

Департамент образования и науки города Москвы
Государственное автономное образовательное учреждение
высшего образования города Москвы
«Московский городской педагогический университет»
Институт цифрового образования
Департамент информатики управления и технологий

Агафонов Антон Александрович БД-241м

Практическая работа 3-1. Dockerfile

Направление подготовки/специальность

38.04.05 - Бизнес-информатика

Бизнес-аналитика и большие данные

(очная форма обучения)

Вариант 1

Москва

2024

| | |
|---------------------|----|
| Введение..... | 3 |
| Основная часть..... | 3 |
| Заключение..... | 15 |

Введение

Практическая работа нацелена на знакомство студентов с Docker, создание хорошего и плохого dockerfile, использование Docker для развертывания приложений.

Цель

Получить практические навыки написания Dockerfile и научиться различать и применять хорошие и плохие практики при создании Docker-образов

Основная часть

1. Написание хорошего и плохого Dockerfile.

Для начала, необходимо на виртуальной машине создать 2 директории “bad_docker_file” и “good_docker_file”.

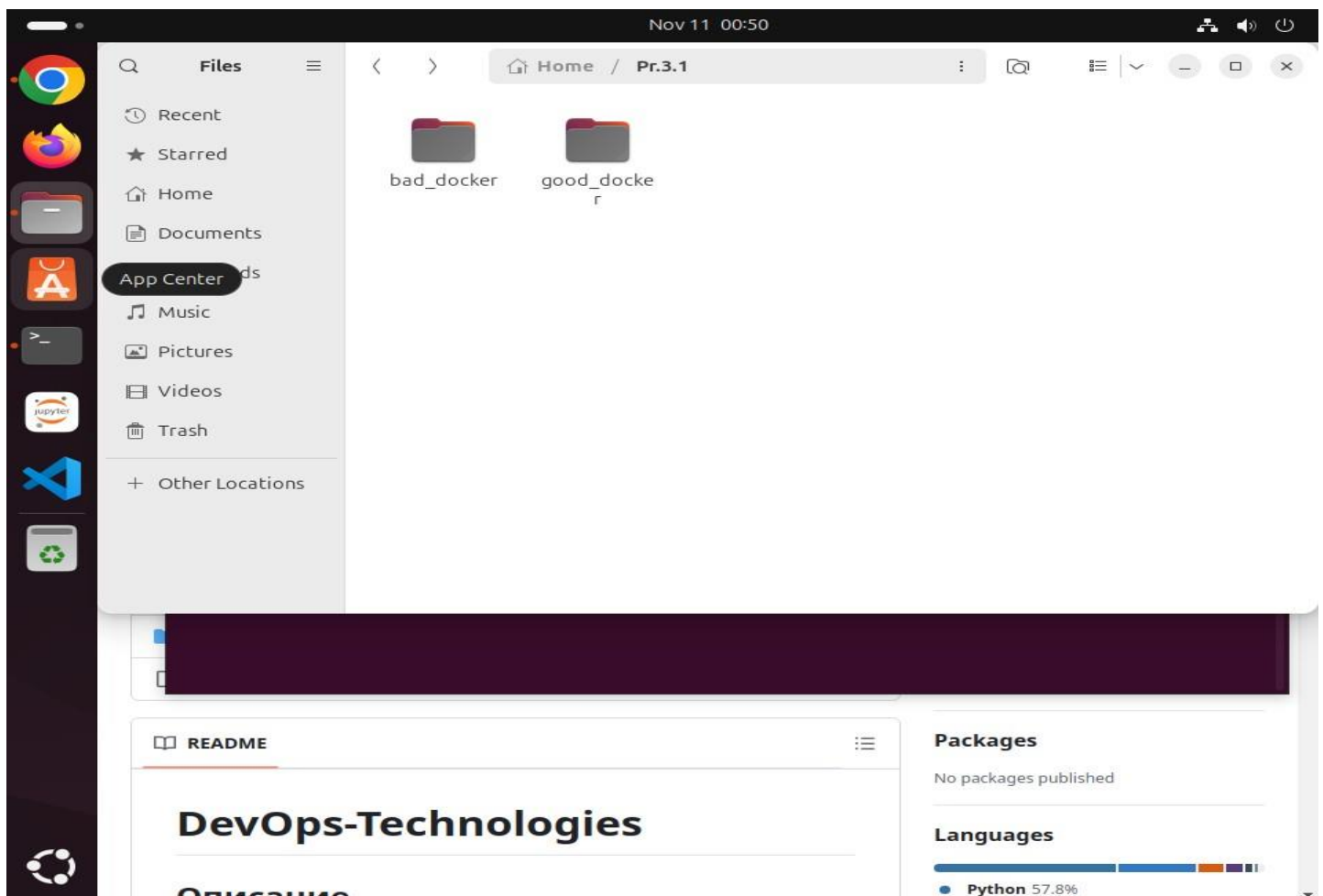


Рис.1

Внутри каждой из этих директорий создадим dockerfile.

