Министерство науки и высшего образования Российской Федерации

федеральное государственное бюджетное образовательное учреждение высшего образования

«Российский экономический университет имени Г.В. Плеханова»

**МОСКОВСКИЙ ПРИБОРОСТРОИТЕЛЬНЫЙ ТЕХНИКУМ**

специальность 09.02.07 «Информационные системы и программирование»

Квалификация: Программист

ПРАКТИЧЕСКИЕ РАБОТЫ

ПО МДК 04.02 «Обеспечение качества функционирования компьютерных систем»

|  |  |
| --- | --- |
| Выполнил студент  группы П50-3-22  Пушкин Илья Александрович | Проверил преподаватель  \_\_\_\_\_\_\_\_\_\_\_ К.С. Образцова  «\_\_\_» \_\_\_\_\_\_\_\_\_ 2025 года |

Москва 2025

[ПРАКТИЧЕСКАЯ РАБОТА №1 3](#_Toc208256029)

# ПРАКТИЧЕСКАЯ РАБОТА №1

Цель работы: Выполнить сборки и развертывание контейнеров из примера, и описать своими словами то, что именно вы сделали и как. Описать вкладки в Docker (включая вкладки внутри собранного образа и контейнера) Написать свои Doсkerfile и собрать контейнеры с программами из архива. Залить готовую практическую ( все программы + Dockerfile к каждой из них) на Gitlab или Github и прикрепить к заданию ссылку.

Ход работы:

1. Dart программа

Для дарта используется FROM dart:stable, далее прописать LABEL version = "1.0.1". который является необязательной частью, затем WORKDIR /dartPR1 для создания рабочей директории внутри контейнера, далее COPY . /dartPR1/ для переноса файлов из текущей директории внутрь контейнера, и CMD ["dart","calc.dart"] позволяет запустить файл calc.dart с помощью dart соответственно.

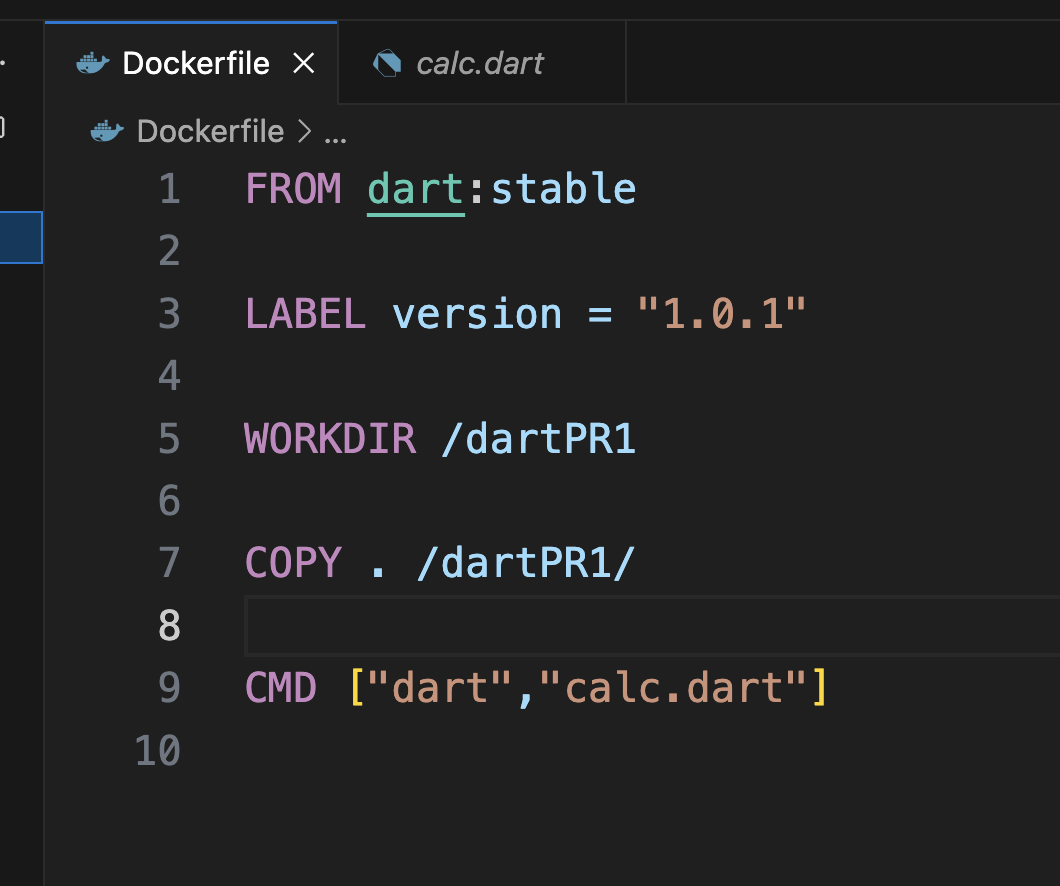


Рисунок 1 – Итоговый Dockerfile к Dart программе

Далее, необходимо собрать образ и запустить контейнер. Сборка образа происходит по команде docker build -t dartpr1-my . флаг -t используется для задания имени образа, это не обязательно, но тогда название будет ужасным. Точка указывает на текущую директорию (там где искать все файлы, которые мы дальше копируем). Затем необходимо запустить контейнер при помощи команды docker run -it --name dokerpr-1-5 dartpri-my. -it создает как бы виртуальный терминал, чтобы дальше с этим можно было работать, как с обычным терминалом в консоли. --name dokerpr-1-5 задаем имя контейнеру и далее имя образа из которого создается контейнер.

