aplikacji, używając curl:

```
PS C:\Users\Who\OneDrive\Рабочий стол\Docker\Lab5> curl http://localhost:8080


StatusCode        : 200
StatusDescription : OK
Content           :
                        <h1>Aplikacja Webowa</h1>
                        <p>Adres IP serwera: ::ffff:172.17.0.2</p>
Headers           : {[Connection, keep-alive], [Keep-Alive, timeout=5], [Transfer-Enco
                    ding, chunked], [Content-Type, text/html]...}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
RawContentLength  : 165
```

Wynik:



Etap 2:
Dodajemy drugi etap do pliku Dockerfile.

```
# Etap 2: Użycie Nginx jako serwera HTTP
FROM nginx:alpine

# Kopiowanie aplikacji i Node.js z obrazu z Etapu 1
COPY --from=webapp-scratch:1.0 /usr/app /usr/app
COPY --from=webapp-scratch:1.0 /usr/local/bin/node /usr/local/bin/node
COPY --from=webapp-scratch:1.0 /lib/ld-musl-x86_64.so.1 /lib/ld-musl-x86_64.so.1
COPY --from=webapp-scratch:1.0 /usr/lib/libstdc++.so.6 /usr/lib/libstdc++.so.6
COPY --from=webapp-scratch:1.0 /usr/lib/libgcc_s.so.1 /usr/lib/libgcc_s.so.1

# Kopiowanie konfiguracji Nginx
COPY ./nginx.conf /etc/nginx/nginx.conf

# Eksponowanie portu
EXPOSE 80

# HEALTHCHECK do sprawdzania działania
HEALTHCHECK --interval=10s --timeout=3s \
  CMD curl -f http://localhost/ || exit 1

# Uruchomienie Node.js i Nginx
CMD ["/bin/sh", "-c", "node /usr/app/index.js & nginx -g 'daemon off;'"]
```

Musimy również stworzyć plik konfiguracyjny Nginx, aby działał jako reverse proxy, przekierowujący ruch z portu 80 na naszą aplikację działającą na porcie 8080. Konfiguracja ta zapewnia prawidłowe działanie serwera oraz dodatkową warstwę bezpieczeństwa
nginx.conf:

```
# Liczba procesów roboczych Nginx (1 - domyślnie wystarczy dla prostych zastosowań)
worker_processes 1;

# Konfiguracja zdarzeń (np. połączeń)
events {
  # Maksymalna liczba jednoczesnych połączeń na jeden proces roboczy
  worker_connections 1024;
}


http {
  server {
    listen 80;
    server_name localhost;

    location / {
      proxy_pass http://localhost:8080;
      # Przekazanie nagłówka 'Host' do serwera docelowego
      proxy_set_header Host $host;
      # Przekazanie prawdziwego adresu IP klienta
      proxy_set_header X-Real-IP $remote_addr;
    }
  }
}
```

Wpisujemy komendę, aby stworzyć obraz Dockera:

```
PS C:\Users\Who\OneDrive\Рабочий стол\Docker\Lab5>
                                        > docker build -f Dockerfile -t webapp-nginx:1.0 .
[+] Building 4.1s (16/16) FINISHED                                                  docker:desktop-linux
 => [internal] load build definition from Dockerfile                                              0.0s
 => => transferring dockerfile: 1.71kB                                                            0.0s
 => [internal] load metadata for docker.io/library/webapp-scratch:1.0                             0.0s
 => [internal] load metadata for docker.io/library/nginx:alpine                                   1.0s
 => [internal] load .dockerignore                                                                 0.0s
 => => transferring context: 2B                                                                   0.0s
 => [stage-2 1/7] FROM docker.io/library/nginx:alpine@sha256:4ff102c5d78d254a6f0da062b3cf39eaf07f  0.0s
 => => resolve docker.io/library/nginx:alpine@sha256:4ff102c5d78d254a6f0da062b3cf39eaf07f01eec0927  0.0s
 => FROM docker.io/library/webapp-scratch:1.0@sha256:e60a8ebc75b2678e59af4bc694618a7d7effdb508cbb  2.1s
 => => resolve docker.io/library/webapp-scratch:1.0@sha256:e60a8ebc75b2678e59af4bc694618a7d7effdb  2.1s
 => [internal] load build context                                                                 0.0s
 => => transferring context: 32B                                                                  0.0s
 => [auth] library/webapp-scratch:pull token for registry-1.docker.io                             0.0s
 => [auth] library/webapp-scratch:pull token for registry-1.docker.io                             0.0s
 => CACHED [stage-2 2/7] COPY --from=webapp-scratch:1.0 /usr/app /usr/app                         0.0s
 => CACHED [stage-2 3/7] COPY --from=webapp-scratch:1.0 /usr/local/bin/node /usr/local/bin/node   0.0s
 => CACHED [stage-2 4/7] COPY --from=webapp-scratch:1.0 /lib/ld-musl-x86_64.so.1 /lib/ld-musl-x8  0.0s
 => CACHED [stage-2 5/7] COPY --from=webapp-scratch:1.0 /usr/lib/libstdc++.so.6 /usr/lib/libstdc+  0.0s
 => CACHED [stage-2 6/7] COPY --from=webapp-scratch:1.0 /usr/lib/libgcc_s.so.1 /usr/lib/libgcc_s.  0.0s
```