Содержание плохого dockerfile from python:latest workdir

/app

copy ./requirements.txt /app/server/requirements.txt copy ./alembic.ini

/app/alembic.ini copy ./src /app/server/src run pip install --no-cache-dir -upgrade

-r /app/server/requirements.txt cmd ["uvicorn",

"server.src.main:app", "--host", "0.0.0.0", "--port", "8000"] volume /app/data

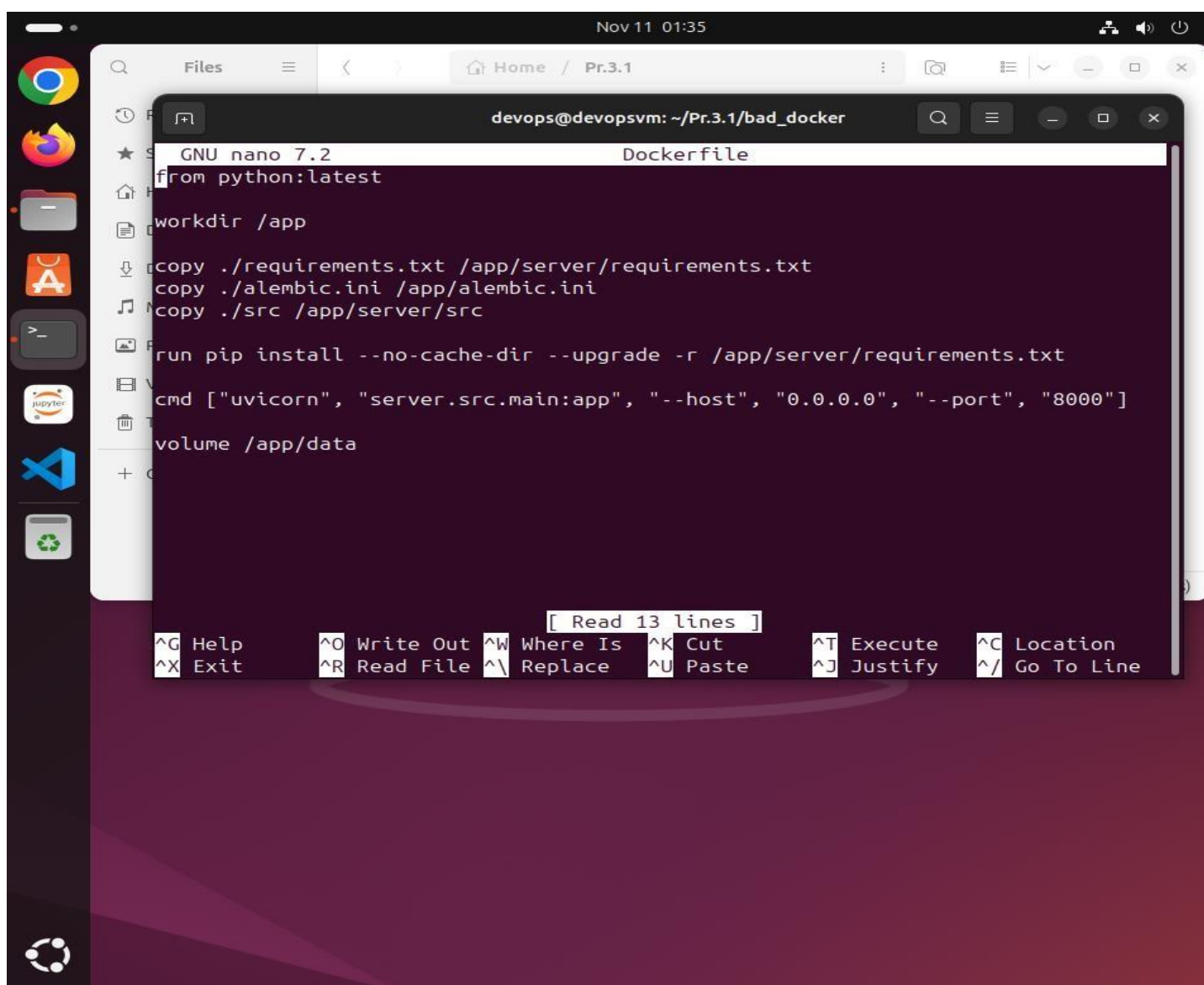


Рис.2 Содержимое плохого dockerfile

Содержание хорошего dockerfile

FROM python:3.10

WORKDIR /app

COPY ./requirements.txt /app/server/requirements.txt

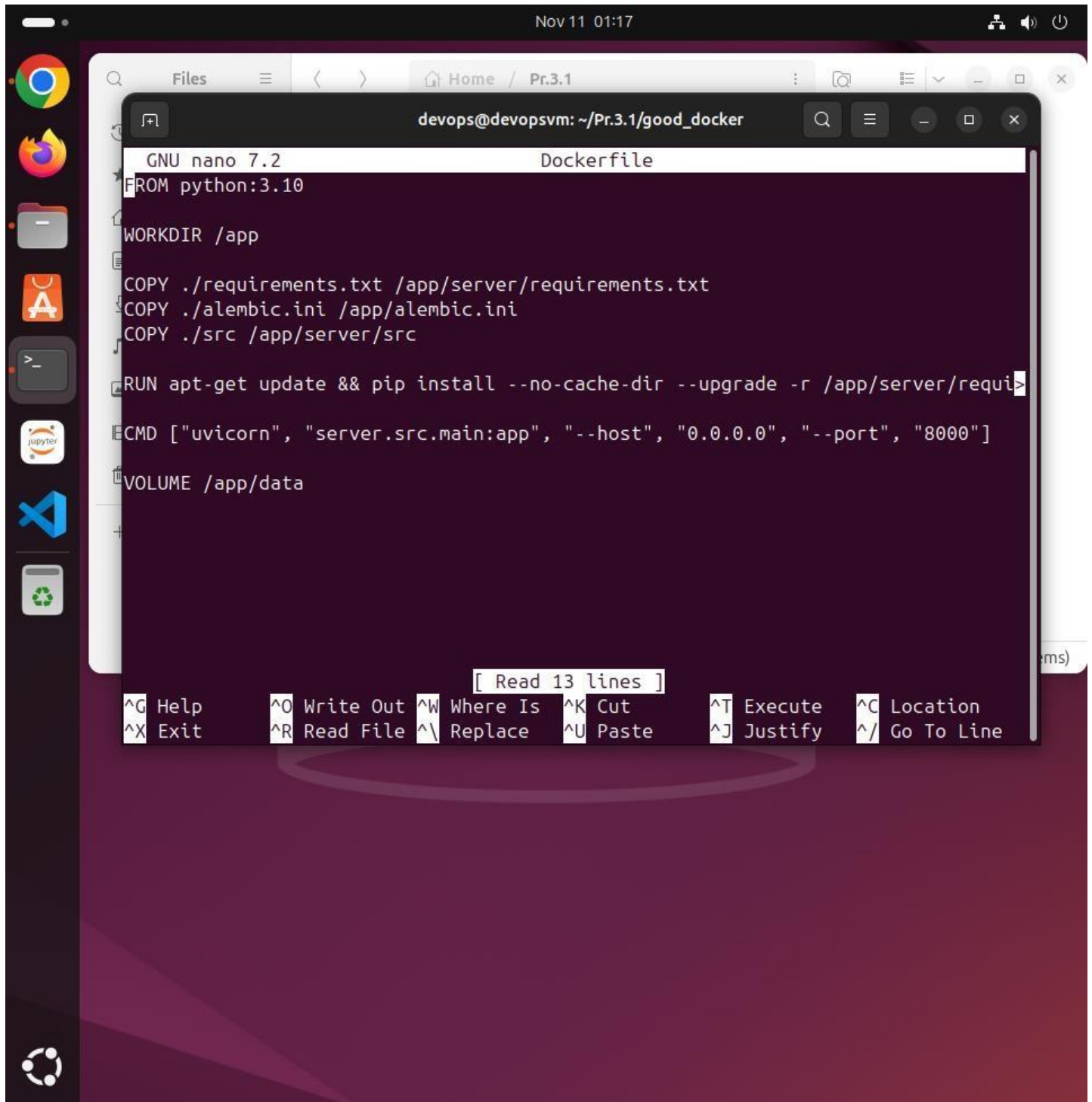
COPY ./alembic.ini /app/alembic.ini

COPY ./src /app/server/src

RUN apt-get update && pip install --no-cache-dir --upgrade -r /app/server/requirements.txt

CMD ["uvicorn", "server.src.main:app", "--host", "0.0.0.0", "--port", "8000"]

VOLUME /app/data



```
GNU nano 7.2 Dockerfile
FROM python:3.10

WORKDIR /app

COPY ./requirements.txt /app/server/requirements.txt
COPY ./alembic.ini /app/alembic.ini
COPY ./src /app/server/src

RUN apt-get update && pip install --no-cache-dir --upgrade -r /app/server/requirements.txt

CMD ["uvicorn", "server.src.main:app", "--host", "0.0.0.0", "--port", "8000"]

VOLUME /app/data
```

[Read 13 lines]

| | | | | | |
|----------------|---------------------|--------------------|-----------------|-------------------|----------------------|
| ^G Help | ^O Write Out | ^W Where Is | ^K Cut | ^T Execute | ^C Location |
| ^X Exit | ^R Read File | ^_ Replace | ^U Paste | ^J Justify | ^/ Go To Line |