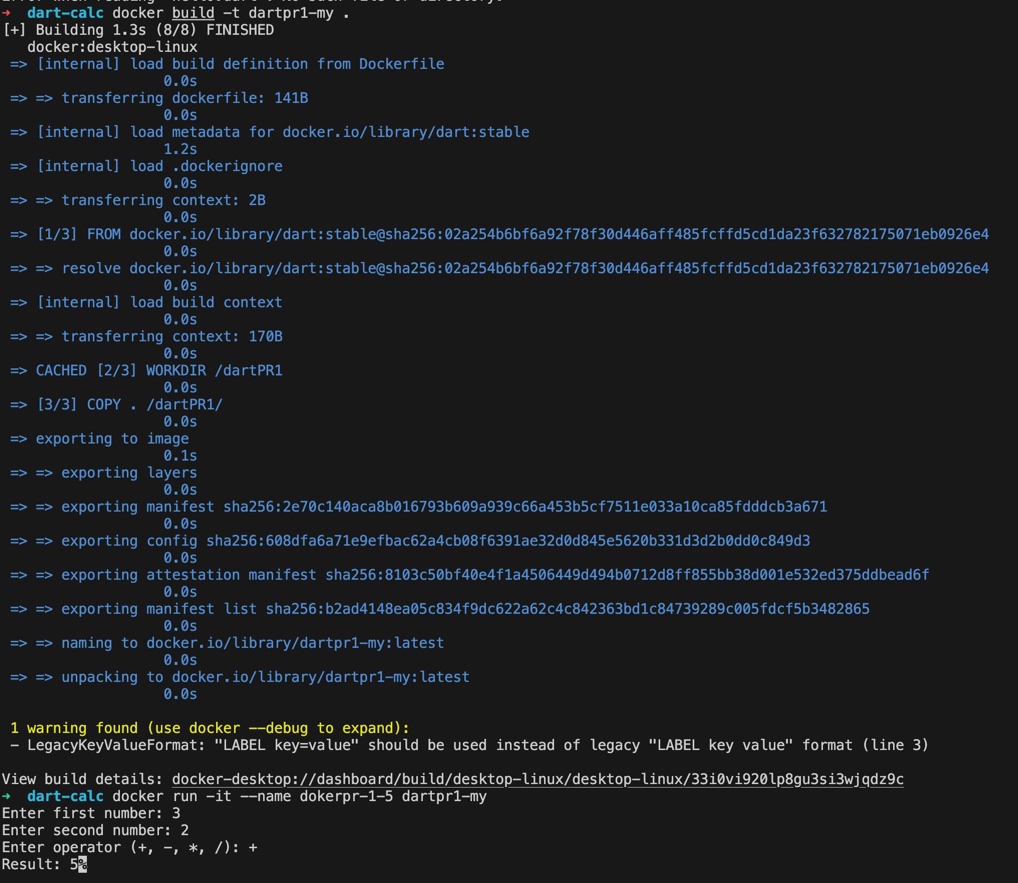


Рисунок 2 – Сборка образа и запуск его контейнера Dart

1. Python калькулятор

Для питона используется FROM python:alpine, далее прописать LABEL version = "1.0.1", затем WORKDIR /pyyPR1 для создания рабочей директории внутри контейнера, далее COPY . /pyyPR1/ для переноса файлов из текущей директории внутрь контейнера, и CMD [ "python" , "calc.py" ] позволяет запустить файл calc.py с помощью python соответственно.

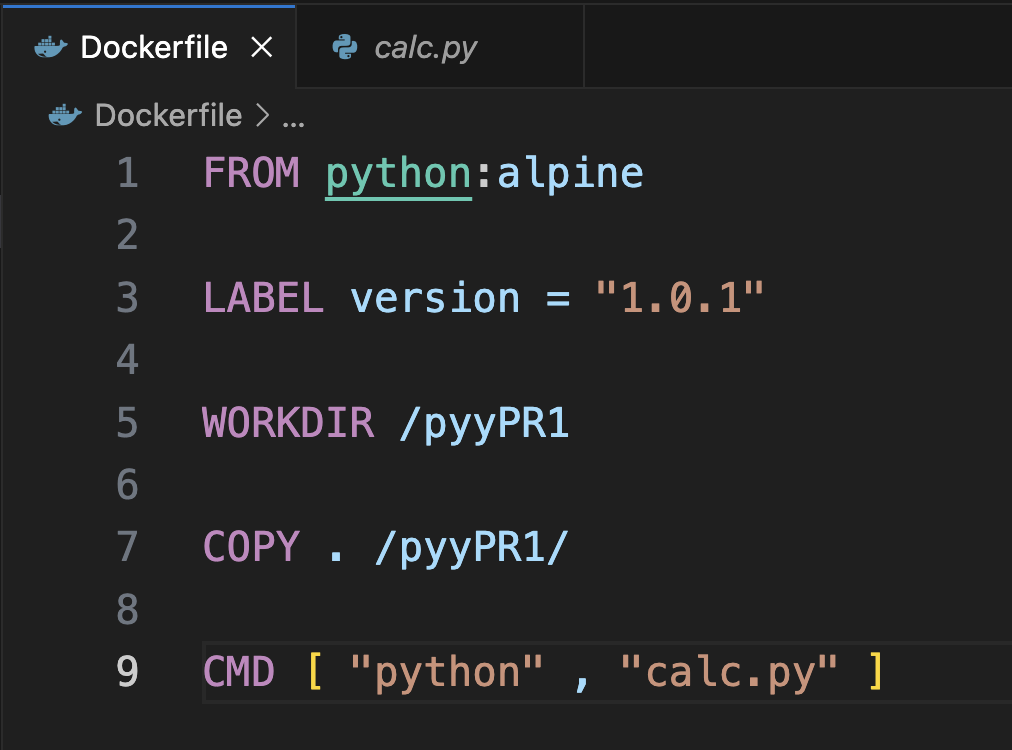


Рисунок 3 - Итоговый Dockerfile к Python-калькулятор

Далее, необходимо собрать образ и запустить контейнер точно так же как и в предыдущих пунктах.

