# Міністерство освіти і науки України Національний технічний університет України

**«Київський політехнічний інститут імені Ігоря Сікорського»**

**Факультет інформатики та обчислювальної техніки**  **Кафедра обчислювальної техніки**

**Лабораторна робота №3**

з дисципліни

« Методології та технології розробки програмного забезпечення »

Виконав: Перевірив:

Студент 2-го курсу групи ІМ-11 Ковальчук О. М.

Алексєєнко Данііл Вадимович

Київ 2023

# Python

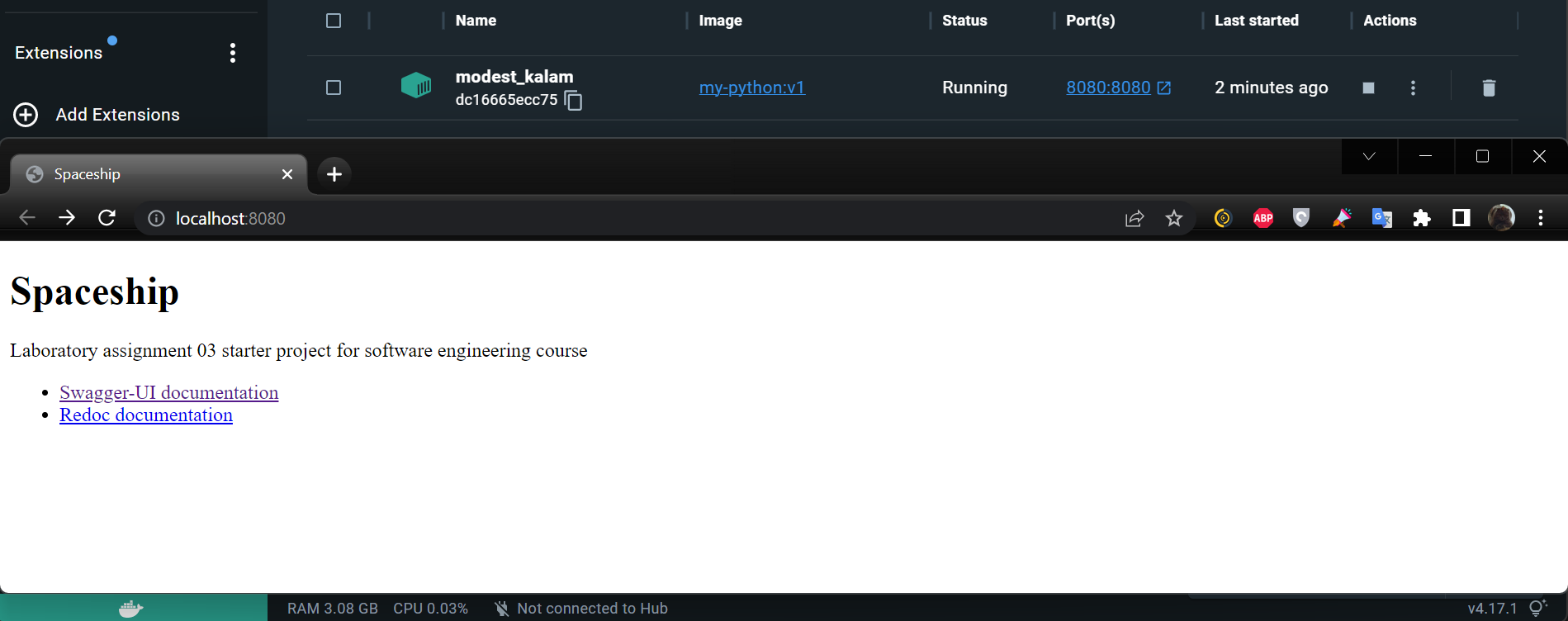
### 1. Перша реалізація

**Команди для створення докеру:**

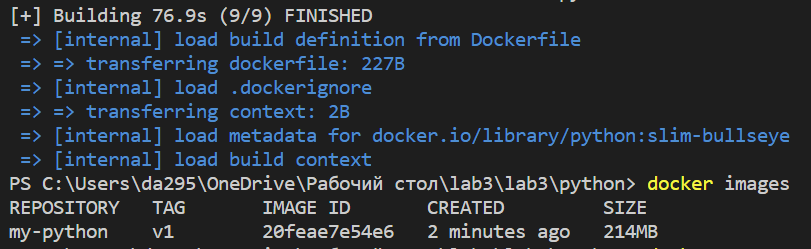
1) docker build . -t my-python:v1 //для старту сбірки

2) docker images //для перегляду сбірки

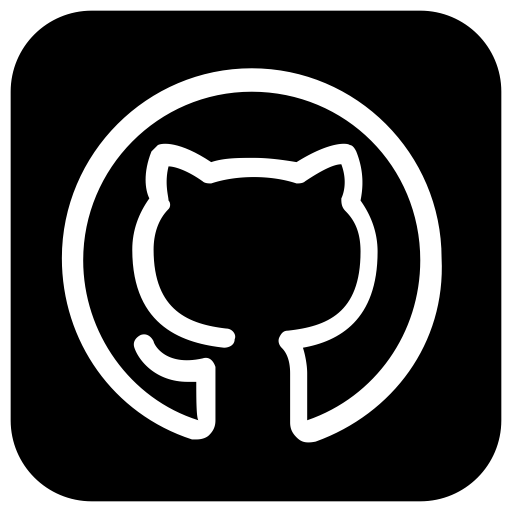
3) docker run -p 8080:8080 --rm my-python:v1 //для запуску сбірки



**Результат:**



Час: **79,6с**; Розмір: **214MB**;

[](https://github.com/Tutinichic/lab3/commit/183c04e266d66fb0470584072887f92a7eb3cc3b)Посилання на комміт:

### 2. Друга реалізація

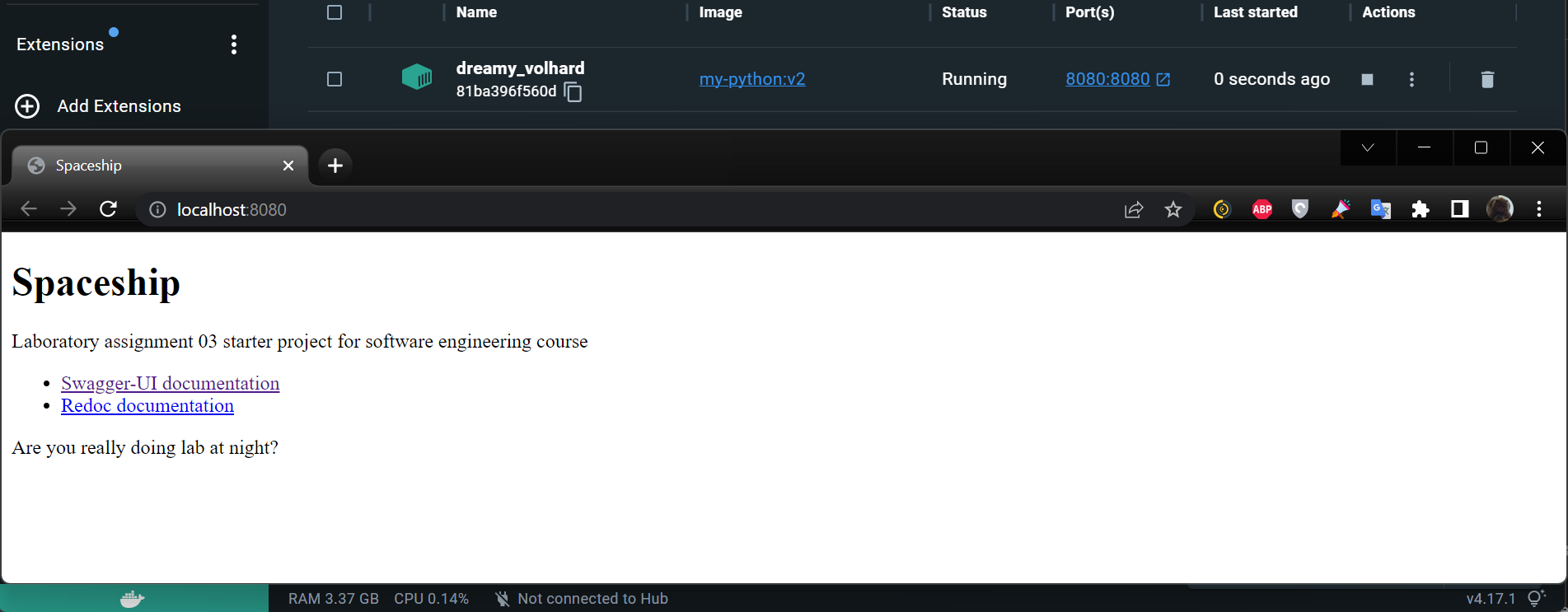
Змінив файл index.html.

