Kurs Dockerfile – absolutne podstawy na przykładzie NGINX

Wprowadzenie

Docker to narzędzie, które pozwala **pakować aplikacje razem z ich środowiskiem** (systemem, bibliotekami, konfiguracją) do jednej całości zwanej **obrazem**.
Z takiego obrazu uruchamia się **kontenery**, czyli działające instancje aplikacji.

Ten kurs pokazuje:

- czym jest **Dockerfile**,
- jak zbudować **własny obraz**,
- jak go **uruchomić**,
- jak **opublikować obraz w Docker Hubie**, aby inni mogli go użyć.

Wszystkie przykłady są oparte o **NGINX**, ponieważ:

- jest prosty,
- natychmiast pokazuje efekt w przeglądarce,
- jest często używany w praktyce.

---

1. Czym jest Dockerfile

Dockerfile to zwykły plik tekstowy o nazwie:

`Dockerfile`

Nie ma rozszerzenia (`.txt`, `.conf` itp.).

Dockerfile:

- opisuje **krok po kroku**, jak zbudować obraz,
- jest czytany przez Dockera linia po linii,
- każda linia to **instrukcja**.

Można powiedzieć, że Dockerfile to **przepis**, a obraz to **gotowe danie**.

---

2. Pierwszy minimalny Dockerfile z NGINX

Struktura katalogu projektu

```
nginx-dockerfile/
├── Dockerfile
└── index.html
```

Plik `index.html`

```
<!DOCTYPE html>
<html>
<head>
  <title>Docker + NGINX</title>
</head>
<body>
  <h1>Hello from Dockerfile</h1>
</body>
</html>
```

To jest zwykła strona HTML, którą NGINX będzie wyświetlał.

Plik `Dockerfile`

```
FROM nginx:alpine


COPY index.html /usr/share/nginx/html/index.html
```

Teraz dokładnie wyjaśnijmy **każdą linijkę**.

---

3. Wyjaśnienie instrukcji Dockerfile

FROM

```
FROM nginx:alpine
```

Ta linia oznacza:

> „Zaczynam budować swój obraz **na bazie istniejącego obrazu nginx**”

- `nginx` to oficjalny obraz serwera NGINX
- `alpine` oznacza, że bazuje na lekkim systemie Linux Alpine

Bez `FROM` **nie da się zbudować obrazu**.

COPY

```
COPY index.html /usr/share/nginx/html/index.html
```

Ta instrukcja:

- bierze plik `index.html` z twojego komputera,
- kopiuje go **do wnętrza obrazu**,
- umieszcza dokładnie w miejscu, z którego NGINX serwuje strony.

---

4. Budowanie obrazu Dockera

Aby z Dockerfile powstał obraz, używa się polecenia:

```
docker build -t sebastian-nginx:1.0 .
```

Wyjaśnienie:

- `docker build` – buduj obraz
- `-t sebastian-nginx:1.0` – nadaj obrazowi nazwę i wersję
- `.` – użyj Dockerfile z bieżącego katalogu

Po tej operacji obraz **istnieje lokalnie na komputerze**, ale:

- jeszcze nie działa,
- nie jest nigdzie opublikowany.

---

5. Uruchamianie kontenera z obrazu

Aby uruchomić obraz jako działającą aplikację:

```
docker run -d \
  --name nginx-test \
  -p 8080:80 \
  sebastian-nginx:1.0
```

Wyjaśnienie:

- `docker run` – uruchom kontener
- `-d` – uruchom w tle
- `--name nginx-test` – nadaj nazwę kontenerowi
- `-p 8080:80` – połącz port komputera z portem kontenera
- `sebastian-nginx:1.0` – obraz, z którego startujemy

Teraz w przeglądarce:

```
http://localhost:8080
```

zobaczysz swoją stronę HTML.

---

6. Czym jest Docker Hub i po co go używamy

Docker Hub to publiczny serwis, który:

- przechowuje obrazy Dockera,
- działa podobnie jak GitHub, ale dla obrazów,
- pozwala innym pobierać twoje obrazy.

Techniczna nazwa Docker Huba (registry) to:

`docker.io`

To bardzo ważne:

- `docker.io` → registry (techniczny adres)
- `hub.docker.com` → strona WWW

---

7. Jawne tagowanie obrazu do Docker Huba

Najpierw obraz istnieje **lokalnie**.
Aby powiedzieć Dockerowi, **gdzie ma go wysłać**, trzeba go poprawnie otagować.

```
docker tag sebastian-nginx:1.0 docker.io/sebastian/nginx-course:1.0
```

Ten zapis oznacza:

- `docker.io` → Docker Hub
- `sebastian` → konto użytkownika
- `nginx-course` → nazwa repozytorium
- `1.0` → wersja obrazu

Tagowanie **nie wysyła obrazu** – tylko go przygotowuje.