webapp-nginx	1.0	1fed92f18e74	3 minutes ago	250.97 MB	

Następnie uruchomimy kontener:

```
PS C:\Users\Who\OneDrive\Рабочий стол\Docker\Lab5> docker run -d -p 8080:80 --name webapp-nginx-test webapp-nginx:1.0
36ff61e2401f8e880fde95afacf7bbafd4b016289b9b87dbda8f859d6d1bd6a6
PS C:\Users\Who\OneDrive\Рабочий стол\Docker\Lab5>
```

webapp-nginx-test	36ff61e2401f	webapp-nginx:1.0	8080:80	10 seconds ago	

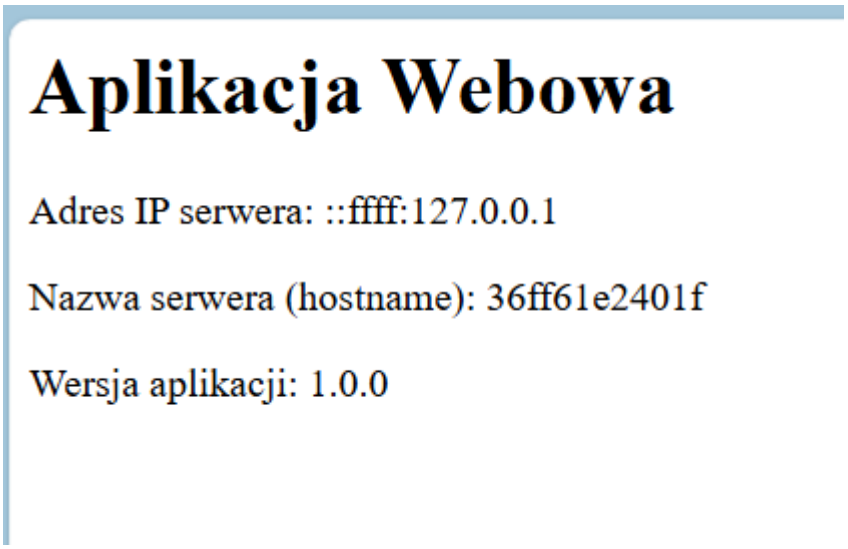
Na końcu przetestowałam działanie aplikacji, używając curl:

```
PS C:\Users\Who\OneDrive\Рабочий стол\Docker\Lab5> curl http://localhost:8080


StatusCode        : 200
StatusDescription : OK
Content           :
                        <h1>Aplikacja Webowa</h1>
                        <p>Adres IP serwera: ::1</p>
                    Connection: keep-alive
                    Content-Type: text/html
                    Date: Sun, 06 Apr 2025 16:08:53 GMT
                    Server: nginx/1.27.4


                        <h1>Aplikacja Webowa</h1>
                        <p>Adres I...
Forms             : {}
Headers           : {[Transfer-Encoding, chunked], [Connection, keep-alive], [Content-Type, text/html], [Date, Sun, 06 Apr 2025 16:08:53 GMT]..
                    .}
Images            : {}
InputFields       : {}
Links             : {}
ParsedHtml        : mshtml.HTMLDocumentClass
```

Wynik:


Aplikacja Webowa
Adres IP serwera: ::ffff:127.0.0.1
Nazwa serwera (hostname): 36ff61e2401f
Wersja aplikacji: 1.0.0