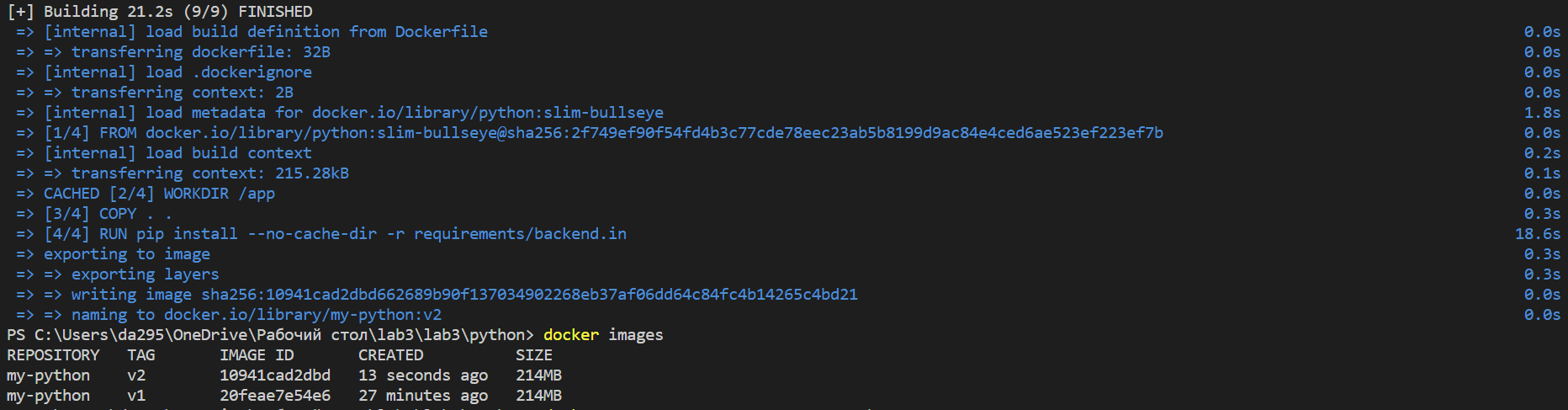
**Команди для створення докеру:**

1) docker build . -t my-python:v2 //для старту сбірки

2) docker images //для перегляду сбірки

3) docker run -p 8080:8080 --rm my-python:v2 //для запуску сбірки

 **Результат:**



Час: **21,2с**; Розмір: **214MB**;

### 3. Третя реалізація

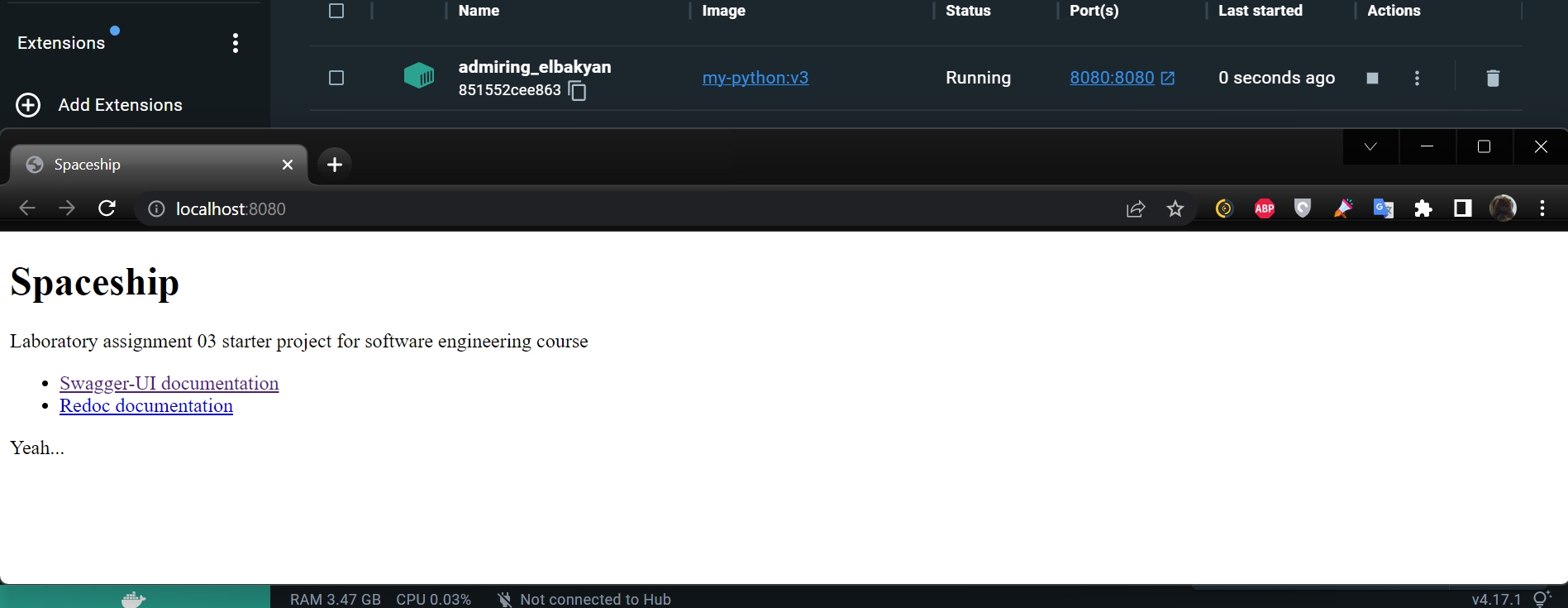
Змінив файли index.html та Dockerfile.

**Команди для створення докеру:**

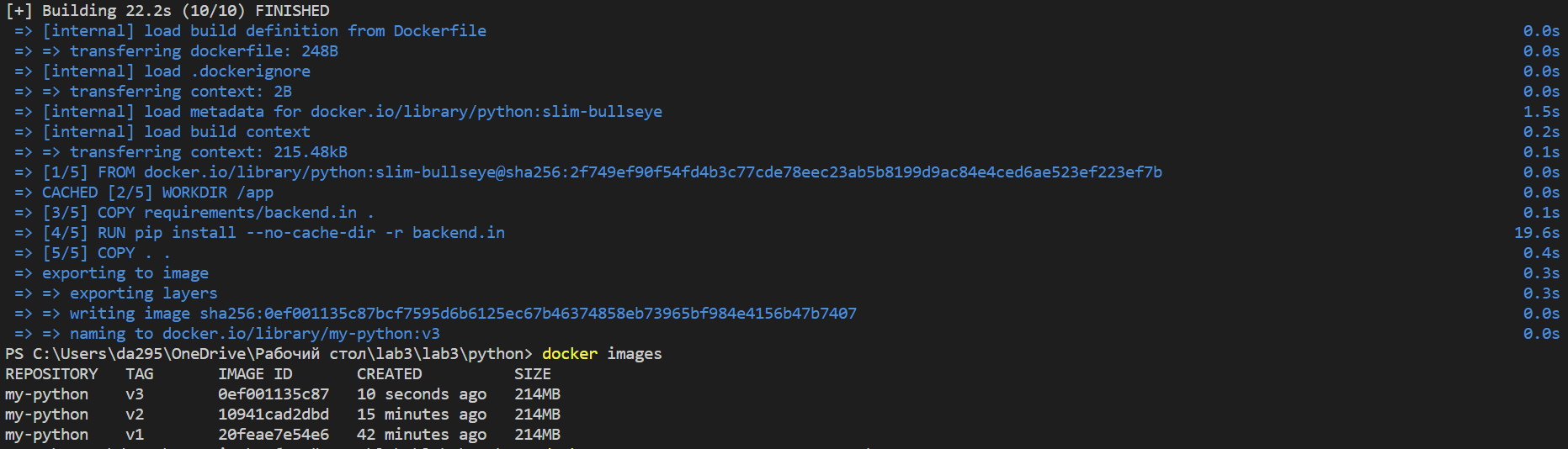
1) docker build . -t my-python:v3 //для старту сбірки

2) docker images //для перегляду сбірки

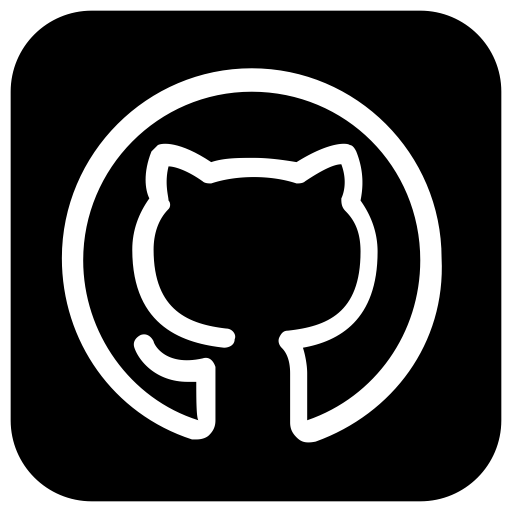
3) docker run -p 8080:8080 --rm my-python:v3 //для запуску сбірки



**Результат:**



Час: **22,2с**; Розмір: **214MB**;

[](https://github.com/Tutinichic/lab3/commit/d9a956910ee5f02a7c9f8ff2c35ee602f274b0e5)Посилання на комміт:

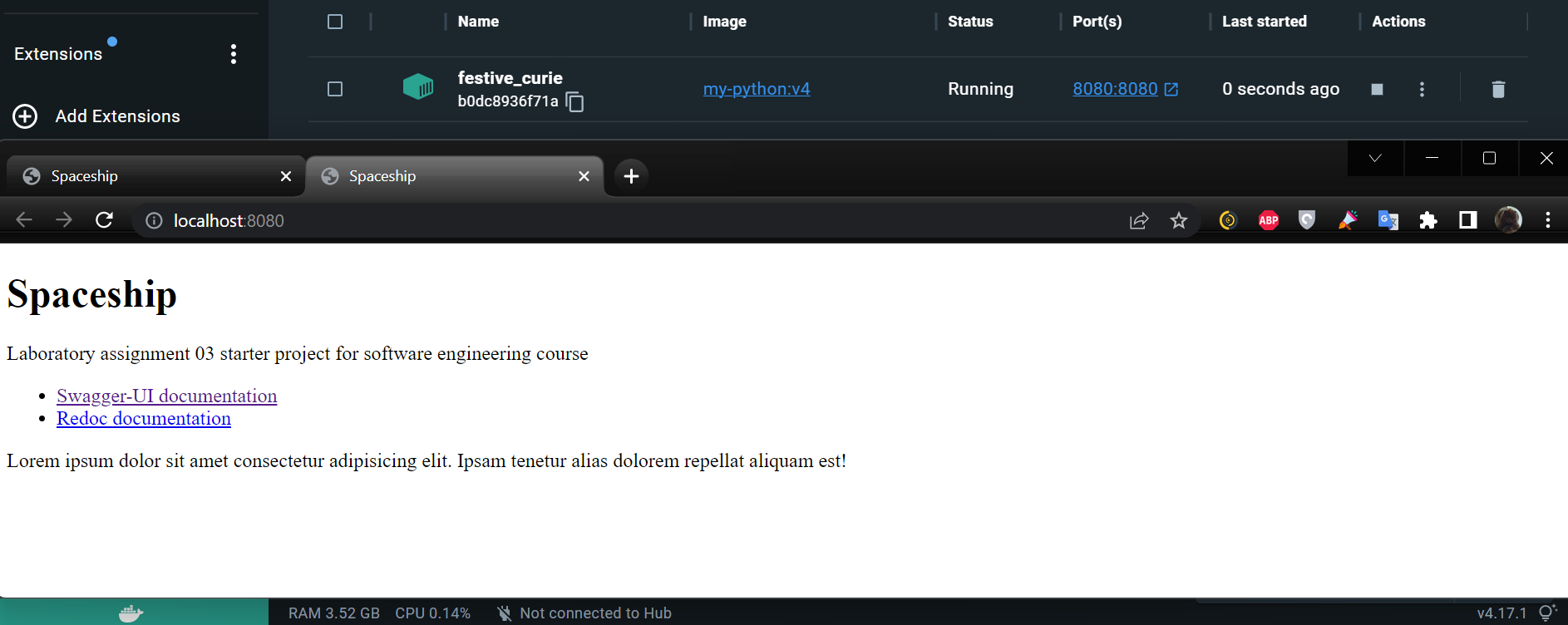
**Перевірка нового докеру:**

**Команди для створення докеру:**

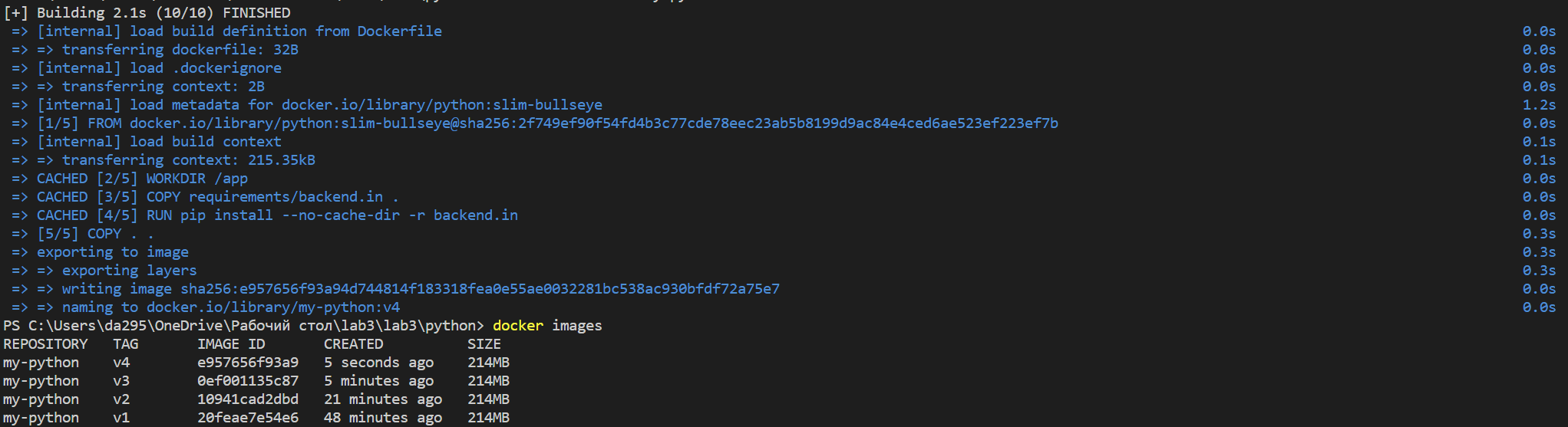
1) docker build . -t my-python:v4 //для старту сбірки

2) docker images //для перегляду сбірки

3) docker run -p 8080:8080 --rm my-python:v4 //для запуску сбірки



**Результат:**



Час: **2,1с**; Розмір: **214MB**;

### 4. Четверта реалізація

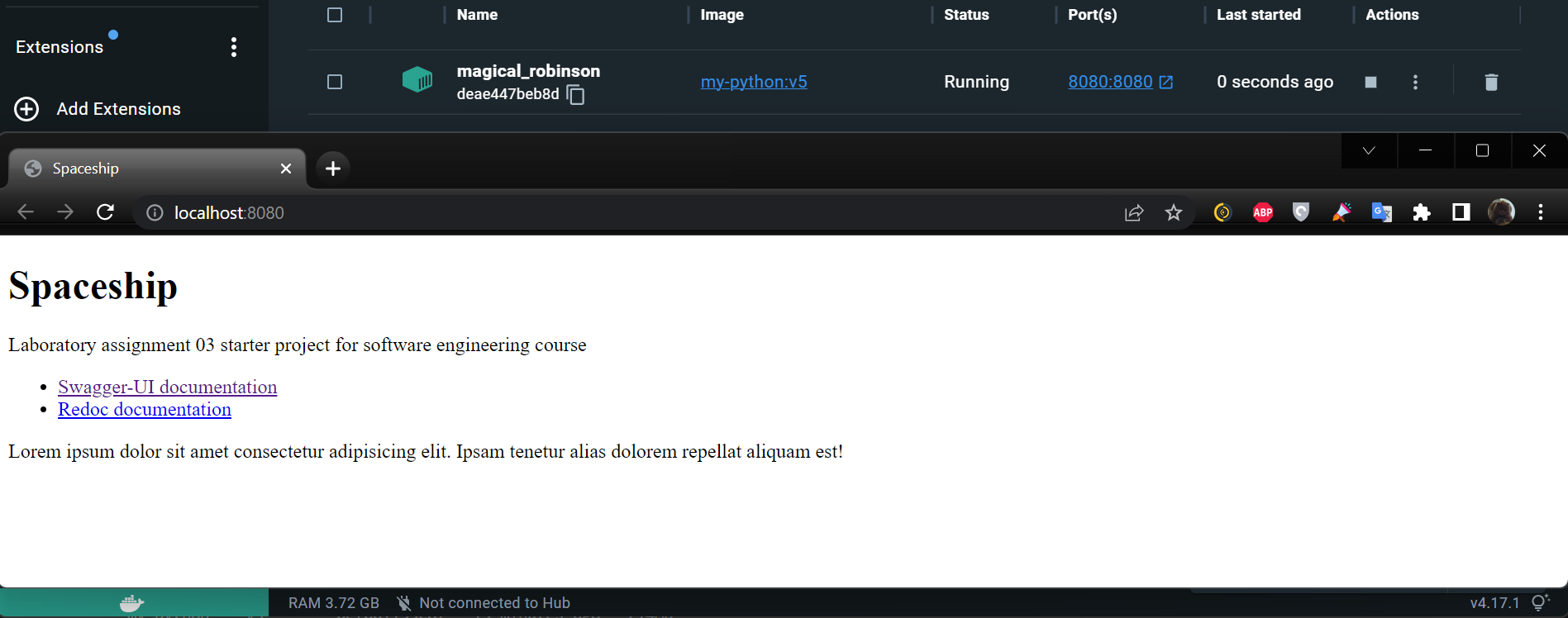
Змінив базу сбірки на python:alpine.

**Команди для створення докеру:**

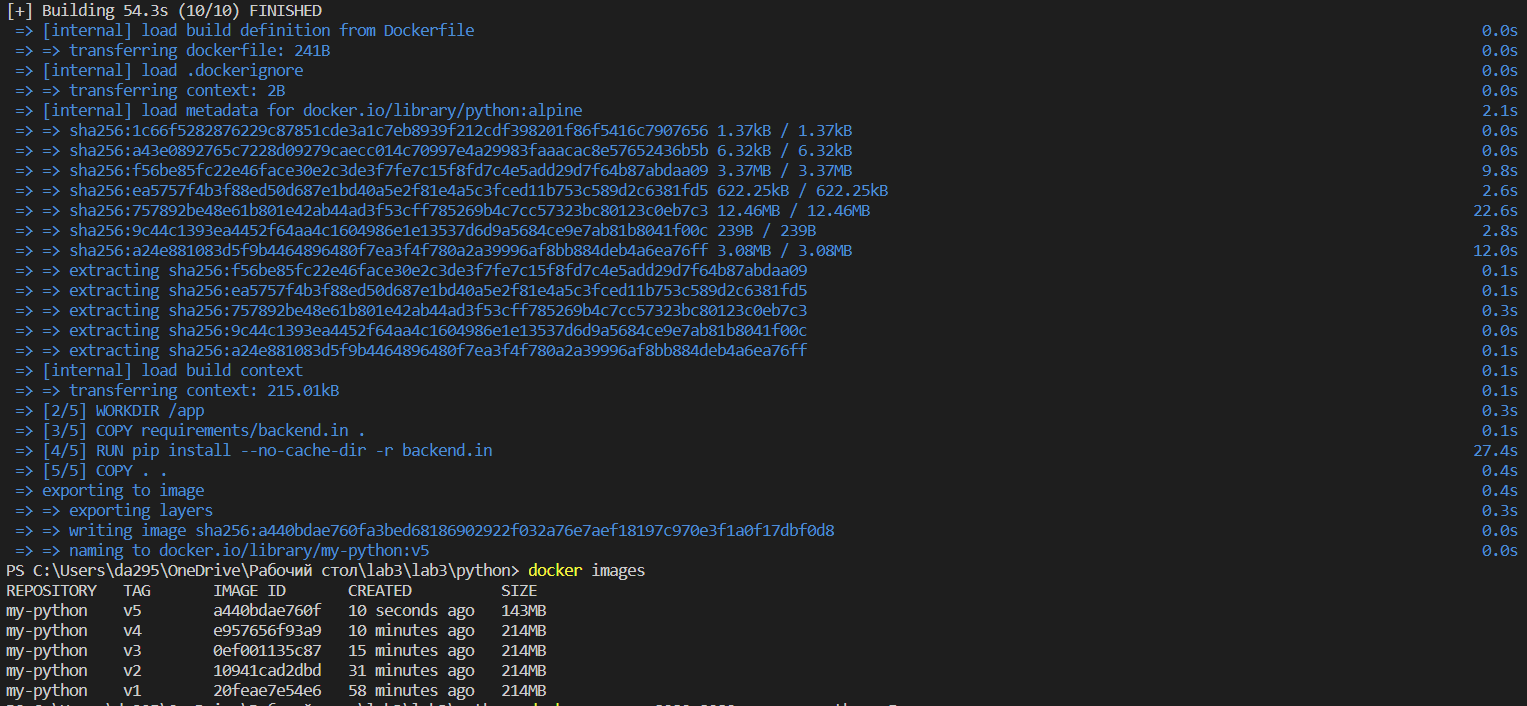
1) docker build . -t my-python:v5 //для старту сбірки