Рис.3 Содержимое хорошего dockerfile

После чего, перенесем содержимое репозитория

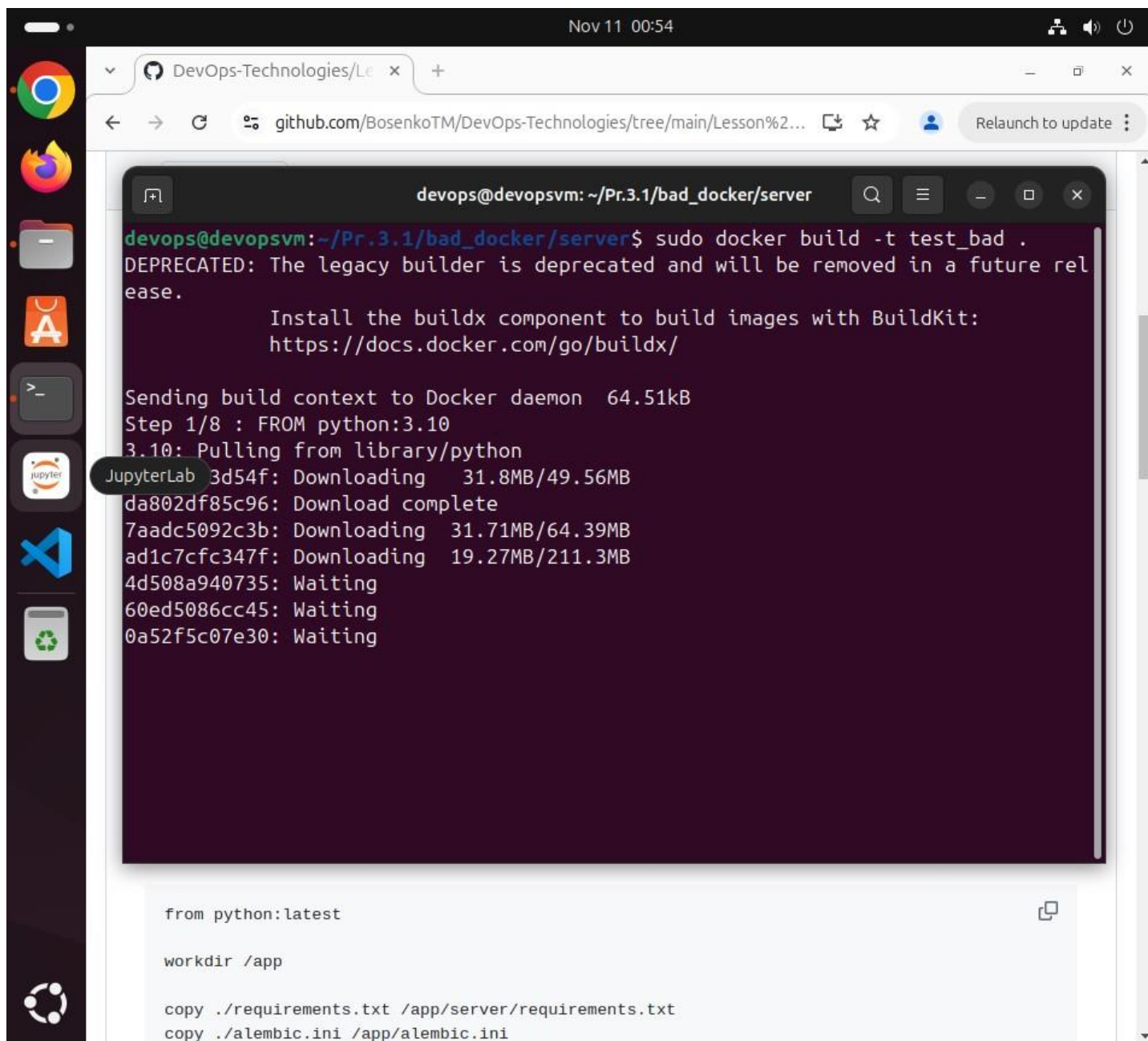
“https://github.com/BosenkoTM/DevOps-Technologies/tree/main/Lesson%203%20Containerizing%20Applications/lab3_1#%D0%B7%D0%B0%D0%BF%D1%83%D1%81%D0%BA”

в обе директории и внутрь их перенесем и заменим Dockerfile.

Далее, начнем построение Docker-образа.

Сначала, перейдем под администратором.

Sudo su docker build -t test_bad .



```
devops@devopsvm: ~/Pr.3.1/bad_docker/server
devops@devopsvm:~/Pr.3.1/bad_docker/server$ sudo docker build -t test_bad .
DEPRECATED: The legacy builder is deprecated and will be removed in a future release.

                Install the buildx component to build images with BuildKit:
                https://docs.docker.com/go/buildx/

Sending build context to Docker daemon 64.51kB
Step 1/8 : FROM python:3.10
3.10: Pulling from library/python
JupyterLab 3d54f: Downloading 31.8MB/49.56MB
da802df85c96: Download complete
7aad5092c3b: Downloading 31.71MB/64.39MB
ad1c7cfc347f: Downloading 19.27MB/211.3MB
4d508a940735: Waiting
60ed5086cc45: Waiting
0a52f5c07e30: Waiting
```

```
from python:latest

workdir /app

copy ./requirements.txt /app/server/requirements.txt
copy ./alembic.ini /app/alembic.ini
```

Рис.4 Запуск плохого файла

После запуска плохого файла получим ошибки, связанные с недостаточными зависимостями.

