|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | |  |  |  | | МИНОБРНАУКИ РОССИИ | | | | Федеральное государственное бюджетное образовательное учреждение  высшего образования  **«МИРЭА – Российский технологический университет»**  **РТУ МИРЭА** | | |   Институт Информационных технологий | |
|  | |
| Кафедра Инструментального и прикладного программного обеспечения (ИиППО) | |
|  | |
|  | |

|  |  |
| --- | --- |
| **ОТЧЕТ**  **ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 9**  «Межконтейнерное взаимодействие docker-контейнеров через localhost» | |
| **по дисциплине** | |
| **«**Настройка и администрирование сервисного программного обеспечения**»** | |
|  | |
| Выполнил студент группы ИКБО-20-21 | Сидоров С.Д. |
| Принял преподаватель | Хозяинов А.В. |

|  |  |  |
| --- | --- | --- |
| Лабораторная работа выполнена | «\_\_»\_\_\_\_\_\_\_202\_ г. | *(подпись студента)* |
|  |  |  |
| «Зачтено» | «\_\_»\_\_\_\_\_\_\_202\_ г. | *(подпись руководителя)* |

Москва 2023

Цель работы

Docker создает сеть по умолчанию, в которой запущены все контейнеры, и задает имя сети для каждого контейнера, используя имя контейнера.

Для межконтейнерного взаимодействия можно использовать брокер сообщений Apache ActiveMQ. Если есть mq для ActiveMQ, то можно использовать так: tcp://mq:61616 (или любой другой настроенный протокол/порт,) из других контейнеров, чтобы подключиться к нему.

При этом не нужно устанавливать параметр --net если не нужно создавать определенную сеть для использования определенных контейнеров.

По умолчанию в Docker включено межконтейнерное взаимодействие, это означает, что все контейнеры могут взаимодействовать между собой (используя сеть docker0). Эта возможность может быть отключена путём запуска Docker сервиса с флагом --icc=false.

Выполнение работы

Создать два докер-контейнера, которые будут одновременно "слушать" localhost на разных портах. Пользователь делает запрос к контейнеру №1. Далее контейнер №1 делает запрос к контейнеру №2. Контейнер №2 обработает запрос, передаст результат контейнеру №1. Контейнер №1 передаст результат пользователю. API может быть реализовано на любой технологии. В итоге необходимо настроить систему из нескольких контейнеров, связанных друг с другом. Результат представлен на рисунках 1, 2, 3 и листингах

Листинг 1 – Серверная часть контейнера 1

|  |
| --- |
| from flask import Flask  import requests  app = Flask(\_\_name\_\_)  @app.route('/api2')  def api():  response = requests.get('http://container2:8001/api')  return response.json()  if \_\_name\_\_ == '\_\_main\_\_':  app.run(host='0.0.0.0', port=8000) |

Листинг 2 – Серверная часть контейнера 2

|  |
| --- |
| from flask import Flask, jsonify  app = Flask(\_\_name\_\_)  @app.route('/api')  def api():  return jsonify({'message': 'Hello from container2!'})  if \_\_name\_\_ == '\_\_main\_\_':  app.run(host='0.0.0.0', port=8001) |

Листинг 3 – Dockerfile контейнера 1

|  |
| --- |
| FROM python:latest  WORKDIR /app  ADD 1.py /app  ADD requirements.txt /app  RUN pip install -r requirements.txt  CMD ["python", "1.py"] |

Листинг 4 – Dockerfile контейнера 2

|  |
| --- |
| FROM python:latest  WORKDIR /app  ADD 2.py /app  ADD requirements.txt /app  RUN pip install -r requirements.txt  CMD ["python", "2.py"] |