2) docker images //для перегляду сбірки

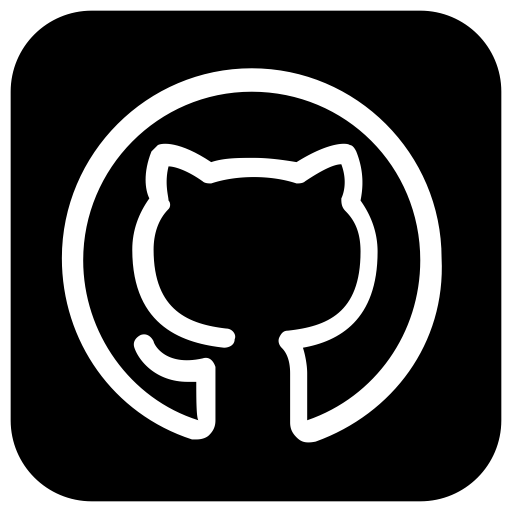
3) docker run -p 8080:8080 --rm my-python:v5 //для запуску сбірки



**Результат:**

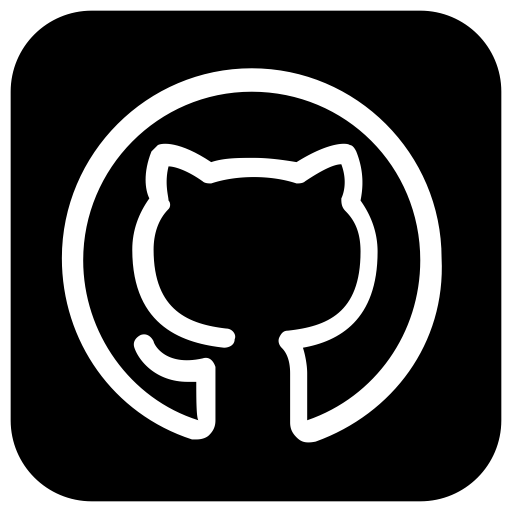


Час: **54,3с**; Розмір: **143MB**;

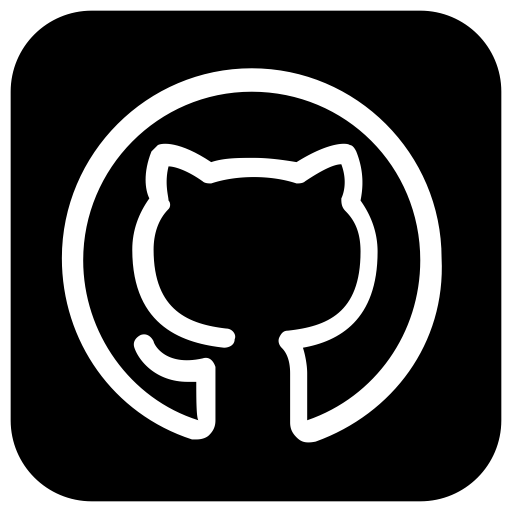
[](https://github.com/Tutinichic/lab3/commit/f3586f63065c22c906bc8dbce926f2648a943602)Посилання на комміт:

### 5. П’ята реалізація

**Змінив Dockerfile та backend.in**

[](https://github.com/Tutinichic/lab3/commit/6ccef4eb9da8b0a825300ddece0d873a6c5f8e36)Посилання на комміт:

**Додав матрицю.**

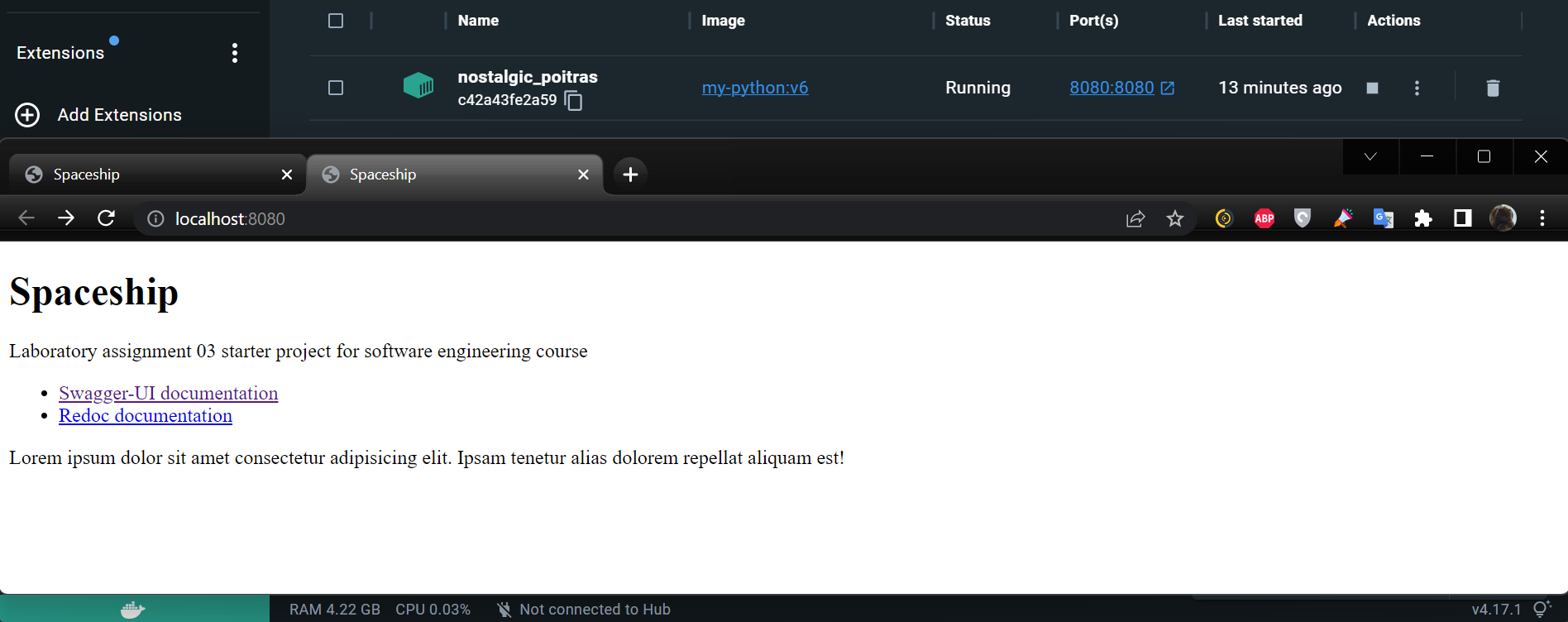
[](https://github.com/Tutinichic/lab3/commit/344db59fc58f9f4ce9e4df3906fea1c35f05631e)Посилання на комміт:

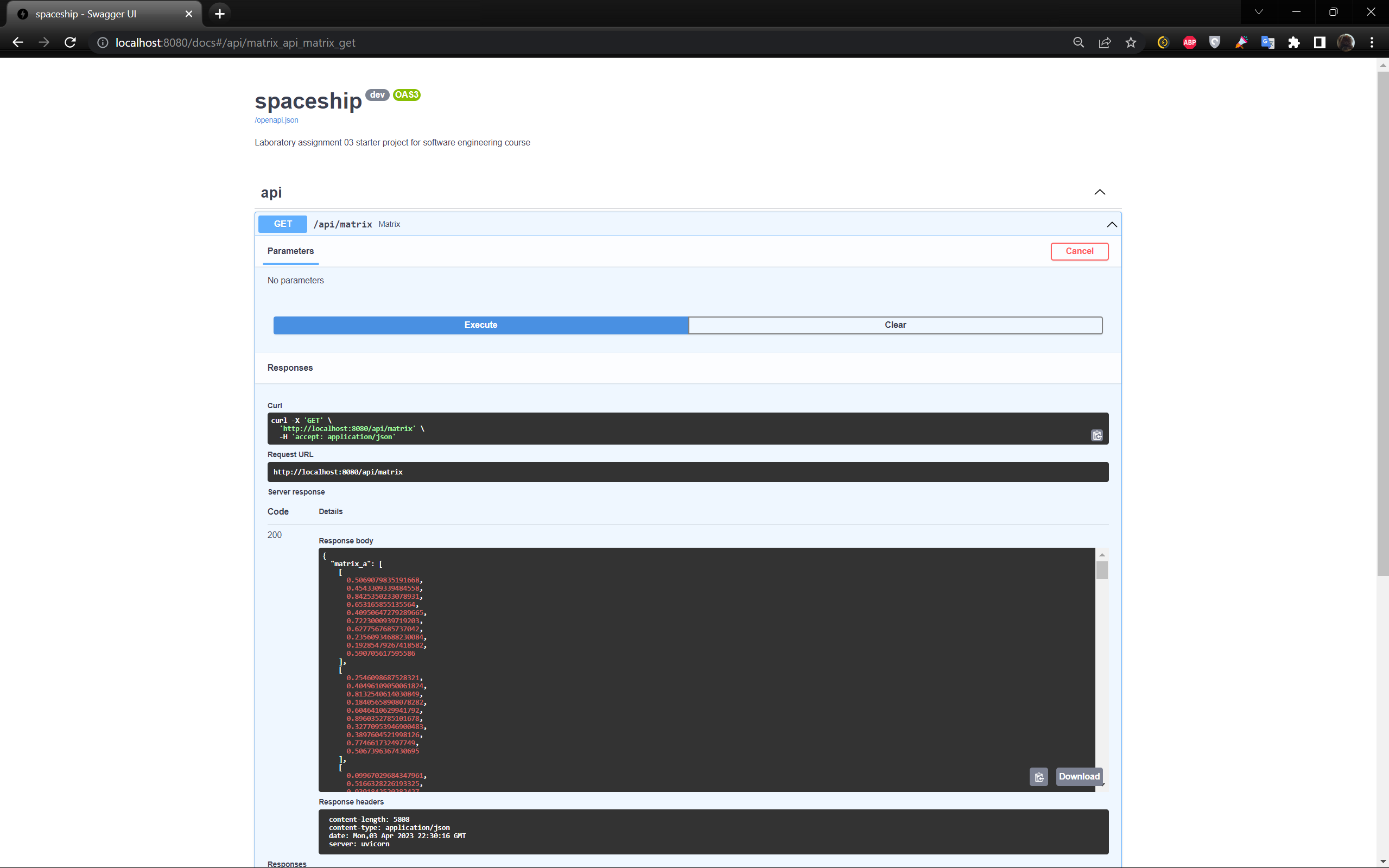
**Команди для створення докеру на alpine:**