Запустим хороший dockerfile. **docker build -t test_good**

. docker run -d --name test_ good -p 8000:8000 test_
good

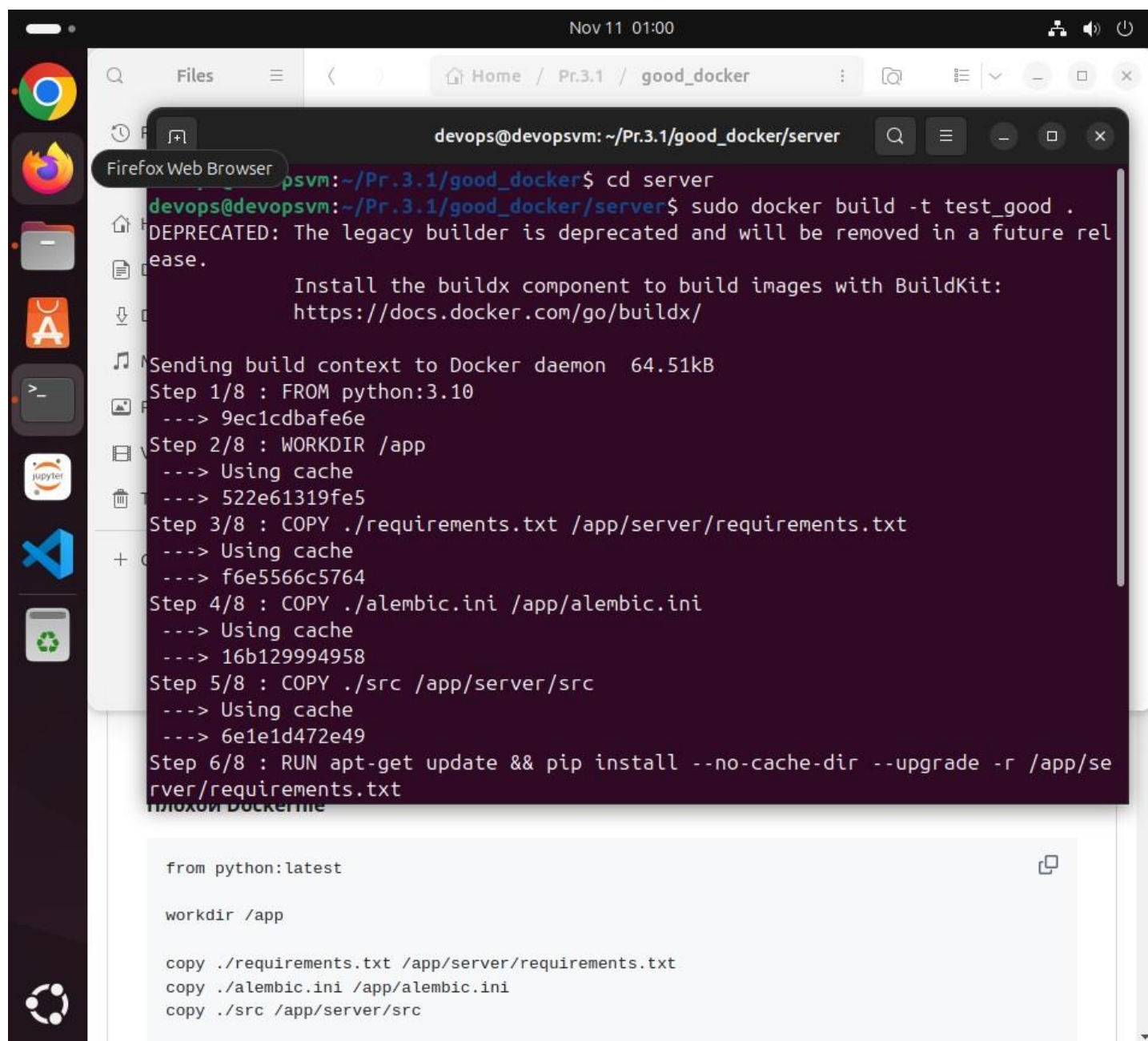


Рис.5 Запуск хорошего файла

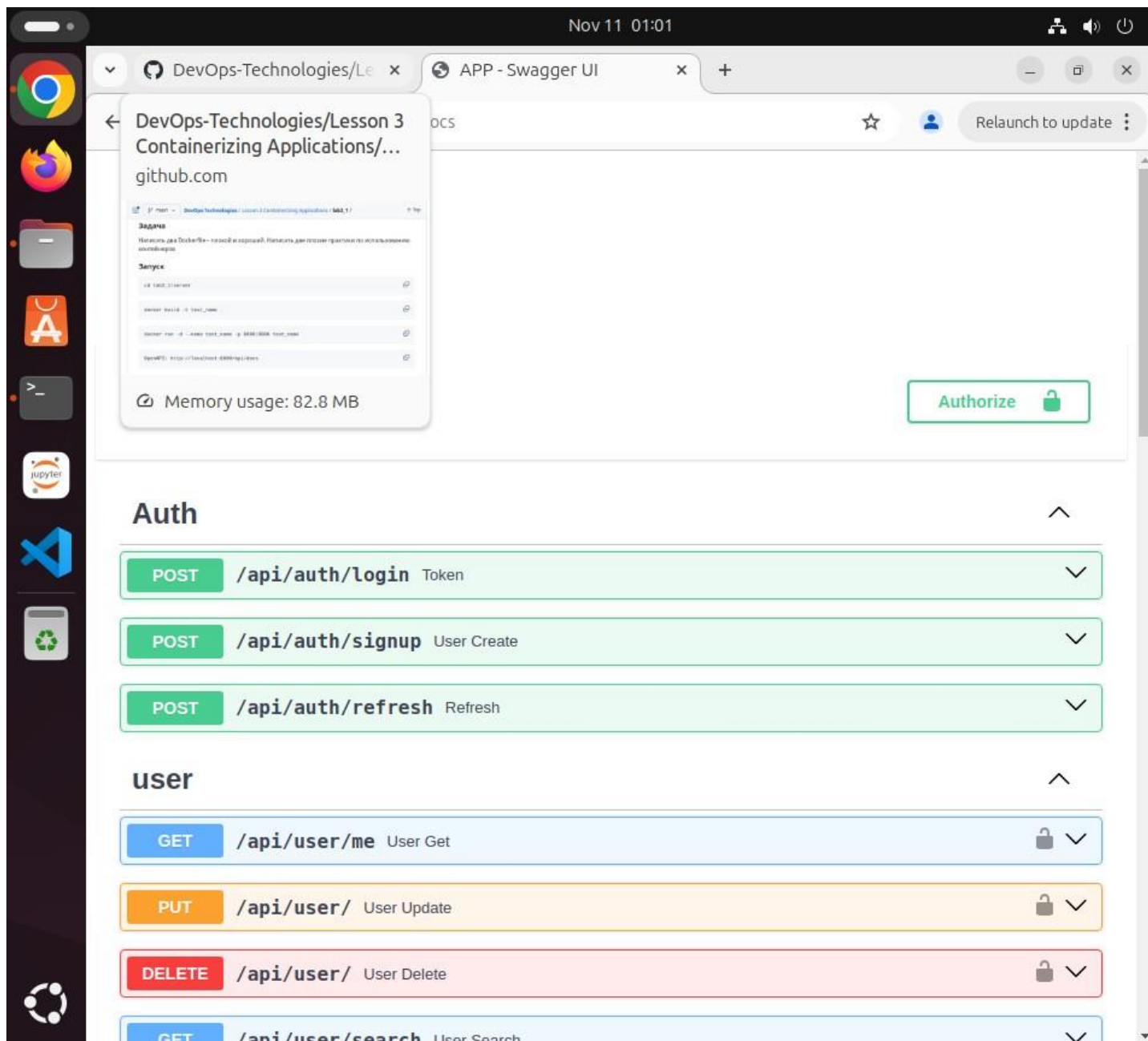


Рис.6 Результат выполнения хорошего dockerfile

Для просмотра размера хорошего образа пропишем **sudo docker images**

```
test_good    latest    8049de4b3236    10 minutes ago    1.1GB
python       3.10     9ec1cdbafe6e    3 weeks ago      1GB
devops@devopsvm:~/Pr.3.1/good_docker/server$
```