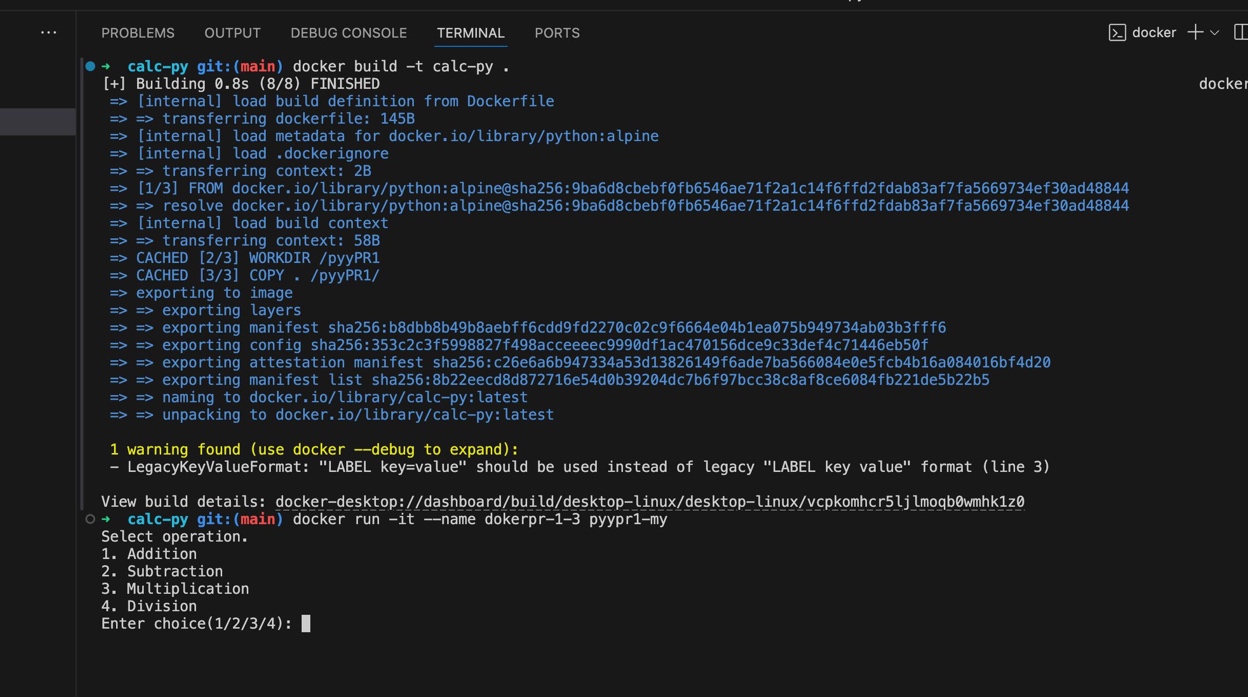


Рисунок 4 - Сборка образа и запуск его контейнера Python-калькулятор

1. Html

Для вебки испоьзуется FROM nginx:alpine. Используем nginx, так как сайты, далее копируем файлы в контейнер COPY . /usr/share/nginx/html и затем указываем порт внутри контейнера EXPOSE 80.

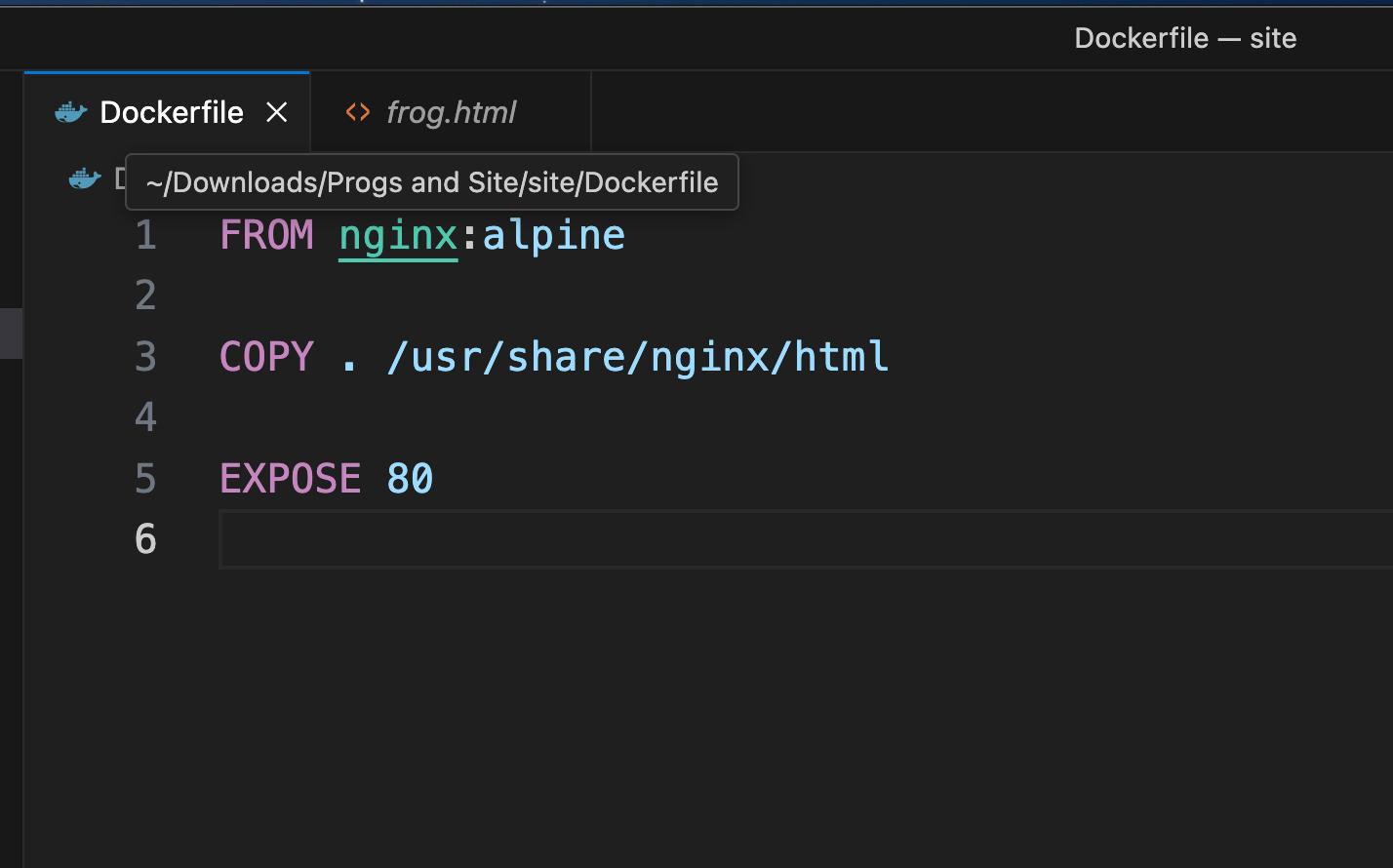


Рисунок 5 - Итоговый Dockerfile к WEB

Далее, необходимо собрать образ и запустить контейнер. Сборка образа происходит по команде docker build -t dartpr1-my . флаг -t используется для задания имени образа, это не обязательно, но тогда название будет ужасным. Точка указывает на текущую директорию (там где искать все файлы, которые мы дальше копируем). Затем необходимо запустить контейнер при помощи команды docker run -d -p 8080:80 --name site-pr1-container site-pr1. С помощью -d запускаем контейнер в фоне, -p 8080:80 подбрасываем порты (порт 80 мы указали в EXPOSE). О остальном все то же самое.