1) docker build . -t my-python:v6 //для старту сбірки

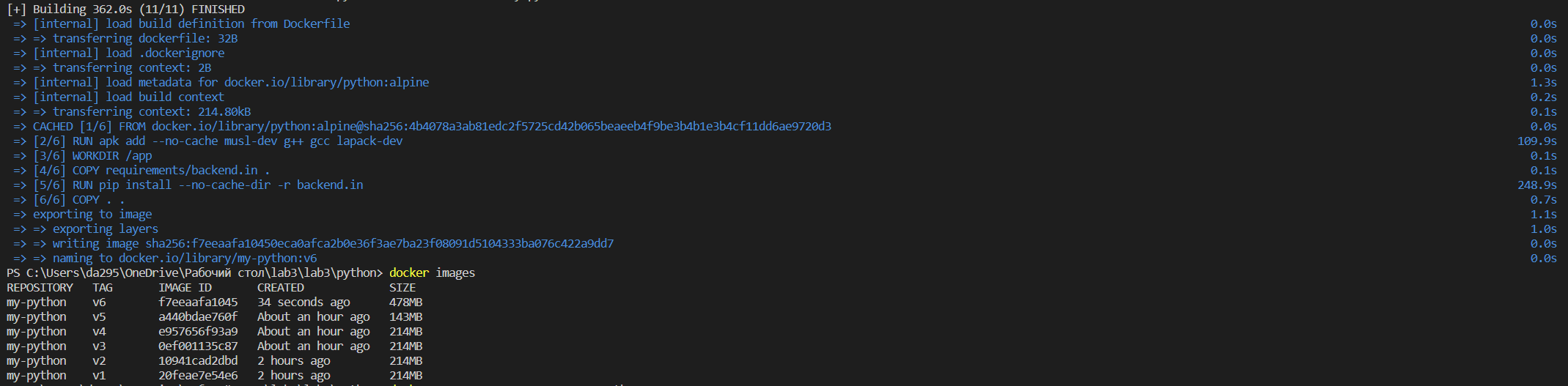
2) docker images //для перегляду сбірки

3) docker run -p 8080:8080 --rm my-python:v6 //для запуску сбірки





**Результат:**

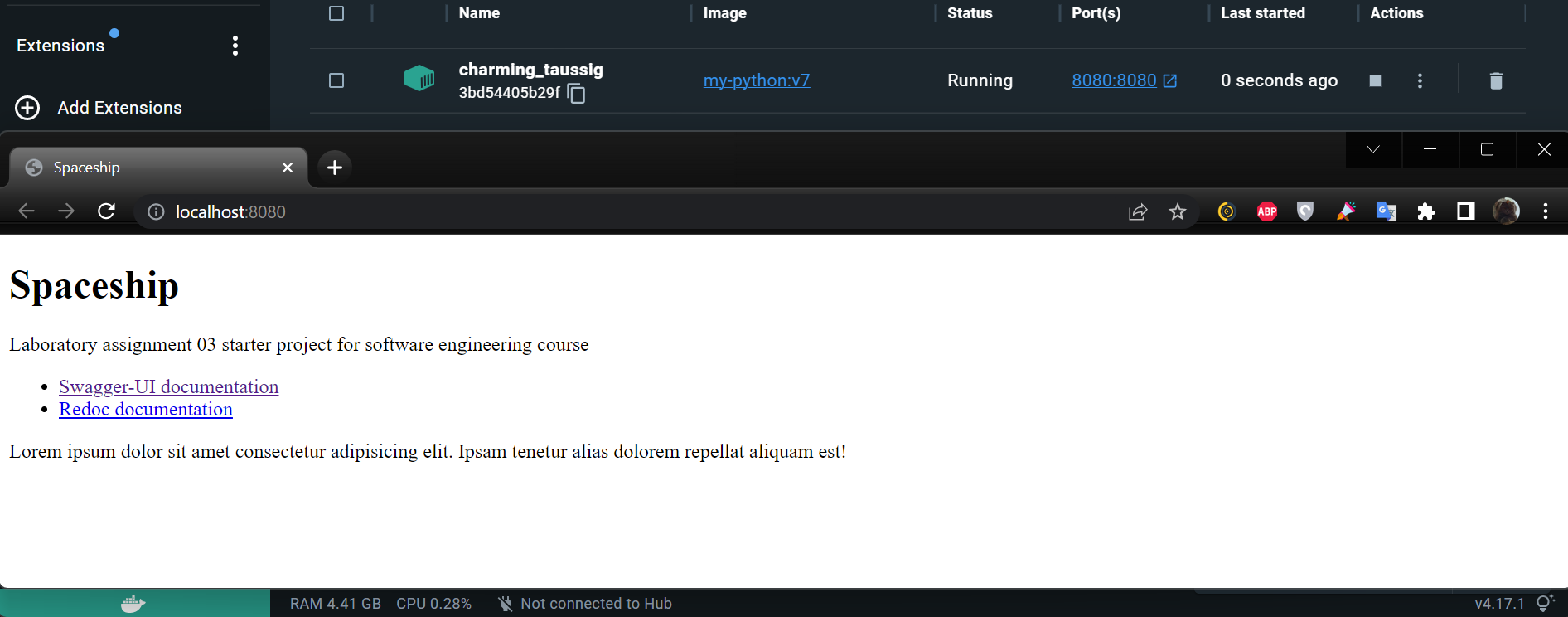
 Час: **362с**; Розмір: **478MB**;

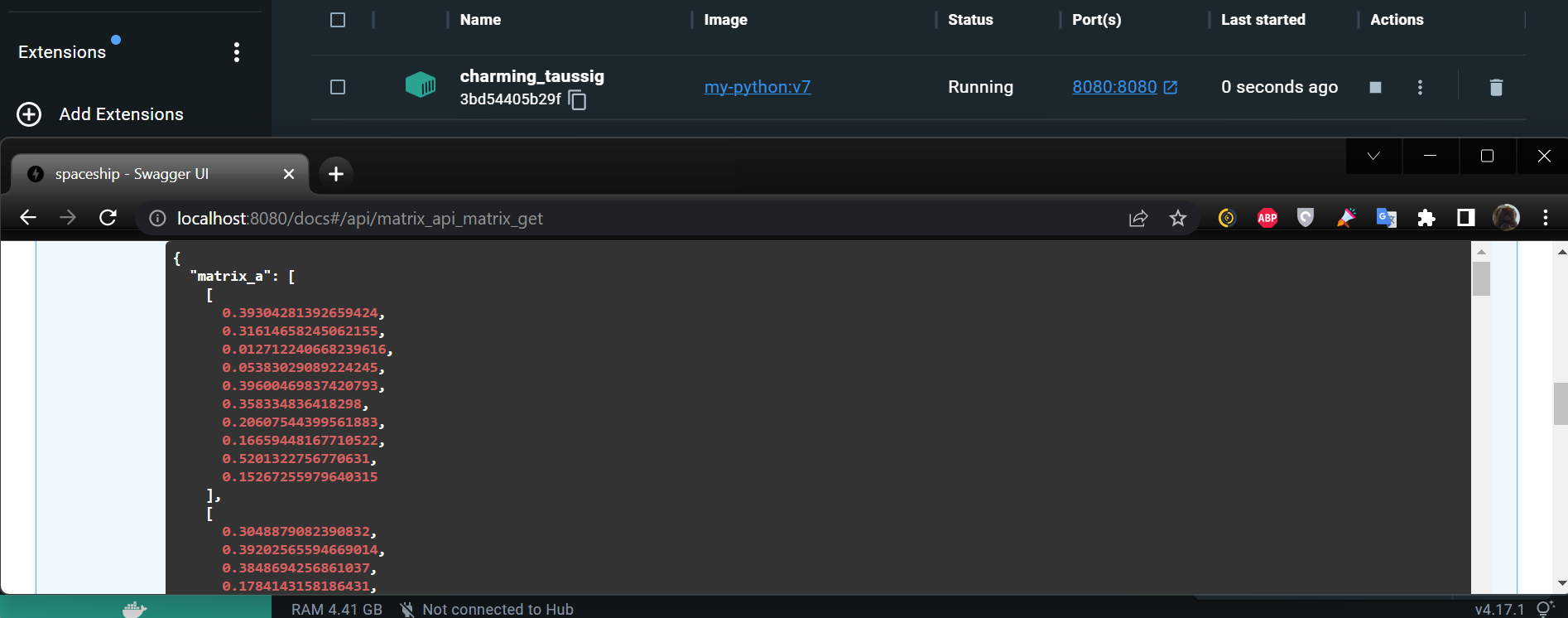
**Команди для створення докеру на debian:**

1) docker build . -t my-python:v7 //для старту сбірки

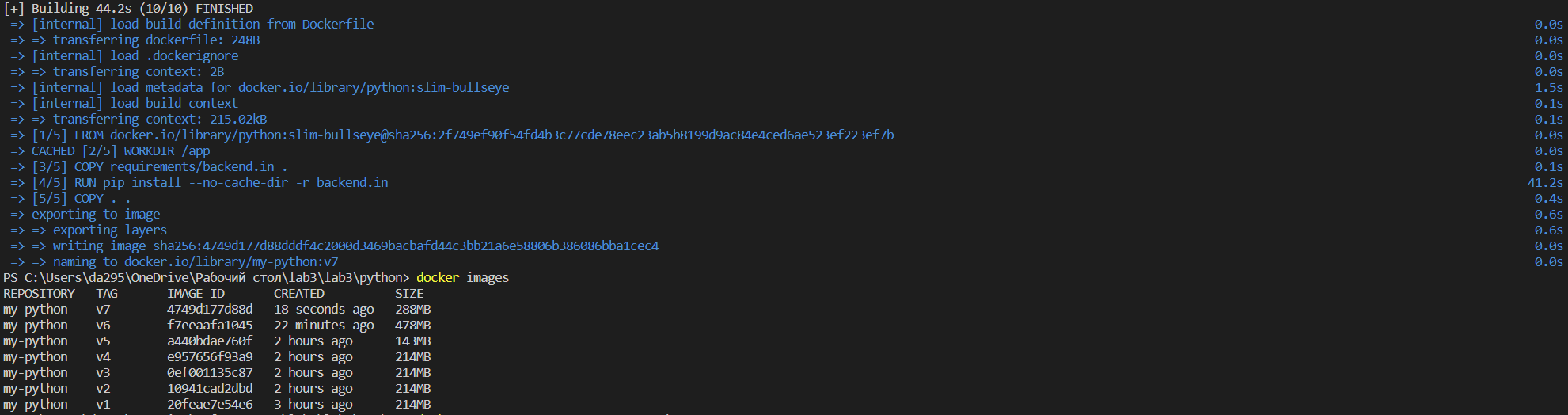
2) docker images //для перегляду сбірки

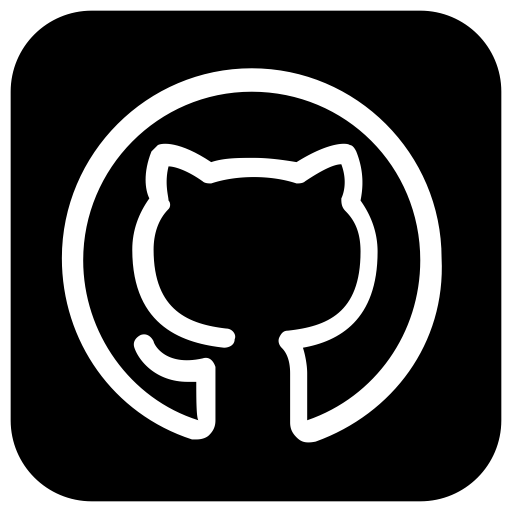
3) docker run -p 8080:8080 --rm my-python:v7 //для запуску сбірки