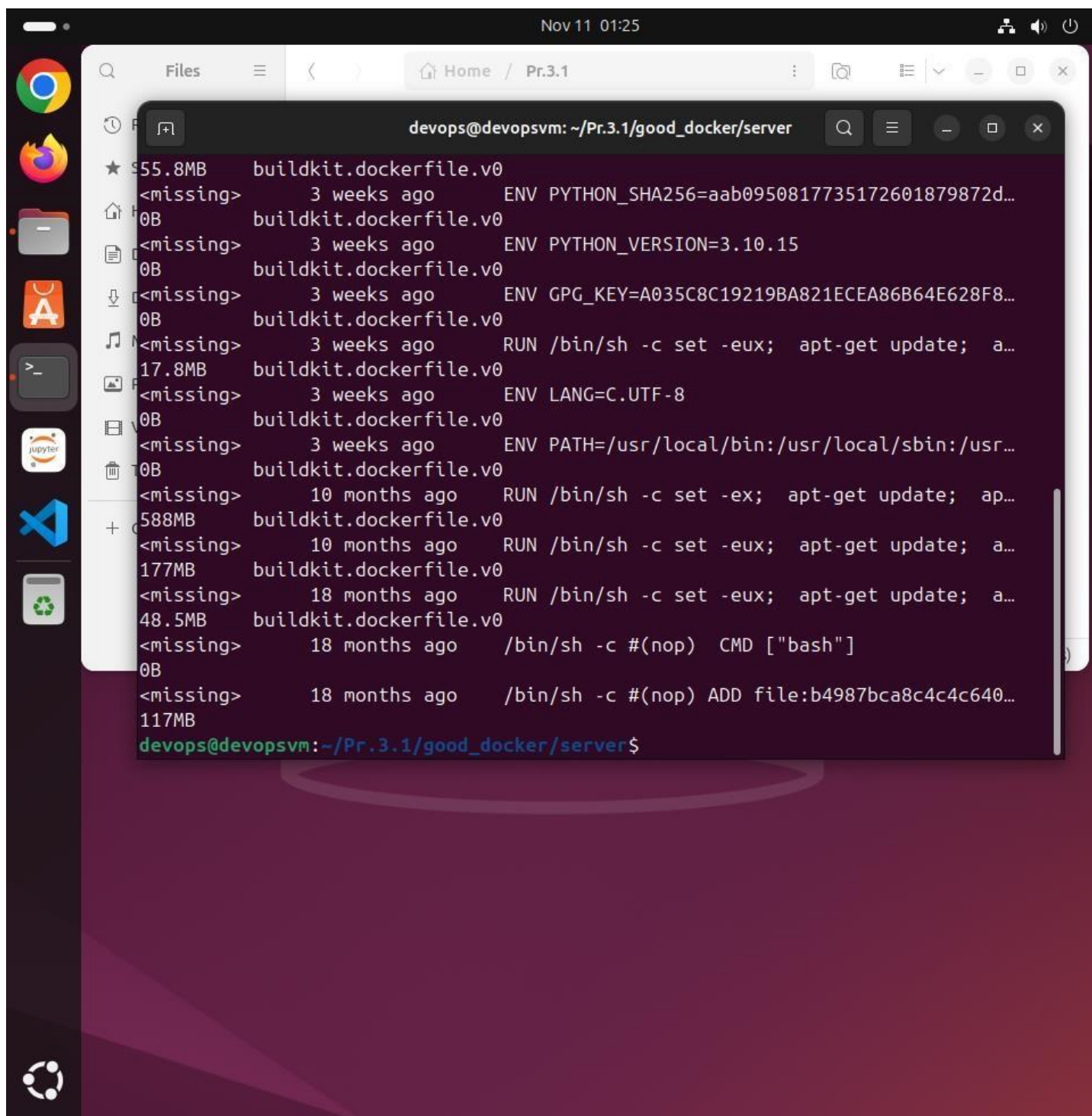
Рис.7

Как видно хороший образ (test_good) весит 1.1GB

Для просмотра кол-ва слоев хорошего образа пропишем

Docker history <image_id>

В нашем случае **Docker history 8049de4b3236**



```
Nov 11 01:25
Files
Home / Pr.3.1
devops@devopsvm: ~/Pr.3.1/good_docker/server
$ docker history 8049de4b3236
55.8MB buildkit.dockerfile.v0
<missing> 3 weeks ago ENV PYTHON_SHA256=aab0950817735172601879872d...
0B buildkit.dockerfile.v0
<missing> 3 weeks ago ENV PYTHON_VERSION=3.10.15
0B buildkit.dockerfile.v0
<missing> 3 weeks ago ENV GPG_KEY=A035C8C19219BA821ECEA86B64E628F8...
0B buildkit.dockerfile.v0
<missing> 3 weeks ago RUN /bin/sh -c set -eux; apt-get update; a...
17.8MB buildkit.dockerfile.v0
<missing> 3 weeks ago ENV LANG=C.UTF-8
0B buildkit.dockerfile.v0
<missing> 3 weeks ago ENV PATH=/usr/local/bin:/usr/local/sbin:/usr...
0B buildkit.dockerfile.v0