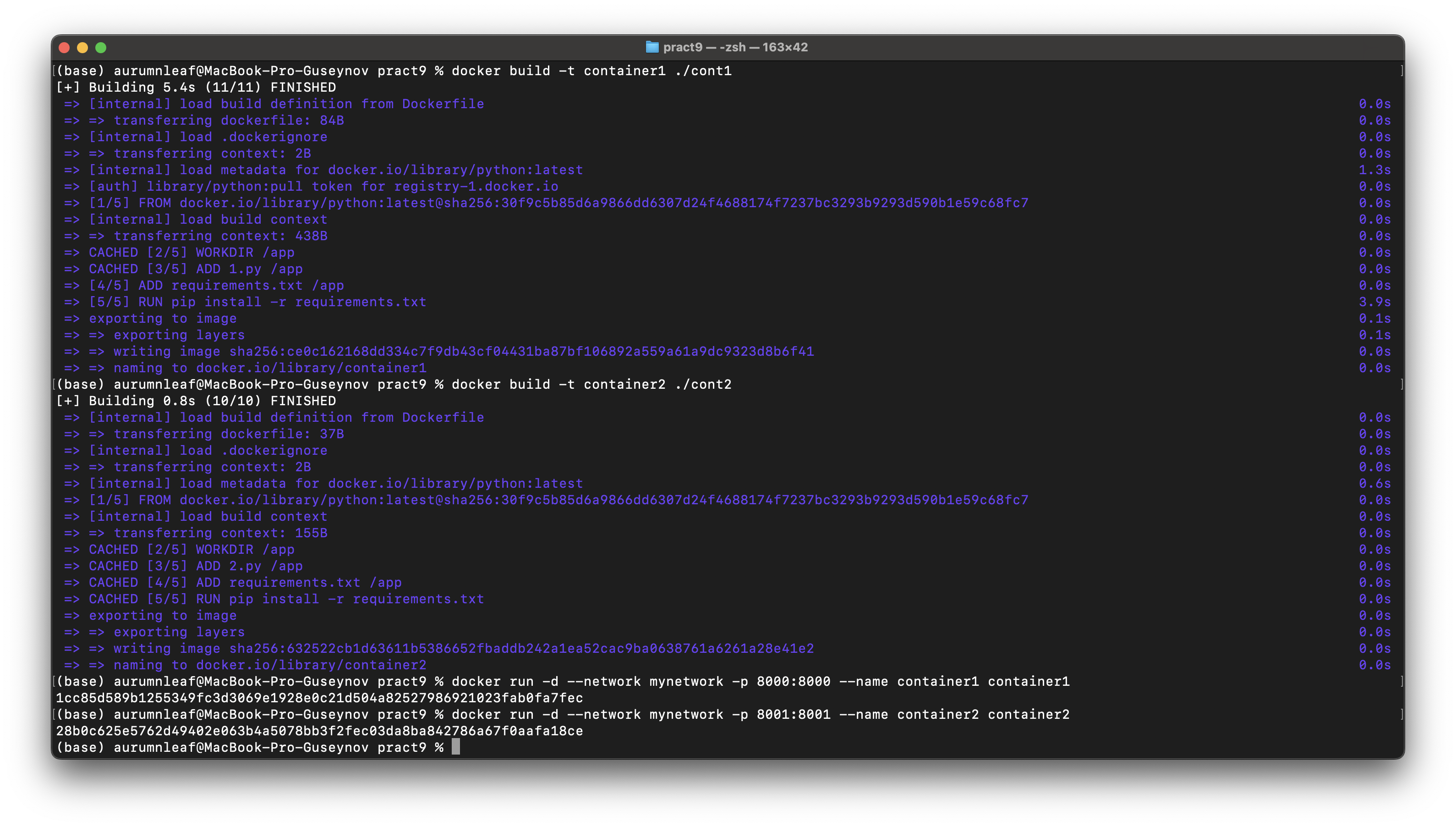


Рисунок 1 - Сборка контейнера 1

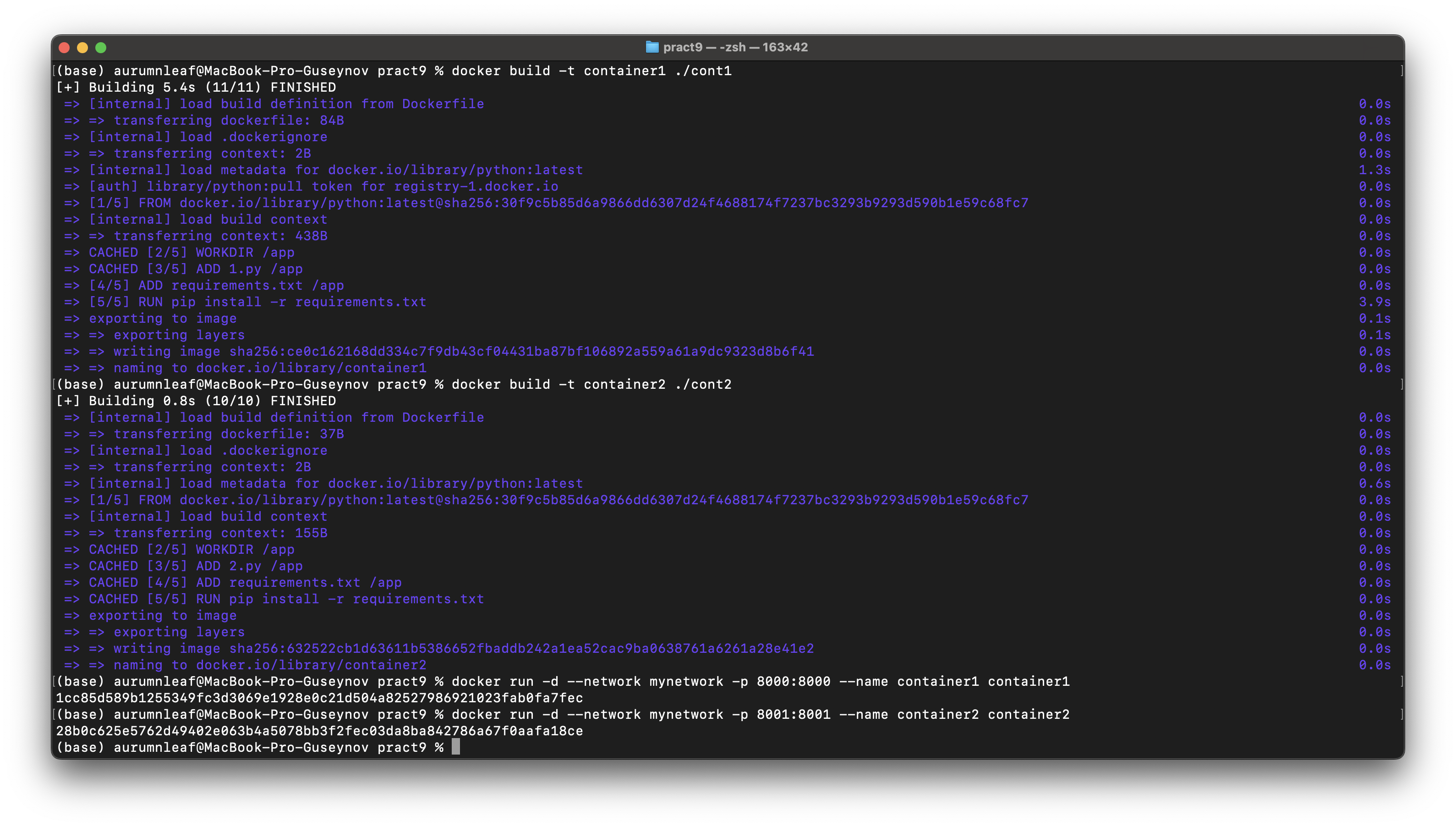


Рисунок 2 – Сборка контейнера 2

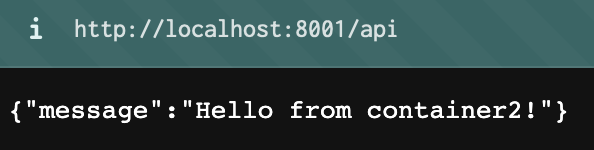


Рисунок 3 – Страница контейнера 1

**ВЫВОД**

В ходе данной работы был получен опыт создания докер контейнеров и отправки запросов из одного контейнера в другой.