**Результат:**

Час: **44,2с**; Розмір: **288MB**;

[](https://github.com/Tutinichic/lab3/commit/da3f13a519c8ce02acef549b4beedb655e37e3db)Посилання на комміт:

На debian сбірка білдиться швидше, бо не скачує додаткові інструменти з якими працює alpine з numpy.

# Golang

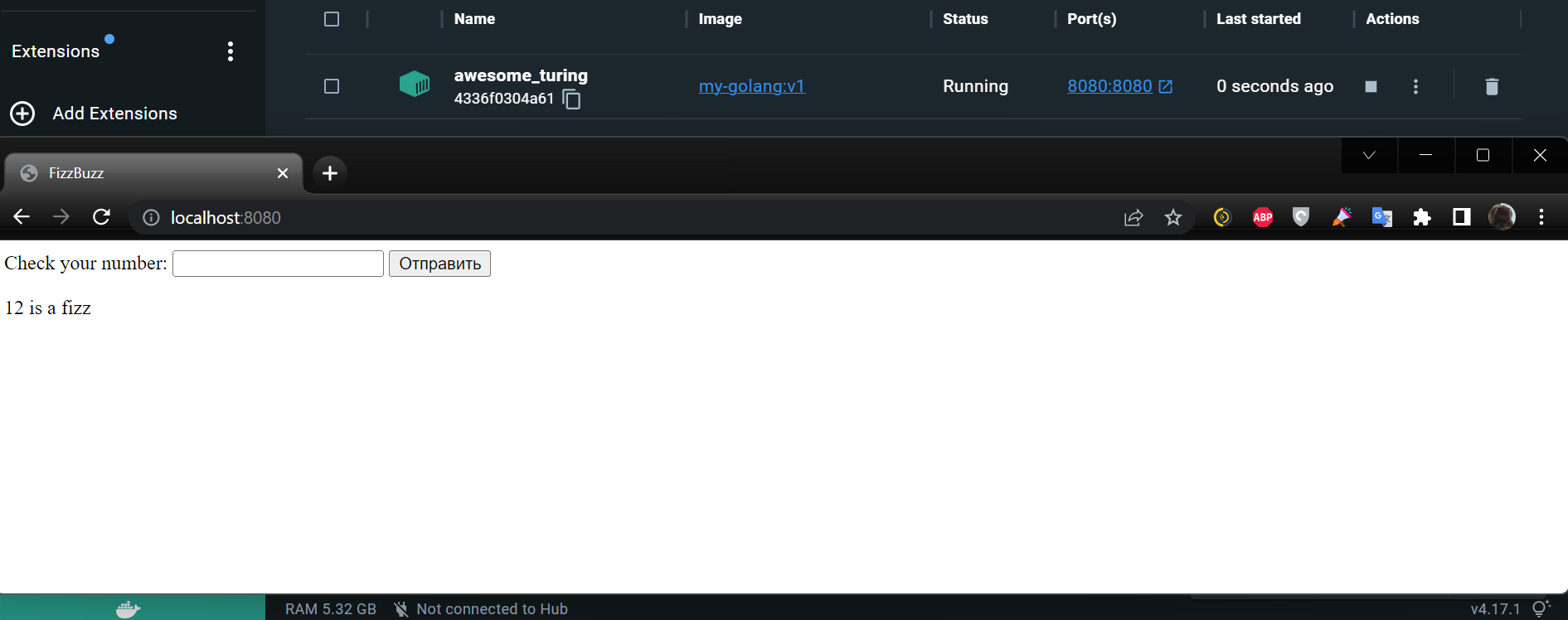
### 1. Перша реалізація

**Команди для створення докеру:**

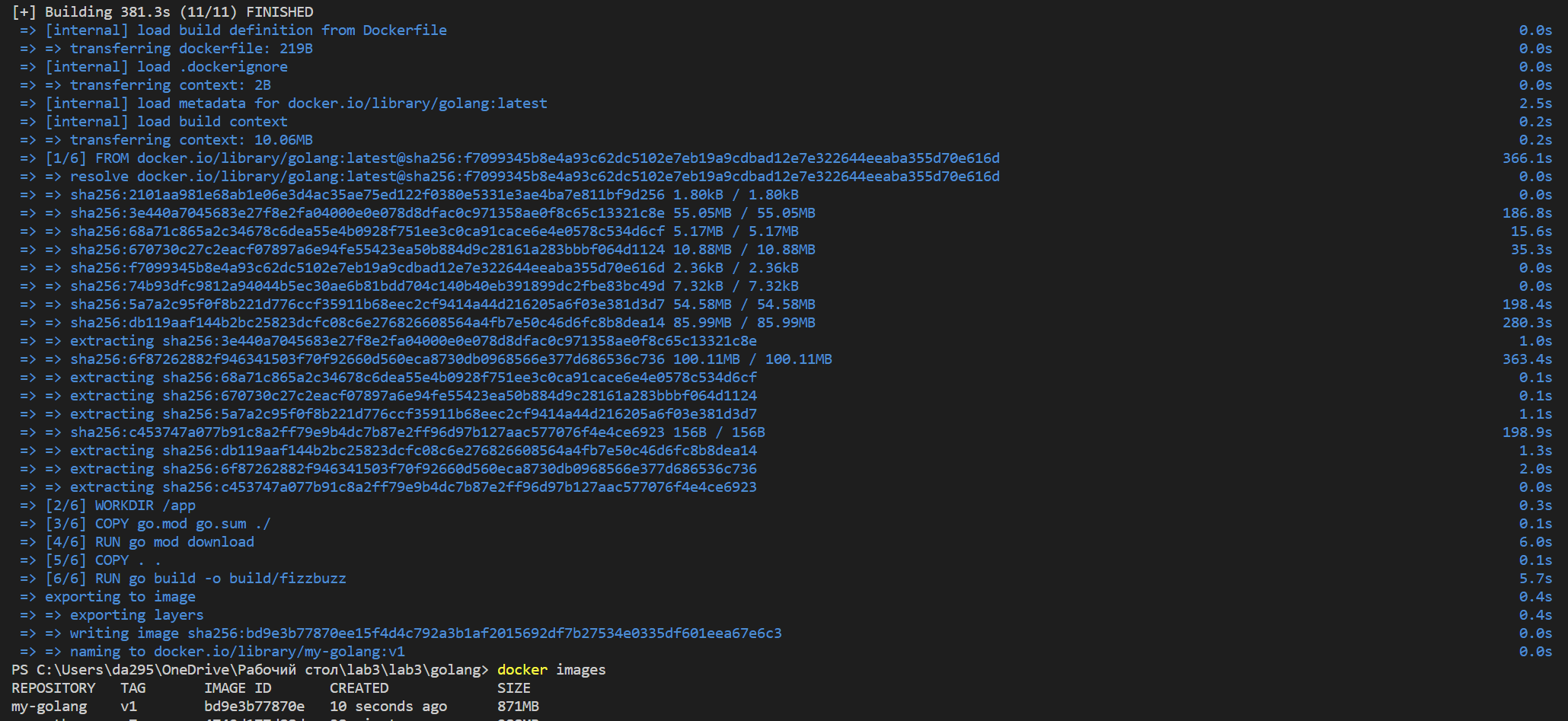
1) docker build . -t my-golang:v1 //для старту сбірки

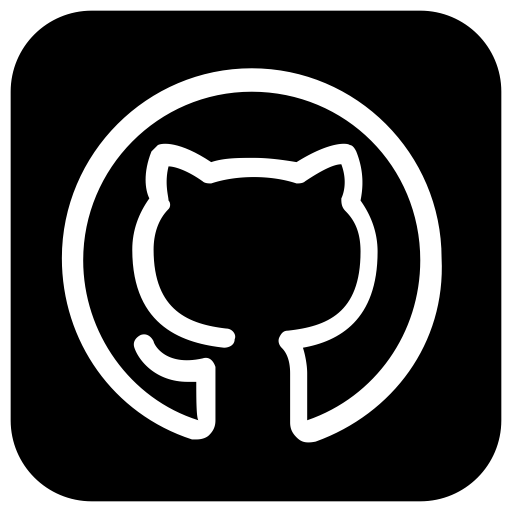
2) docker images //для перегляду сбірки

3) docker run -p 8080:8080 --rm my-golang:v1 //для запуску сбірки



**Результат:**

Час: **381,3с**; Розмір: **871MB**;

[](https://github.com/Tutinichic/lab3/commit/52ea26b5903048548376f6925af7e8d70f28b4e0)Посилання на комміт:

### 2. Друга реалізація

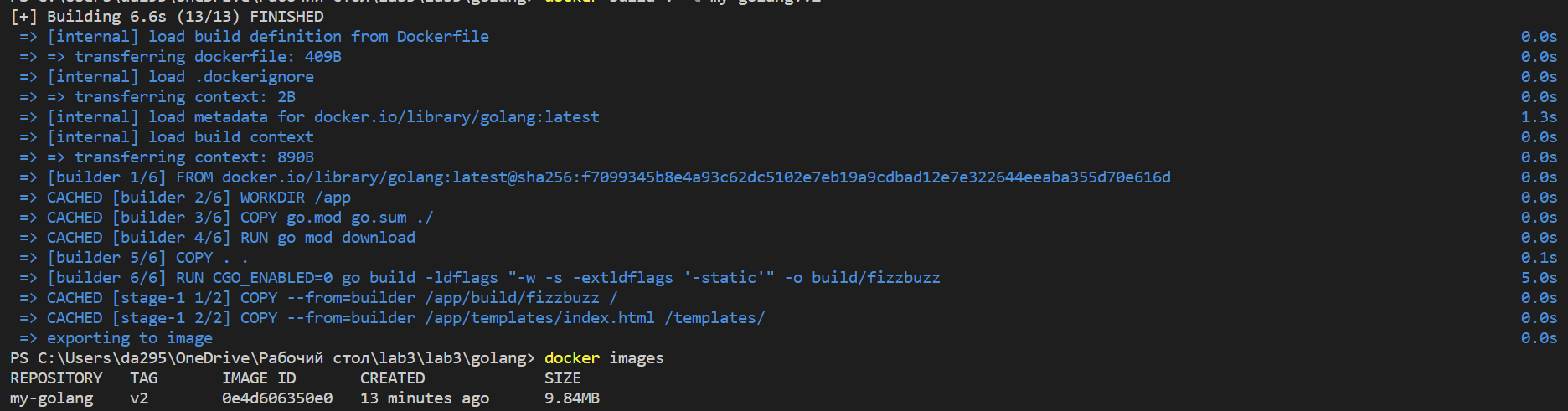
**Команди для створення докеру:**

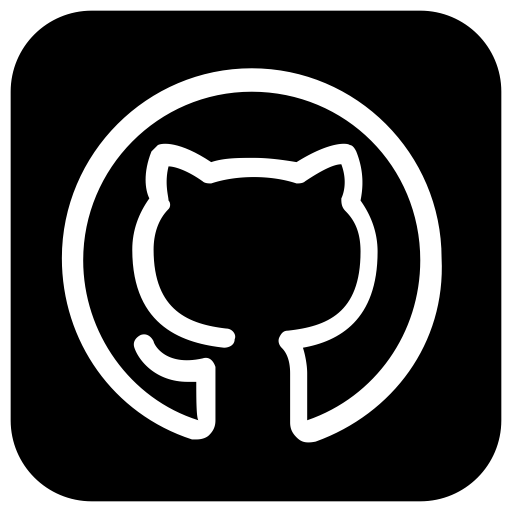
1) docker build . -t my-golang:v2 //для старту сбірки

2) docker images //для перегляду сбірки

3) docker run -p 8080:8080 --rm my-golang:v2 //для запуску сбірки

**Результат:**

Час: **6,6с**; Розмір: **9,84MB**;

[](https://github.com/Tutinichic/lab3/commit/7616a51d4157368a39b33f8991b64a171dc56fcc)Посилання на комміт:

Для такого білду файлів достатньо, бо нам потрібна лише сторінка та код розрахунку.

Цим образом дуже зручно користуватися, бо ми отримуємо доволі малий розмір білду та велику швидкість його складання та розгортання. І все це завдяки тому, що ми не додали до образу зайві залежності та файли, які нам не потрібні для запуску програми.

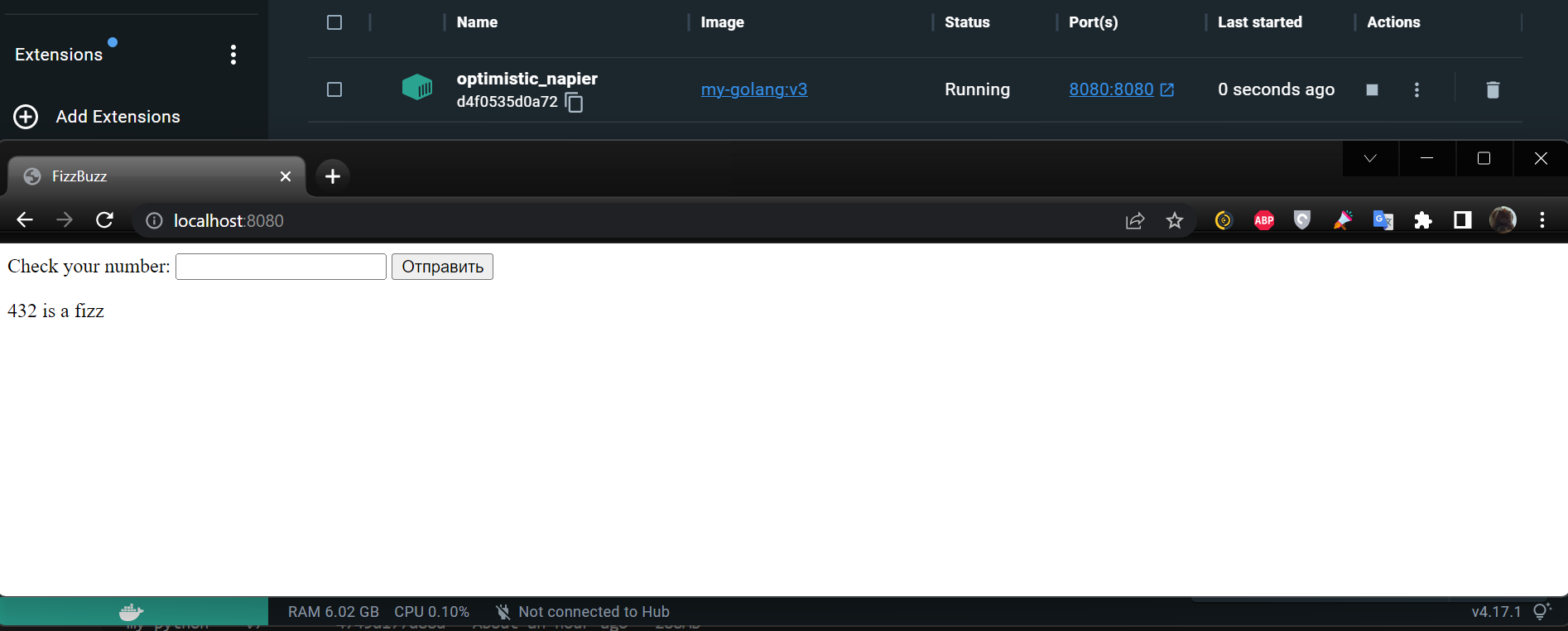
### 3. Третя реалізація

**Команди для створення докеру:**

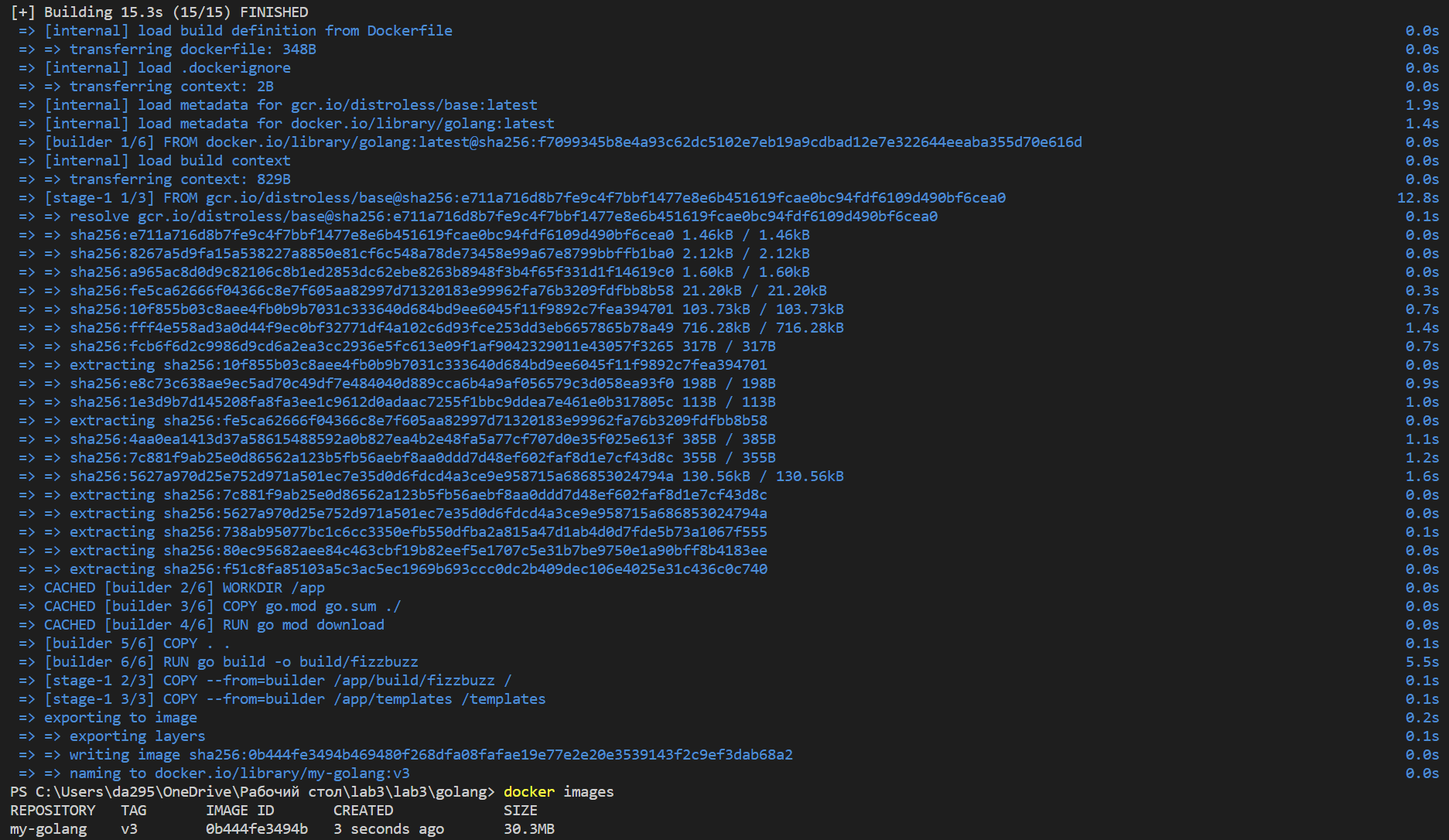
1) docker build . -t my-golang:v3 //для старту сбірки

