

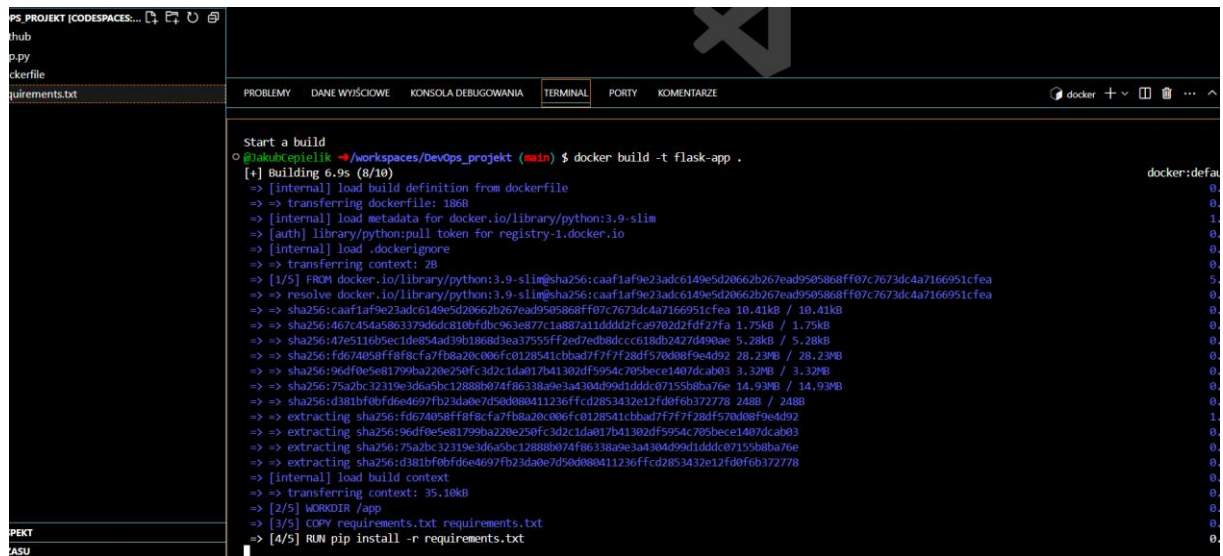
Dokumentacja opisuje proces tworzenia obrazu Dockera oraz jego uruchamiania z wykorzystaniem platformy GitHub.dev. Znajdziesz tutaj szczegóły dotyczące działania aplikacji, a także zrzuty ekranu ilustrujące kolejne kroki.

1. Budowa obrazu Dockera

Aplikacja została skonfigurowana w pliku Dockerfile. Obraz został zbudowany za pomocą polecenia:

docker build -t flask-app .

Poniżej znajduje się zrzut ekranu przedstawiający proces budowy obrazu Dockera.