<missing> 10 months ago RUN /bin/sh -c set -ex; apt-get update; ap...
588MB buildkit.dockerfile.v0
<missing> 10 months ago RUN /bin/sh -c set -eux; apt-get update; a...
177MB buildkit.dockerfile.v0
<missing> 18 months ago RUN /bin/sh -c set -eux; apt-get update; a...
48.5MB buildkit.dockerfile.v0
<missing> 18 months ago /bin/sh -c #(nop) CMD ["bash"]
0B
<missing> 18 months ago /bin/sh -c #(nop) ADD file:b4987bca8c4c4c640...
117MB
devops@devopsvm:~/Pr.3.1/good_docker/server$
```

Рис.8

При создании Docker-образа он использовал кэшированные слои из базового образа, которые были созданы ранее

Время сборки

Для просмотра времени сборки используем команду

time docker build -t test_good . – для просмотра сборки при использовании кэшированных слоев

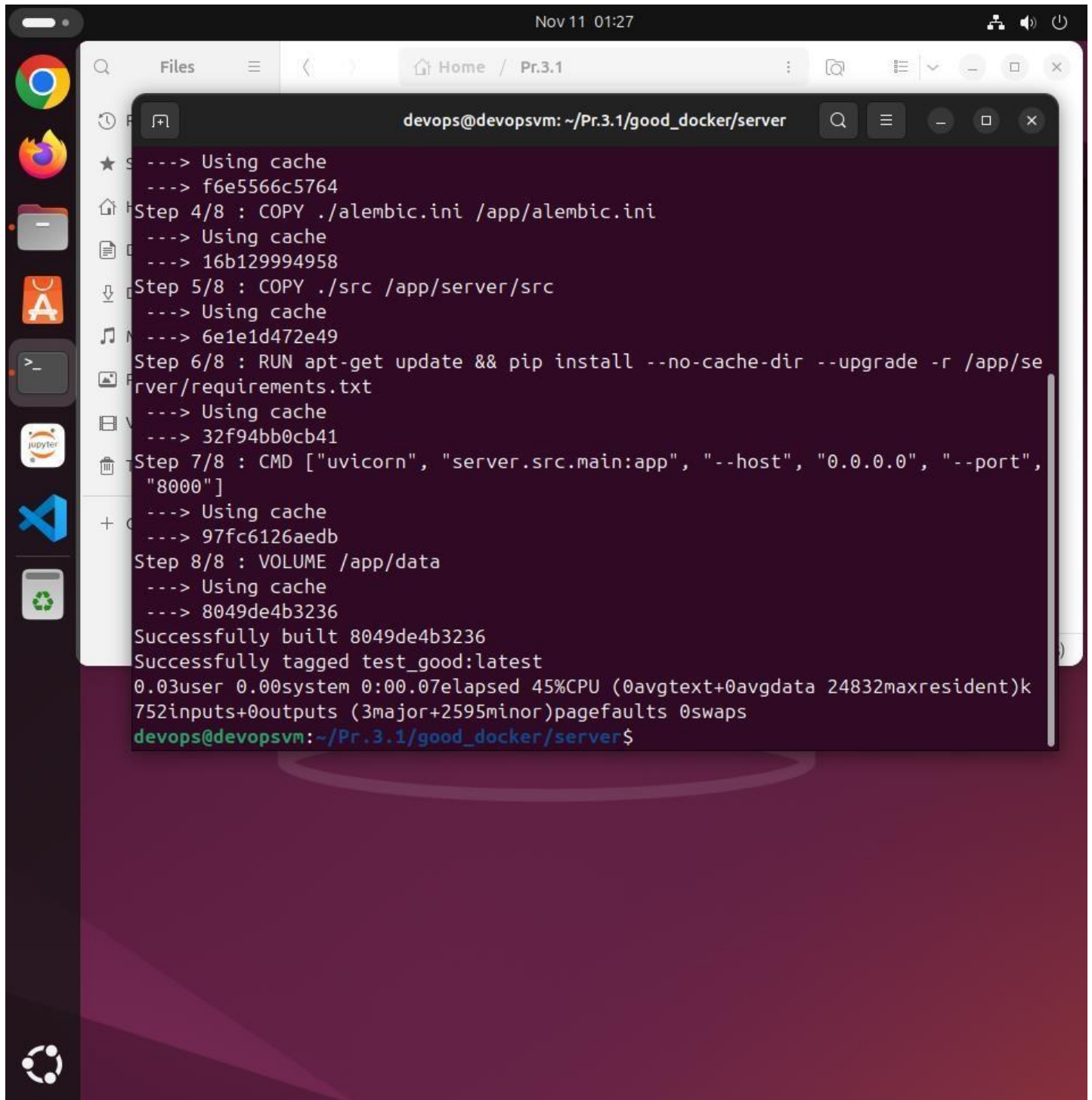


Рис.9

Время сборки без кэшированных слоев

Docker build --no-cache -t test_good .