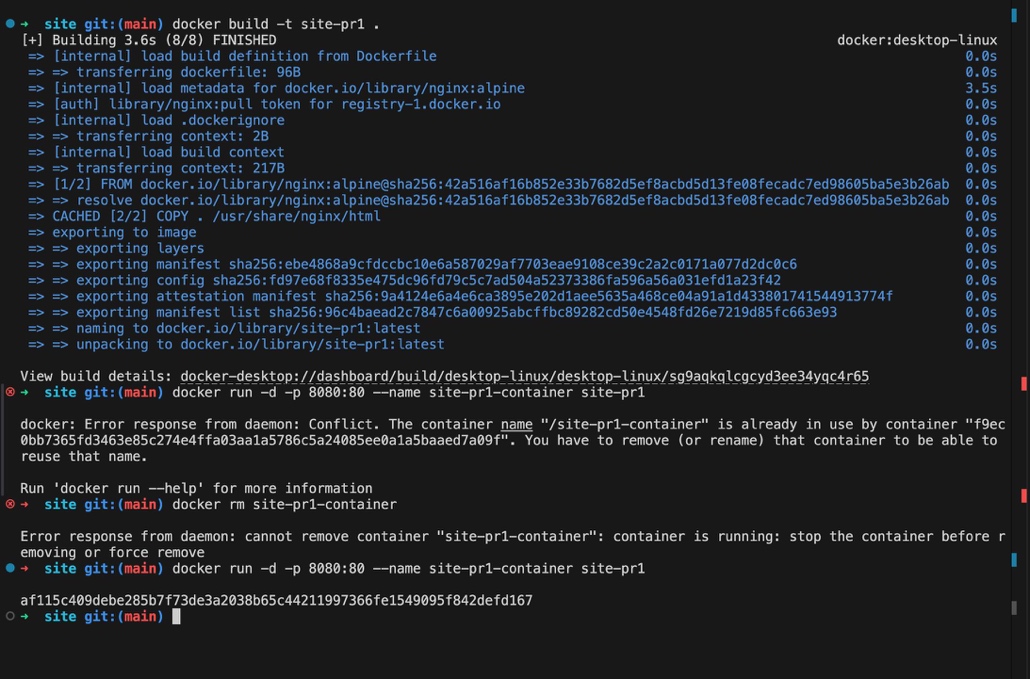


Рисунок 6 - Сборка образа и запуск его контейнера WEB

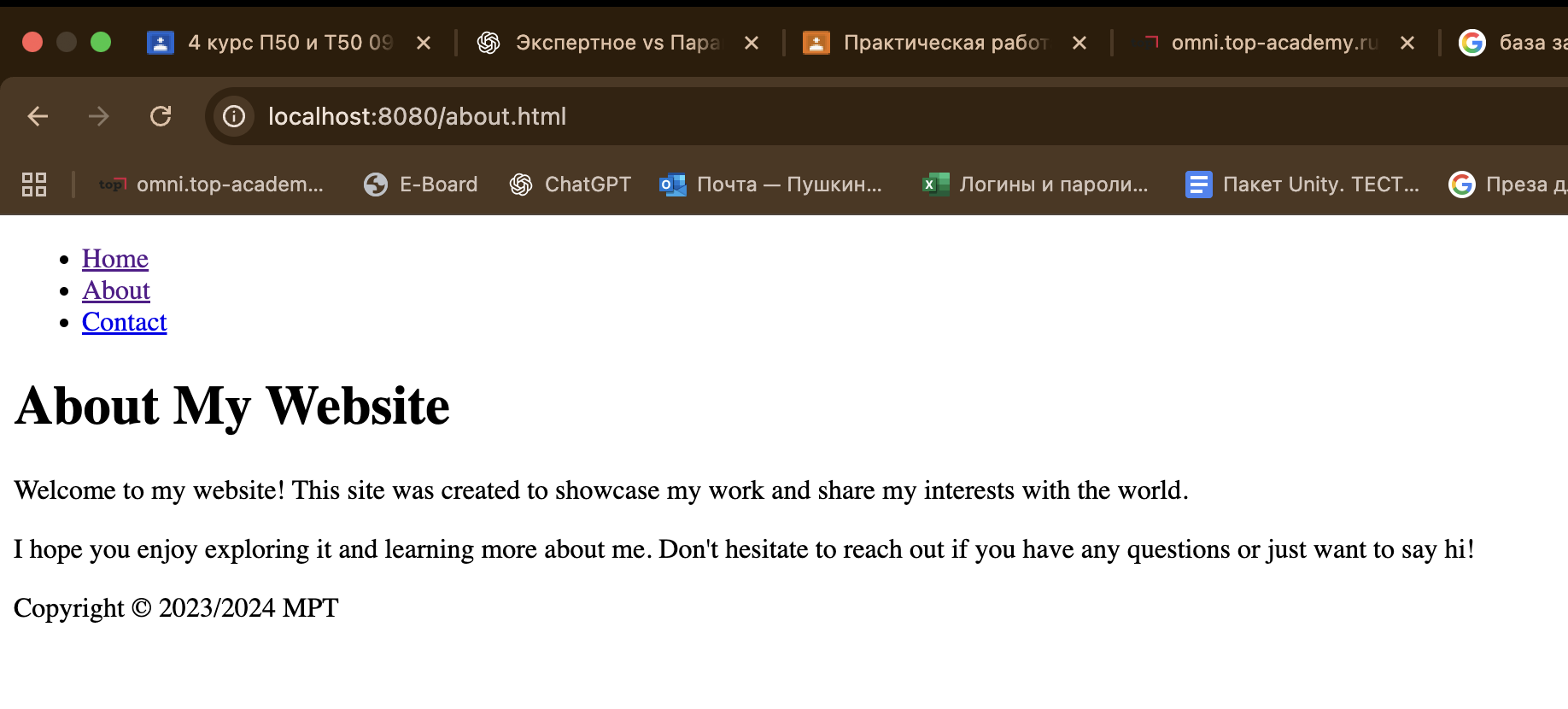


Рисунок 7 – Результат запуска WEB контейнера.