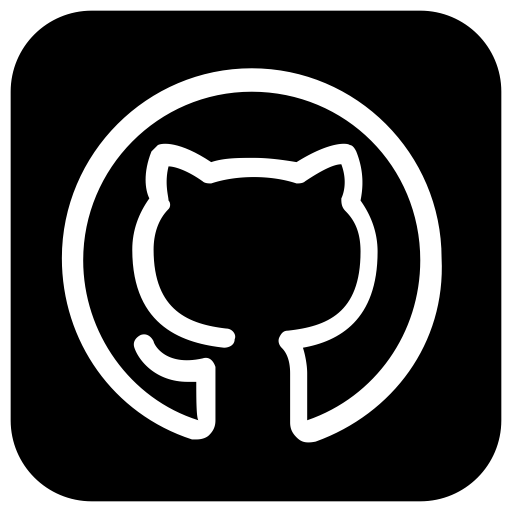
2) docker images //для перегляду сбірки

3) docker run -p 8080:8080 --rm my-golang:v3 //для запуску сбірки



**Результат:**

Час: **15,3с**; Розмір: **30,3MB**;

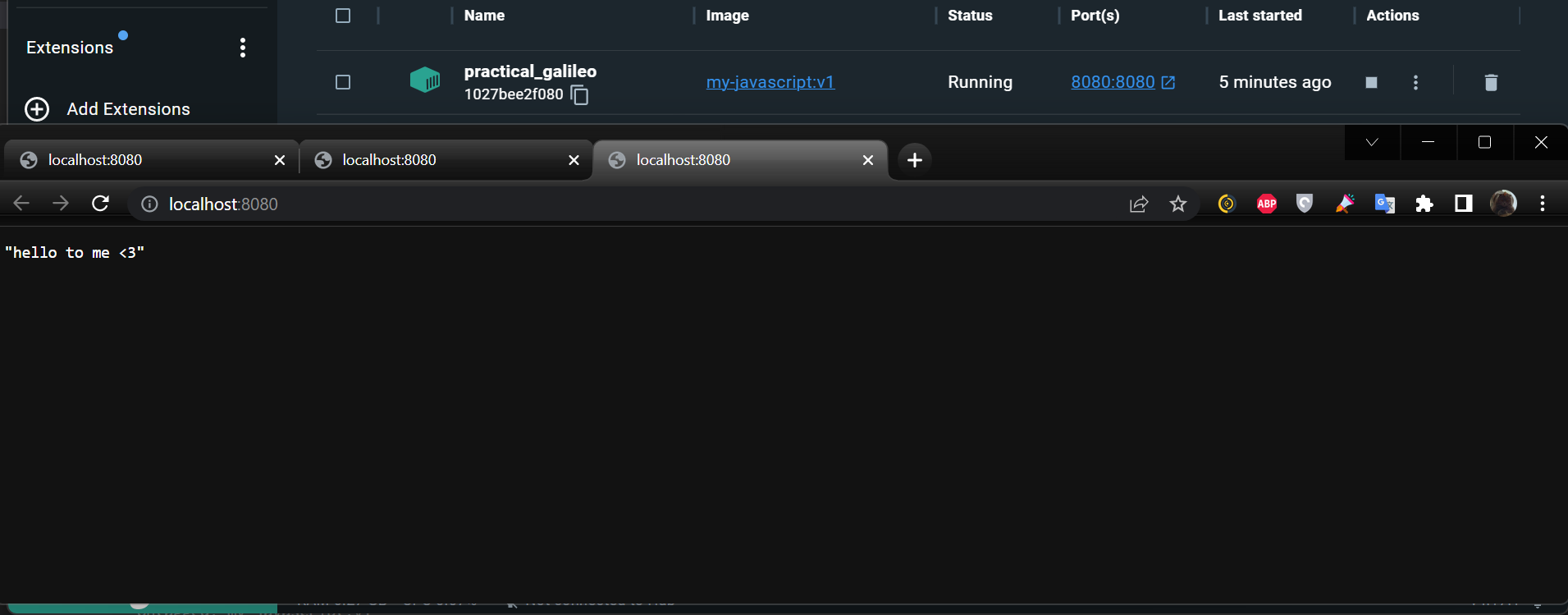
[](https://github.com/Tutinichic/lab3/commit/b7e9e969c11b83a56509f677ab8d0f3db2427e18)Посилання на комміт:

Для Скретч білду потрібно власноруч встановлювати всі залежності та бібліотеки для образу, це потребує часу та знань, проте на виході користувач отримує максимально оптимізованний та малий розмір у образу, який не буде мати нічого непотрібного.

Дистролес образи вже будут містити у собі всі необідні залежності та бібліотеки для мов програмування та фреймворків. Також для створення образу не потрібно прописувати всі бібліотеки та ін. для запуску програми, це дуже сберігає час.

Як на мене, дистролесс образ значно краще скретча, хоча і має більший розмір.

# JavaScript



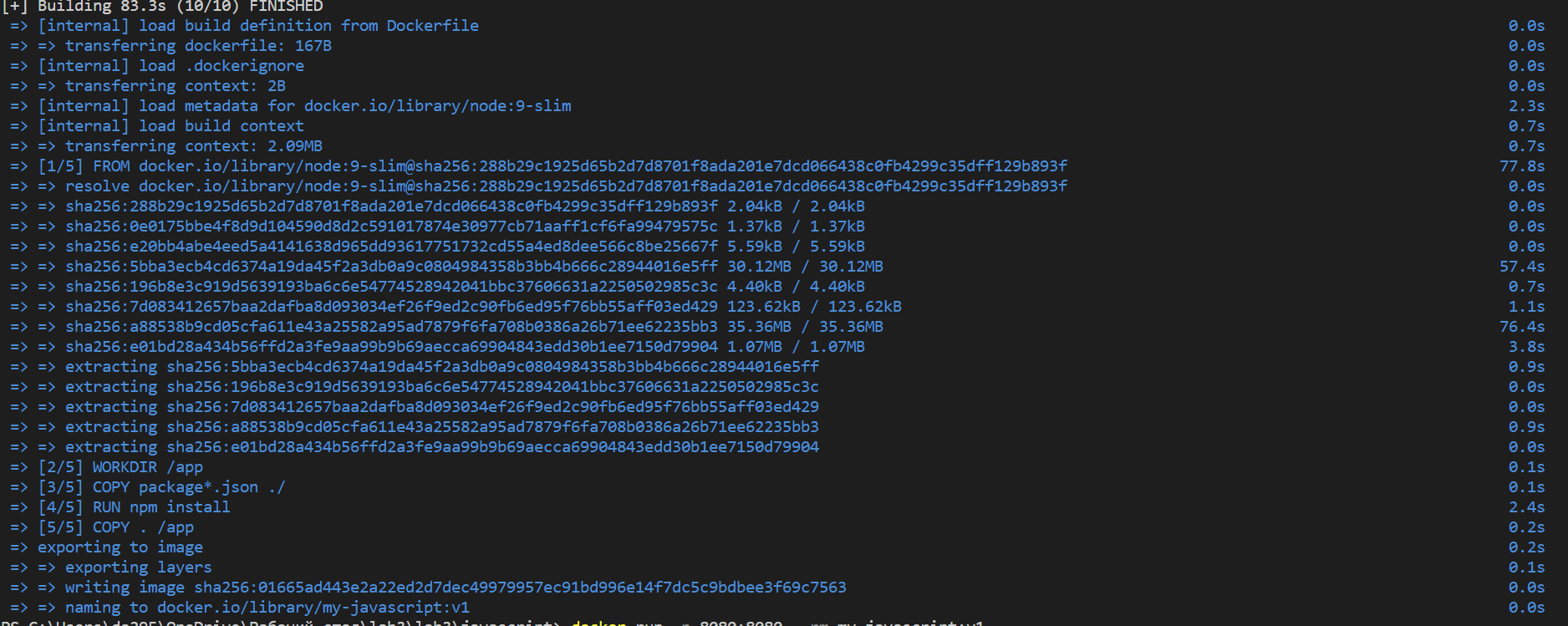
**Команди для створення докеру:**

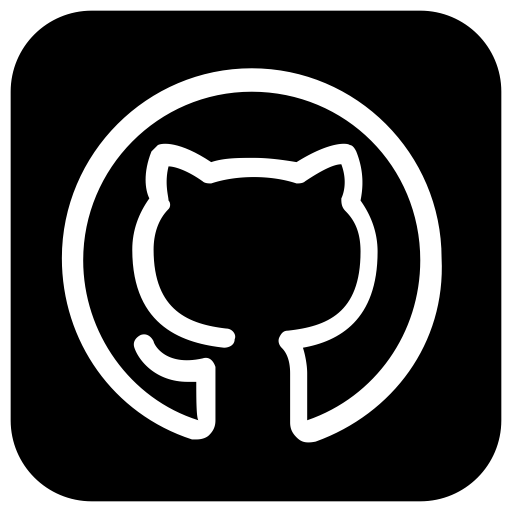
1) docker build . -t my-javascript:v1 //для старту сбірки

2) docker images //для перегляду сбірки

3) docker run -p 8080:8080 --rm my-javascript:v1 //для запуску сбірки

**Результат:**

Час: **83с**; Розмір: **185MB**;

[](https://github.com/Tutinichic/lab3/commit/bee2dd1e0e7aa7f0ebdd409973431ef2b5b9e248)Посилання на комміт:

**Висновки:**

З докером раніше не працював. Від лабороторної отримав загальне уявлення про докер і контейнеризацію, навчися робити найпростіші образи.

Отримав різні дані: різний час сбірки та розмір. Для мене улюбленим залишився скретч, бо хоча він і має такий мінус, як настроювання власноруч всіх компонентів та бібліотек, проте для користувача цей образ буде найшвидшив та буде займати мало пам’яті.

Труднощів з лабороторною особливо не було, але багато часо витратив на ознайомлення з докером та різними типами білдів та їх особливостей.

Якщо результатом проекту є виконуваний файл, то гарним рішенням буде багатоетапна збірка образу, в якій спочатку встановлюються залежності та вихідний код компілюється у бінарний файл, scratch у таких випадках зробить максимально оптимізований та легкий образ, але потребує налаштування з нуля, а у distroless вже є всі необхідні компоненти для запуску програми.