---

8. Logowanie do Docker Huba

Przed publikacją trzeba się zalogować:

```
docker login docker.io
```

Docker zapyta o:

- login,
- hasło lub token.

---

9. Publikowanie obrazu (push)

Dopiero to polecenie **wysyła obraz do Docker Huba**:

```
docker push docker.io/sebastian/nginx-course:1.0
```

Po zakończeniu:

- obraz jest dostępny w Docker Hubie,
- inni mogą go pobrać.

---

10. Użycie obrazu przez inną osobę

Inny użytkownik może uruchomić twój obraz jednym poleceniem:

```
docker run -d -p 8080:80 docker.io/sebastian/nginx-course:1.0
```

Nie potrzebuje:

- Dockerfile,
- plików projektu,
- twojego kodu.

Wystarczy obraz.

---

11. Zadania ćwiczeniowe

Zadanie 1

Zmień treść `index.html` i zbuduj nową wersję obrazu `1.1`.

Zadanie 2

Nadaj kontenerowi inną nazwę i uruchom go na innym porcie.

Zadanie 3

Opublikuj nową wersję obrazu w Docker Hubie.

Zadanie 4 (bonus)

Usuń lokalny obraz i pobierz go ponownie z Docker Huba.

Podsumowanie

- Dockerfile opisuje **jak zbudować obraz**
- Obraz to **szablon**
- Kontener to **działająca aplikacja**
- Docker Hub to **repozytorium obrazów**
- `docker tag` przygotowuje obraz
- `docker push` publikuje obraz

Jeżeli chcesz, kolejne logiczne kroki to:

- **Dockerfile + konfiguracja NGINX**
- **.dockerignore**
- **Docker Compose**
- **obrazy wieloetapowe (multi-stage)**

Możemy z tego zrobić **pełny, spójny kurs od zera do poziomu junior DevOps**.

uruchamianie zrob tylko w formie docker compose

Dobrze — poniżej masz **poprawioną wersję kursu**, w której **URUCHAMIANIE JEST WYŁĄCZNIE PRZEZ DOCKER COMPOSE**.
Styl pozostaje **ekspercki, dydaktyczny, dla osoby początkującej**, bez założeń o wiedzy domyślnej.

Możesz to **bezpośrednio podmienić w kursie**.

Kurs Dockerfile – absolutne podstawy na przykładzie NGINX

(uruchamianie wyłącznie przez Docker Compose)

---

1. Wprowadzenie

Docker pozwala pakować aplikacje wraz z ich środowiskiem do **obrazu**.
Z obrazu uruchamia się **kontenery** – czyli działające instancje aplikacji.

W tym kursie:

- **budujemy obraz** za pomocą Dockerfile,
- **uruchamiamy go WYŁĄCZNIE przez Docker Compose**,
- **publikujemy obraz w Docker Hubie** w sposób jawny i czytelny.

Nie używamy `docker run`, aby:

- od początku uczyć dobrych praktyk,
- jasno oddzielić **budowanie** od **uruchamiania**.

---

2. Czym jest Dockerfile

Dockerfile to plik tekstowy o dokładnej nazwie:

```
Dockerfile
```

Dockerfile:

- opisuje **jak zbudować obraz**,
- jest wykonywany linia po linii,
- każda instrukcja tworzy kolejną warstwę obrazu.

Dockerfile **nie uruchamia aplikacji** — on ją **pakuje**.

---

3. Minimalny projekt z NGINX

Struktura katalogu

```
nginx-dockerfile/
├── Dockerfile
├── docker-compose.yml
└── index.html
```

Plik `index.html`

```
<!DOCTYPE html>
<html>
<head>
  <title>Docker + NGINX</title>
</head>
<body>
  <h1>Hello from Dockerfile</h1>
</body>
</html>
```

---

4. Dockerfile – dokładne wyjaśnienie

Plik `Dockerfile`

```
FROM nginx:alpine

COPY index.html /usr/share/nginx/html/index.html
```

Co oznacza każda linia

FROM

```
FROM nginx:alpine
```

- używamy gotowego obrazu NGINX,
- bazuje on na lekkim systemie Alpine Linux,
- bez tej linijki Dockerfile **nie zadziała**.

COPY

```
COPY index.html /usr/share/nginx/html/index.html
```

- kopiuje plik z twojego komputera,
- umieszcza go **wewnątrz obrazu**,

- NGINX automatycznie go serwuje.