1. Java калькулятор

Для джавы используется FROM openjdk:alpine, далее прописываем WORKDIR /javaaPR1 и COPY . /javaaPR1/ для создания директории и для переноса в нее файлов проекта. В джаве, необходимо прописать RUN javac calc.java так как в файле с калькулятором есть класс, докеру надо указать, что у нас есть класс, иначе, он его не найдет. И теперь прописываем CMD ["java", "calc"] для того, чтобы запустить класс calc в файле с джавой, который мы указали выше в строчке run, с помощью java.

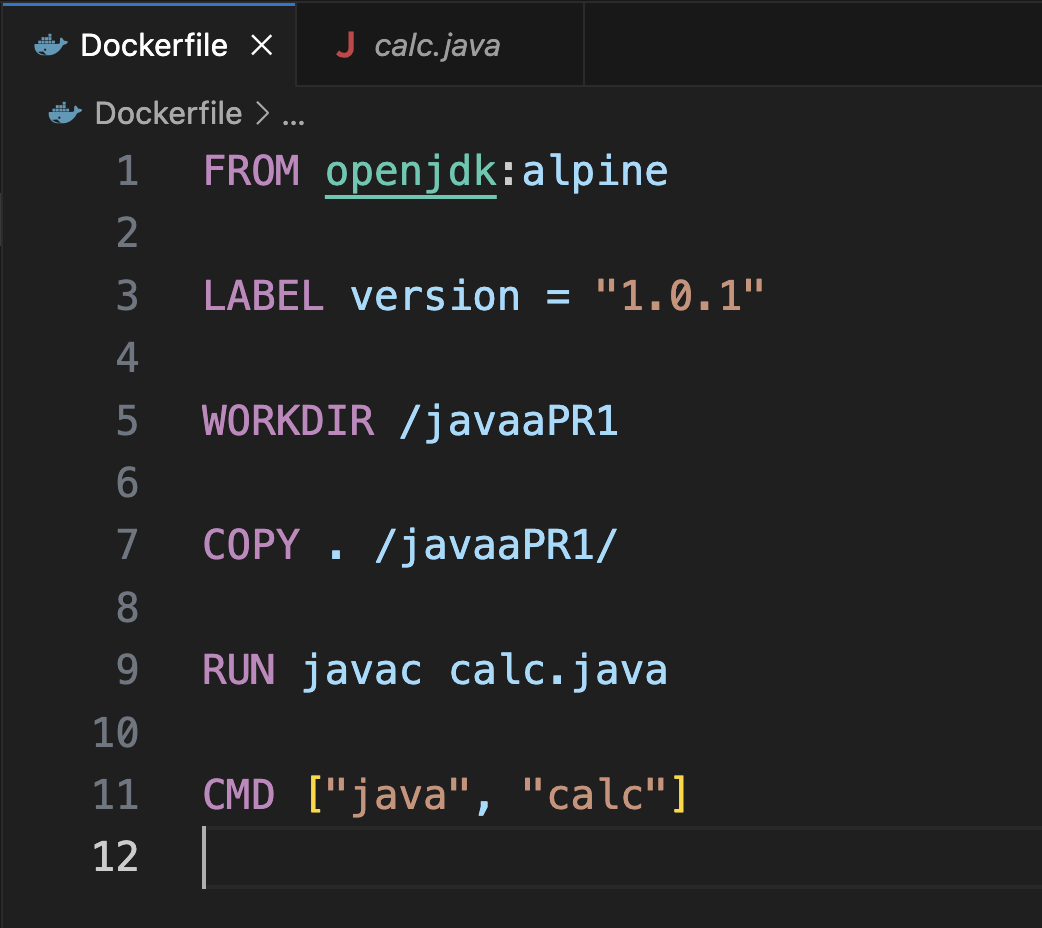


Рисунок 8 - Итоговый Dockerfile к Java-калькулятору

Далее, необходимо собрать образ и запустить контейнер точно так же, как и в предыдущих пунктах.