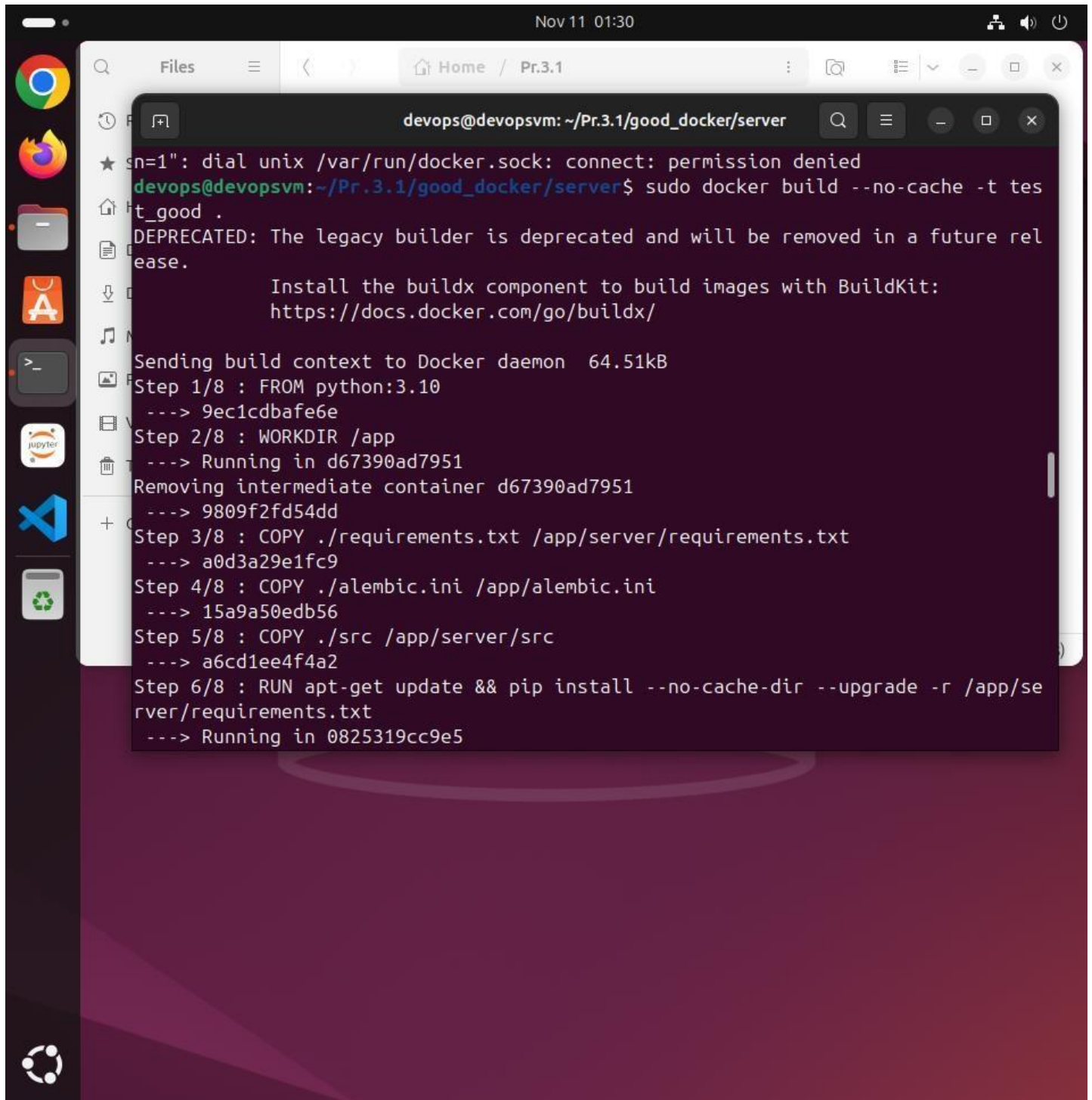


Рис.10

Исходный код

Содержание плохого dockerfile

```
from python:latest workdir
```



```
/app copy ./requirements.txt
/app/server/requirements.txt copy
./alembic.ini /app/alembic.ini copy ./src /app/server/src run pip install
--no-cache-dir --upgrade -r /app/server/requirements.txt cmd
["uvicorn", "server.src.main:app", "--host", "0.0.0.0", "--port",
"8000"] volume /app/data
```

Содержание хорошего dockerfile

```
FROM python:3.10
WORKDIR /app
COPY ./requirements.txt /app/server/requirements.txt
COPY ./alembic.ini /app/alembic.ini
COPY ./src /app/server/src
RUN apt-get update && pip install --no-cache-dir --upgrade -r
/app/server/requirements.txt
CMD ["uvicorn", "server.src.main:app", "--host", "0.0.0.0", "--port", "8000"]
VOLUME /app/data
```

Анализ различий между подходами

В плохом докерфайле использовался python тег latest , из за чего не контролируется версия образа, что в дальнейшем может привести к изменениям и несовместимостям, если latest обновится

RUN pip install, зависимости устанавливаются без обновления, может привести к проблемам с зависимостями. Необходимо перед установкой зависимостей обновить пакетный менеджер **RUN apt-get update && pip install**.

Заключение

После выполнения практической работы, были получены практические и теоретические навыки по работе с Docker, Dockerfile, рассмотрены «плохие» и «хорошие» практики по использованию контейнеров.