---

5. Budowanie obrazu Dockera

Aby z Dockerfile powstał obraz:

```
docker build -t sebastian-nginx:1.0 .
```

Wyjaśnienie:

- `docker build` – buduj obraz
- `-t sebastian-nginx:1.0` – nazwa i wersja
- `.` – użyj Dockerfile z bieżącego katalogu

Na tym etapie:

- obraz istnieje **lokalnie**,
- nic jeszcze nie działa,
- nic nie jest opublikowane.

---

6. Docker Compose – czym jest i po co go używamy

Docker Compose to narzędzie, które:

- opisuje **jak uruchamiać kontenery**,
- korzysta z pliku `docker-compose.yml`,
- eliminuje długie polecenia w terminalu.

Compose mówi:

> „Z jakiego obrazu uruchomić kontener,
> jak go nazwać,
> jakie porty wystawić,
> jak ma działać”.

---

7. Plik `docker-compose.yml`

```
version: "3.9"


services:
  nginx:
    image: sebastian-nginx:1.0
    container_name: nginx-test
    ports:
      - "8080:80"
```

Wyjaśnienie linia po linii

- `services` – lista kontenerów
- `nginx` – nazwa usługi (dowolna)
- `image` – obraz, z którego startujemy
- `container_name` – czytelna nazwa kontenera
- `ports` – mapowanie portów:
    - `8080` na komputerze
    - `80` w kontenerze

---

8. Uruchamianie kontenera przez Docker Compose

W katalogu projektu:

```
docker compose up -d
```

To polecenie:

- uruchamia kontener w tle,
- korzysta z `docker-compose.yml`,
- używa wcześniej zbudowanego obrazu.

Sprawdzenie w przeglądarce:

```
http://localhost:8080
```

---

9. Zatrzymywanie kontenera

```
docker compose down
```

To polecenie:

- zatrzymuje kontener,
- usuwa go,
- **nie usuwa obrazu**.

---

10. Docker Hub – czym jest registry

Docker Hub to publiczne registry obrazów Dockera.

Techniczna nazwa registry:

```
docker.io
```

- `docker.io` → registry (adres techniczny)
- `hub.docker.com` → strona internetowa

W tym kursie **zawsze podajemy registry jawnie**, aby nie było niejasności.

---

11. Tagowanie obrazu do Docker Huba

```
docker tag sebastian-nginx:1.0 docker.io/sebastian/nginx-course:1.0
```

Znaczenie:

- `docker.io` – Docker Hub
- `sebastian` – konto użytkownika
- `nginx-course` – repozytorium
- `1.0` – wersja

Tagowanie **nie wysyła obrazu**.

---

12. Logowanie do Docker Huba

```
docker login docker.io
```

---

13. Publikowanie obrazu

```
docker push docker.io/sebastian/nginx-course:1.0
```

Dopiero to polecenie:

- wysyła obraz do Docker Huba,
- czyni go dostępnym dla innych.

---

14. Uruchamianie obrazu z Docker Huba przez Compose

Inna osoba może użyć tego samego `docker-compose.yml`,
zmieniając tylko obraz:

```
services:
  nginx:
    image: docker.io/sebastian/nginx-course:1.0
    ports:
      - "8080:80"
```

I uruchomić:

```
docker compose up -d
```

---

15. Zadania ćwiczeniowe

Zadanie 1

Zmień treść strony i zbuduj wersję `1.1`.

Zadanie 2

Zmień port na `9090` w `docker-compose.yml`.

Zadanie 3

Opublikuj nową wersję obrazu w Docker Hubie.

Zadanie 4 (bonus)

Usuń lokalny obraz i uruchom aplikację **wyłącznie z Docker Huba**.

Podsumowanie

- Dockerfile **buduje obraz**
- Docker Compose **uruchamia kontenery**
- Obraz może być lokalny lub z registry
- `docker tag` przygotowuje obraz
- `docker push` publikuje obraz

WAŻNE: nazwa użytkownika w Docker Hubie

W przykładach w tym kursie używana jest nazwa użytkownika:

`sebastian`

Jest to **przykład dydaktyczny**.

W praktyce:

- **nie musisz** mieć loginu `sebastian`,
- **nie możesz** użyć loginu, który jest już zajęty,
- musisz użyć **własnego loginu zarejestrowanego w Docker Hubie**.

Jak dostosować przykłady do własnego konta

Jeżeli twój login w Docker Hubie to na przykład:

`jan_kowalski`

to **we wszystkich poleceniach**:

ZAMIENIASZ

```
sebastian
```

NA

```
jan_kowalski
```

Przykład – poprawne tagowanie z własnym loginem

```
docker tag sebastian-nginx:1.0 docker.io/jan_kowalski/nginx-course:1.0
```

Przykład – publikacja obrazu

```
docker push docker.io/jan_kowalski/nginx-course:1.0
```