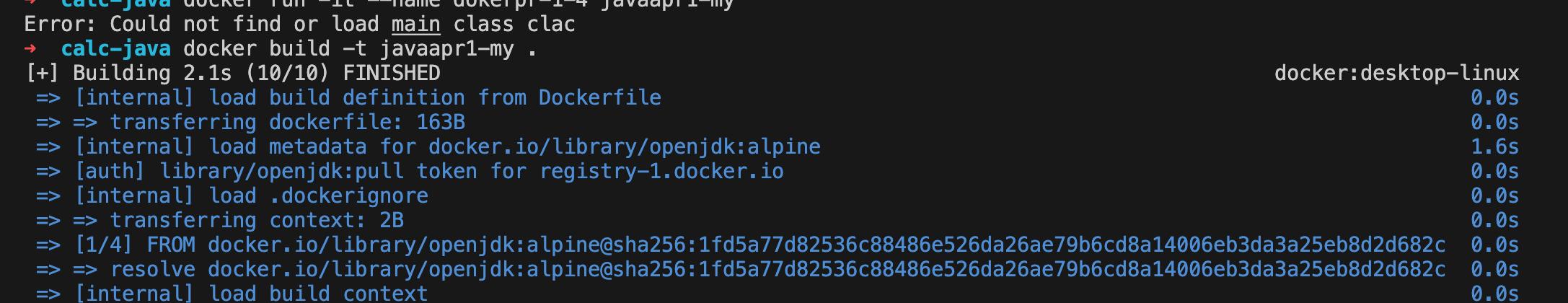


Рисунок 9 - Сборка образа Java-калькулятора

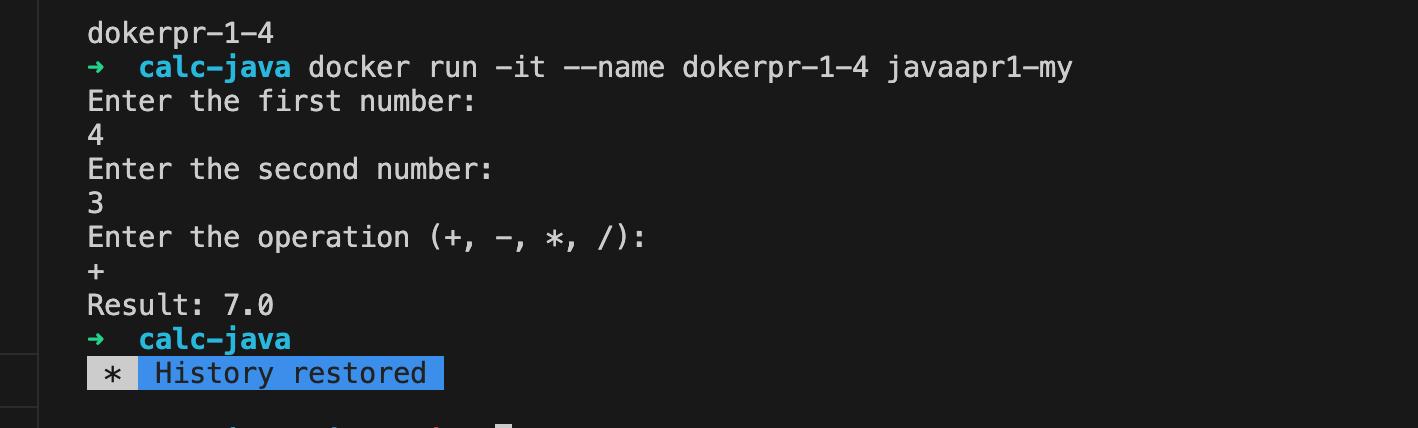


Рисунок 10 - Запуск контейнера Java-калькулятора

1. Java игра

Здесь все тоже самое, что и в калькуляторе, различие в создаваемой директории, так как не могут быть одинаковые WORKDIR /javaPR1 и COPY . /javaPR1/. И запуск игры по тому же сценарию RUN javac game.java и CMD [ "java" , "game" ].

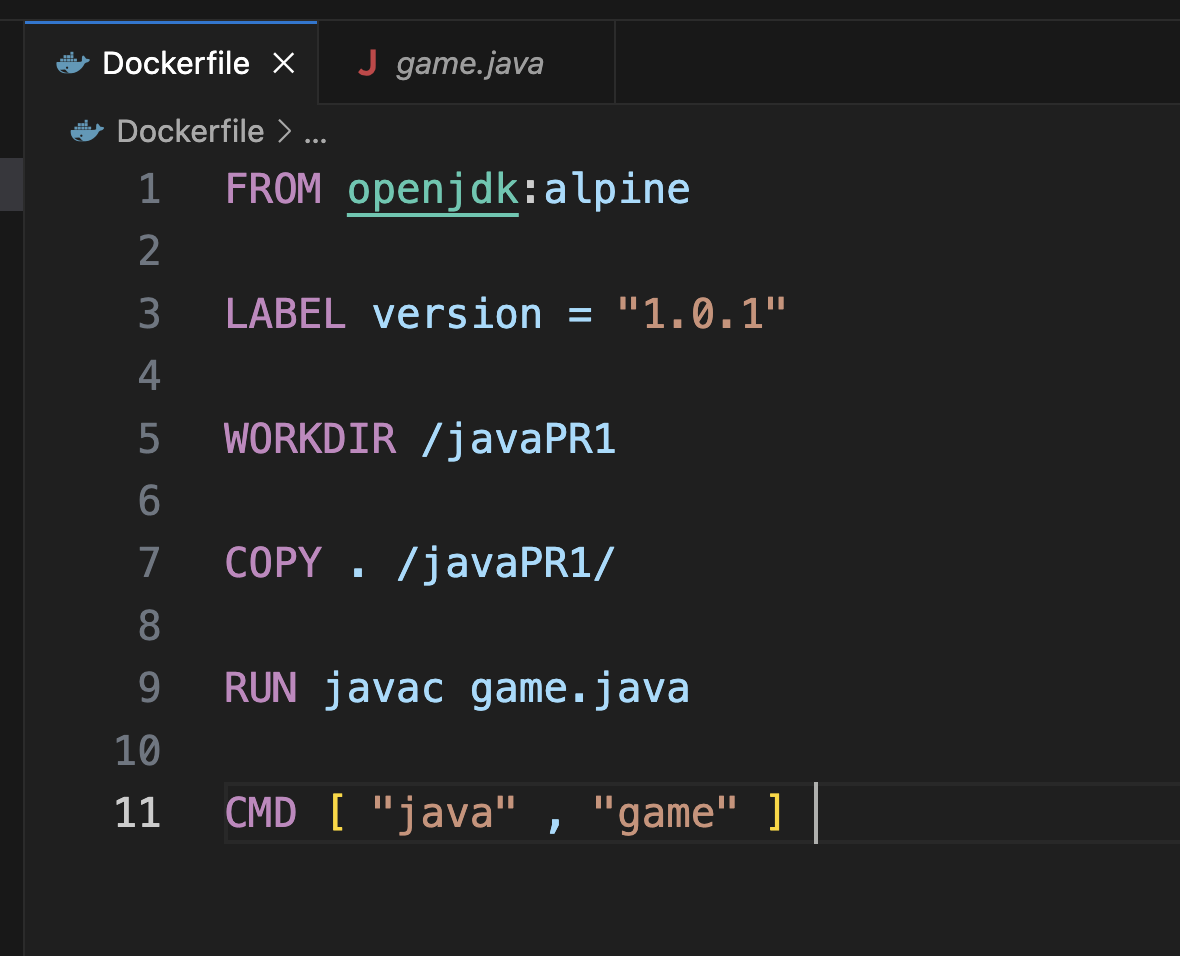


Рисунок 11 - Итоговый Dockerfile к Java-игре

Далее, необходимо собрать образ и запустить контейнер точно так же, как и в предыдущих пунктах.