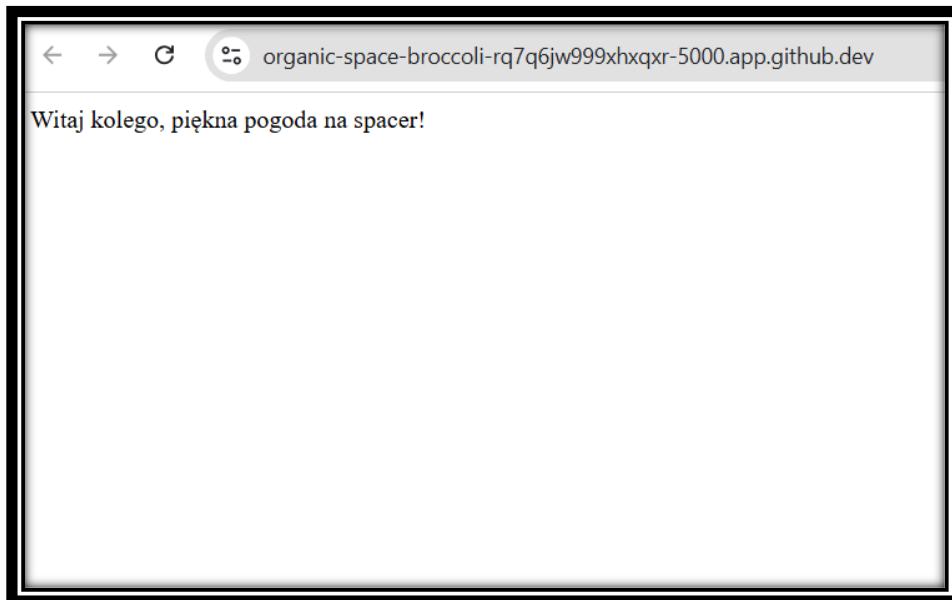
```
Start a build
@jalu@kali:~/workspaces/DevOps_projekt (main) $ docker build -t flask-app .
[*] Building 6.9s (8/10)
-> [internal] load build definition from dockerfile
-> => transferring dockerfile: 186B
-> [internal] load metadata for docker.io/library/python:3.9-slim
-> [auth] library/python:pull token for registry-1.docker.io
-> [internal] load .dockerignore
-> => transferring context: 2B
-> [1/5] FROM docker.io/library/python:3.9-slim@sha256:caaf1af9e23adc6149e5d20662b267ead9505868ff07c7673dc4a7166951cfea
-> resolve docker.io/library/python:3.9-slim@sha256:caaf1af9e23adc6149e5d20662b267ead9505868ff07c7673dc4a7166951cfea
-> sha256:caaf1af9e23adc6149e5d20662b267ead9505868ff07c7673dc4a7166951cfea 10.41kB / 10.41kB
-> sha256:467c454a5863379d6dc810b6b3e3e877c1a887a11ddddd2fca9702d2f2f7a 1.75kB / 1.75kB
-> sha256:47e5116b5ec1d854ad39b1868d3ea37555ff2ed7ed8dccc618db2427d490ae 5.28kB / 5.28kB
-> sha256:fd674058ff8f8cfa7fbaa20c006fc0128541cbbad7f7f728df570d08f9e4d92 28.23kB / 28.23kB
-> sha256:96df0e5e81799ba220e250fc3d2c1da017b41302df5954c705bec1407dcab03 3.32kB / 3.32kB
-> sha256:75a2bc32319e3d6a5bc12888074f06338a9e3a4304d99d1ddc071558ba70e 14.93kB / 14.93kB
-> sha256:d381bf0bf6e4697fb23da0e7d5e0880411236ffc2854132e12f40f0b372778 248B / 248B
-> extracting sha256:fd674058ff8f8cfa7fbaa20c006fc0128541cbbad7f7f728df570d08f9e4d92
-> extracting sha256:96df0e5e81799ba220e250fc3d2c1da017b41302df5954c705bec1407dcab03
-> extracting sha256:75a2bc32319e3d6a5bc12888074f06338a9e3a4304d99d1ddc071558ba70e
-> extracting sha256:d381bf0bf6e4697fb23da0e7d5e0880411236ffc2854132e12f40f0b372778
-> [internal] load build context
-> => transferring context: 35.10kB
-> [2/5] WORKDIR /app
-> [3/5] COPY requirements.txt requirements.txt
-> [4/5] RUN pip install -r requirements.txt
```

2. Uruchomienie aplikacji webowej

Aplikacja została uruchomiona w kontenerze Dockera za pomocą następującego polecenia:

docker run -p 5000:5000 flask-app

Poniżej znajduje się zrzut ekranu prezentujący działającą aplikację w przeglądarce internetowej.



3. Pull Request i scalanie zmian

Gałąź feature/change-content została utworzona w celu wprowadzenia zmian, w tym dodania pliku ci.yml. Po zakończeniu prac została scalona z główną gałęzią main. Poniżej zamieszczono zrzut ekranu dokumentujący Pull Request.

4. Wykorzystanie Pythona i narzędzi wspierających

Python to uniwersalny język programowania wysokiego poziomu, szeroko stosowany w tworzeniu aplikacji webowych, analizie danych czy automatyzacji procesów. W projekcie wykorzystano framework Flask, który pozwala na szybkie tworzenie aplikacji webowych.

5. Zależności projektu

Zależności wymagane do działania aplikacji zostały zdefiniowane w pliku requirements.txt i obejmują:

Flask==2.0.3: Framework do budowy aplikacji webowych.

Werkzeug==2.0.3: Narzędzie wspierające Flask, odpowiedzialne za routing i debugowanie.

Instalacja zależności

Aby zainstalować wszystkie wymagane biblioteki, należy użyć polecenia:

pip install -r requirements.txt

Polecenie to instaluje pakiety zgodnie z wersjami wskazanymi w pliku requirements.txt, co zapewnia kompatybilność używanych bibliotek z projektem.

6. Repozytorium projektu

Autor: Jakub Cepielik

Kod studenta: 14242

GitHub Repo: https://github.com/JakubCepielik/DevOps_projekt