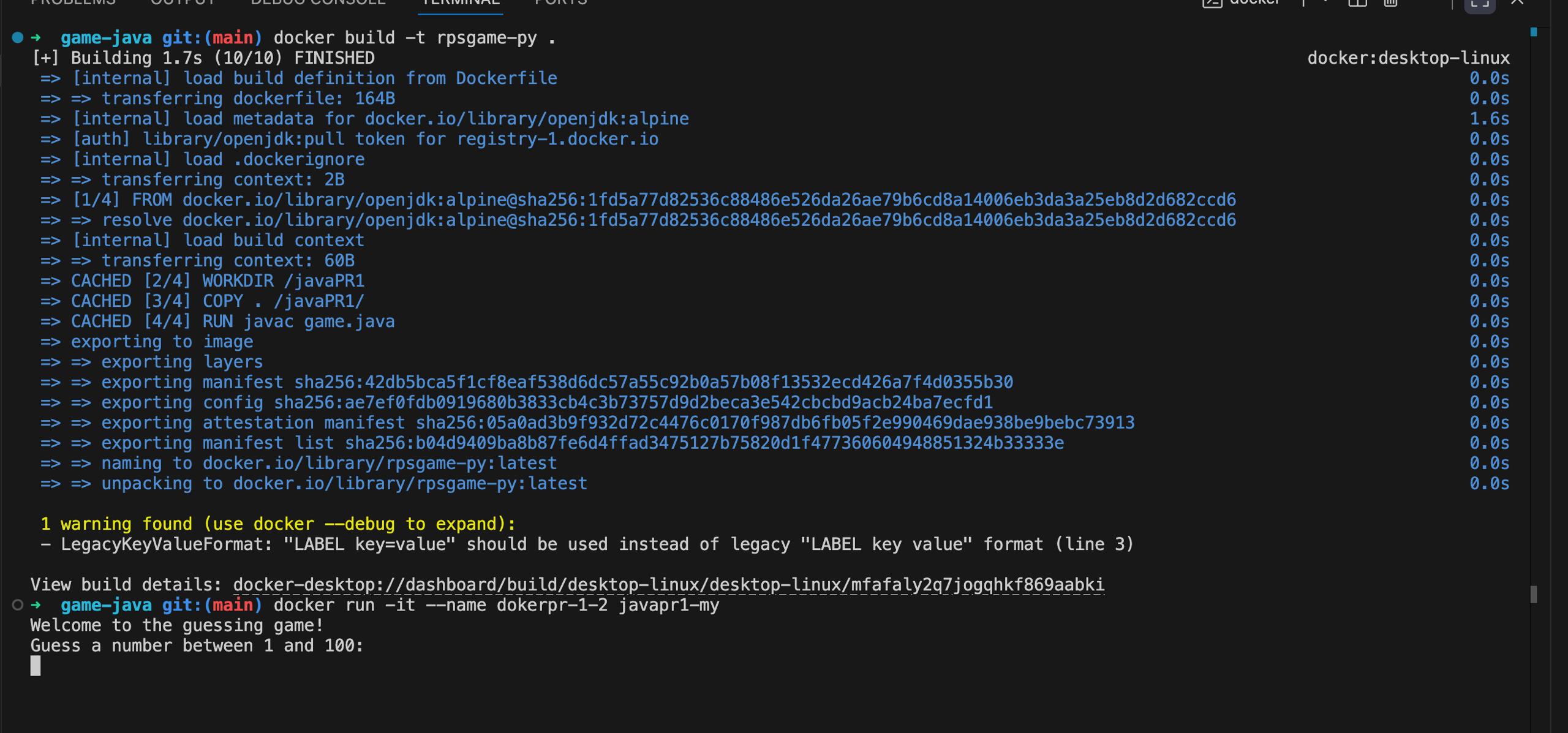


Рисунок 12 - Сборка образа и запуск его контейнера Java-игры

1. Python-игра

Здесь все то же самое, что и в Python-калькуляторе. Отличаются только директории и файлы которые мы запускаем.

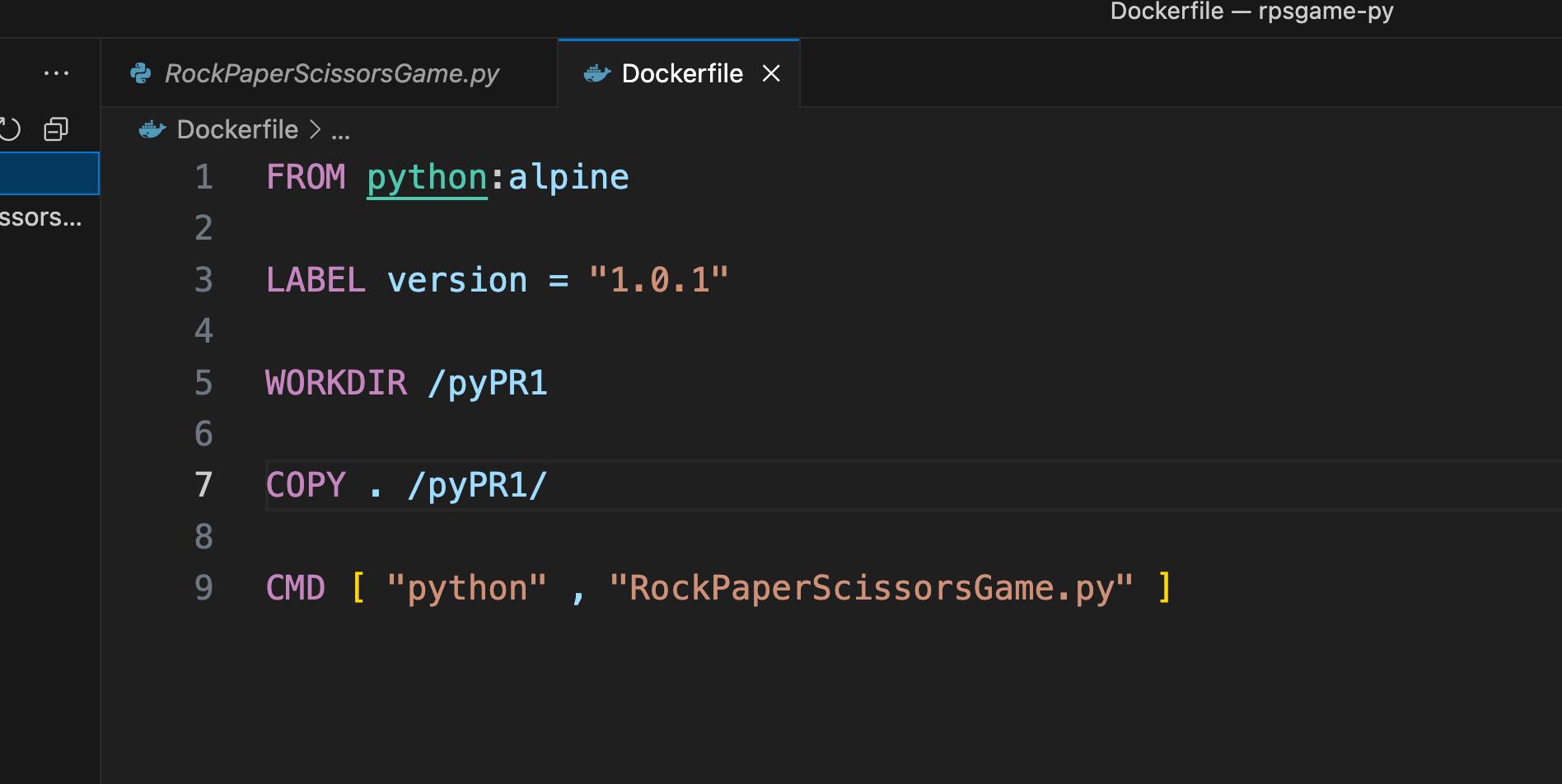


Рисунок 13 - Итоговый Dockerfile к Python-игре

Далее, необходимо собрать образ и запустить контейнер точно так же, как и в предыдущих пунктах.

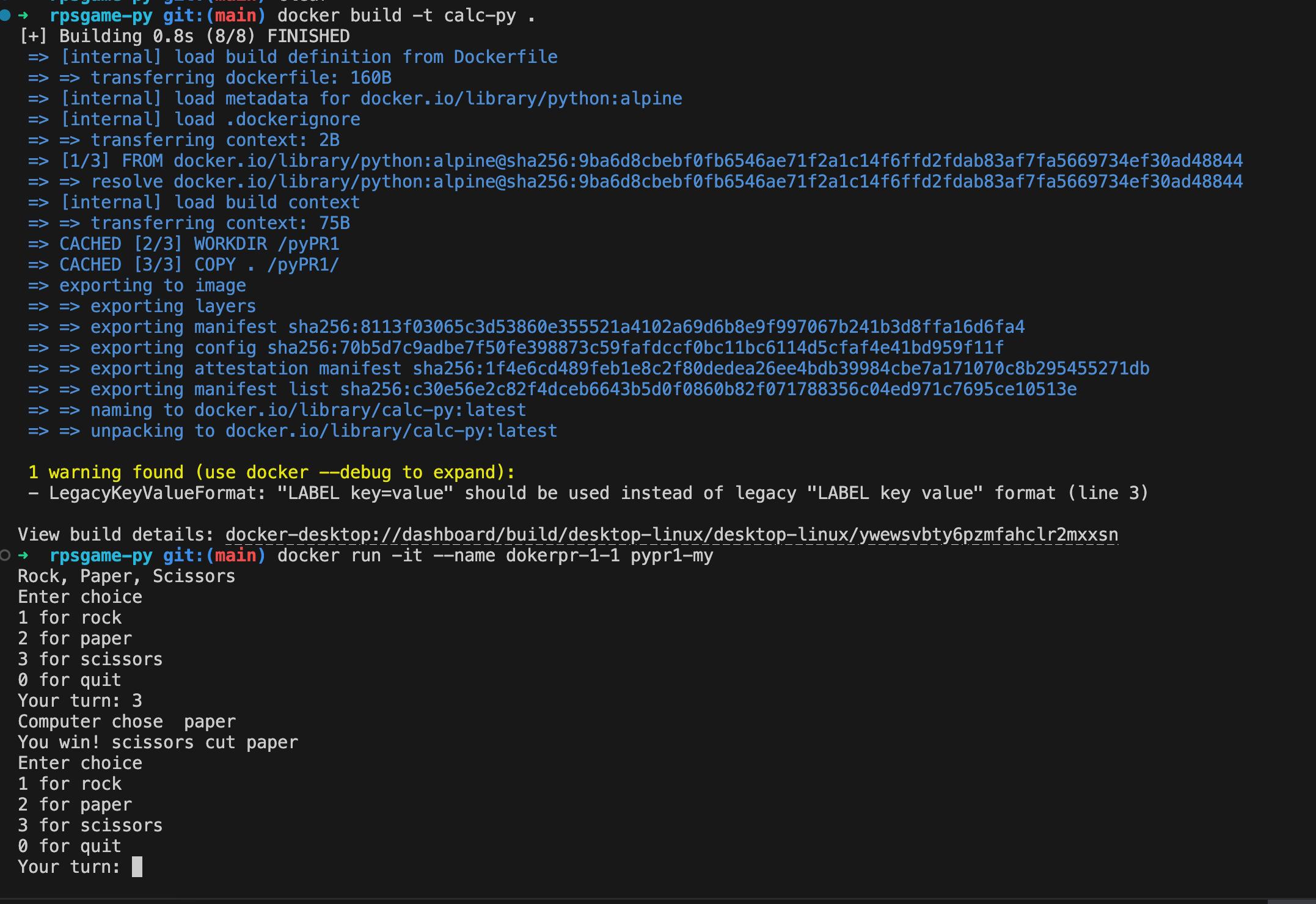


Рисунок 14 - Сборка образа и запуск его контейнера Python-игры

1. Описание вкладок Docker Desktop.

Containers - Здесь список **контейнеров** — запущенных и остановленных.

Images - Это список всех образов.

Volumes - Здесь отображаются тома

Builds - Это раздел про сборку образов.

Logs – Здесь показываются логи работы контейнера (что он выводит в консоль).

Inspect – Здесь можно посмотреть подробную «техническую информацию» о контейнере (IP, порты, настройки и т. д.).

Bind mounts – Здесь отображаются папки на компьютере, которые подключены внутрь контейнера.

Exec – Здесь можно запустить команду внутри контейнера (как будто ты зашёл в него через терминал).

Files – Здесь видна файловая система контейнера (все папки и файлы внутри него).

Stats – Здесь показываются ресурсы, которые потребляет контейнер (CPU, память, сеть, диск).

Info – Здесь показывается общая информация о сборке образа (название, идентификатор, ошибки).

Source – Здесь можно посмотреть, из каких файлов/директорий собирался образ.

Logs – Здесь отображаются логи процесса сборки (что делал Docker при docker build).

History – Здесь видна история слоёв образа (какие команды из Dockerfile создавали текущий результат).

Вывод: В ходе практической работы было выполнено развертывание контейнеров по примерам, изучены и описаны вкладки Docker Desktop и их назначение. Были написаны собственные Dockerfile, собраны контейнеры с программами из архива и загружены в репозиторий GitLab/GitHub. Работа позволила закрепить навыки по сборке и запуску контейнеров, а также работе с Dockerfile.