**МЕТОДИЧЕСКИЕ УКАЗАНИЯ ПО РАБОТЕ С DOCKER**

**ЧЕРЕЗ ОПЕРАЦИОННУЮ СИСТЕМУ**

**LINUX MINT CINNAMON**

Для того чтобы начать работу с Docker **необходима** установка операционной системы Linux Mint Cinammon или любой другой дистрибутив.

Если на вашем компьютере установлена в качестве **основной операционной системы** **Microsoft Windows или macOS**, необходимо установить специальную программу для создания виртуальных машин такие как VirtualBox, VMware Workstation и другие.

В данном гайде будет описана работа, непосредственно связанная с дистрибутивом **Linux Mint Cinammon через VirtualBox.**

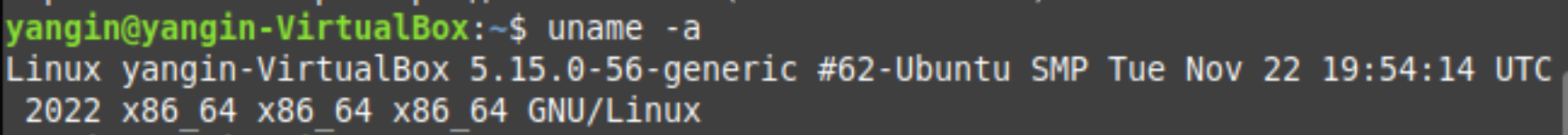
**ВАЖНО:** дальнейшая работа с Docker зависит от версии ядра Linux. Проверьте чтобы версия ядра была **НЕ НИЖЕ 3.8.**

Чтобы проверить версию ядра, откройте терминал в ОС Linux и выполните команду:

**uname -a**

Данная команда покажет всю информацию об установленной операционной системе (ее разрядность, версия ядра).

**Пример вывода:**



В данном случае мы видим что версия ядра нашей операционной системы 5.15.0-56-generic и имеет x64 разрядность.

Вторым важным шагом является **обновление пакетов через права root**. Если у вас таких прав не имеется, обратитесь к **сотруднику отдела безопасности.**

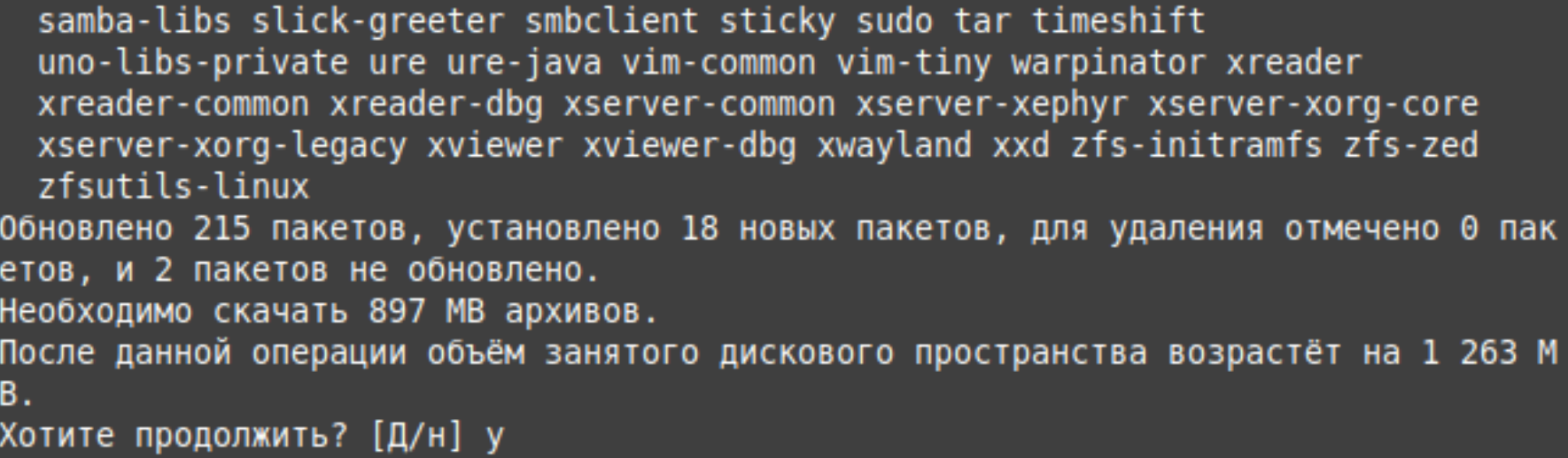
В терминале прописываем команды:

**Sudo apt-get update**

Затем,

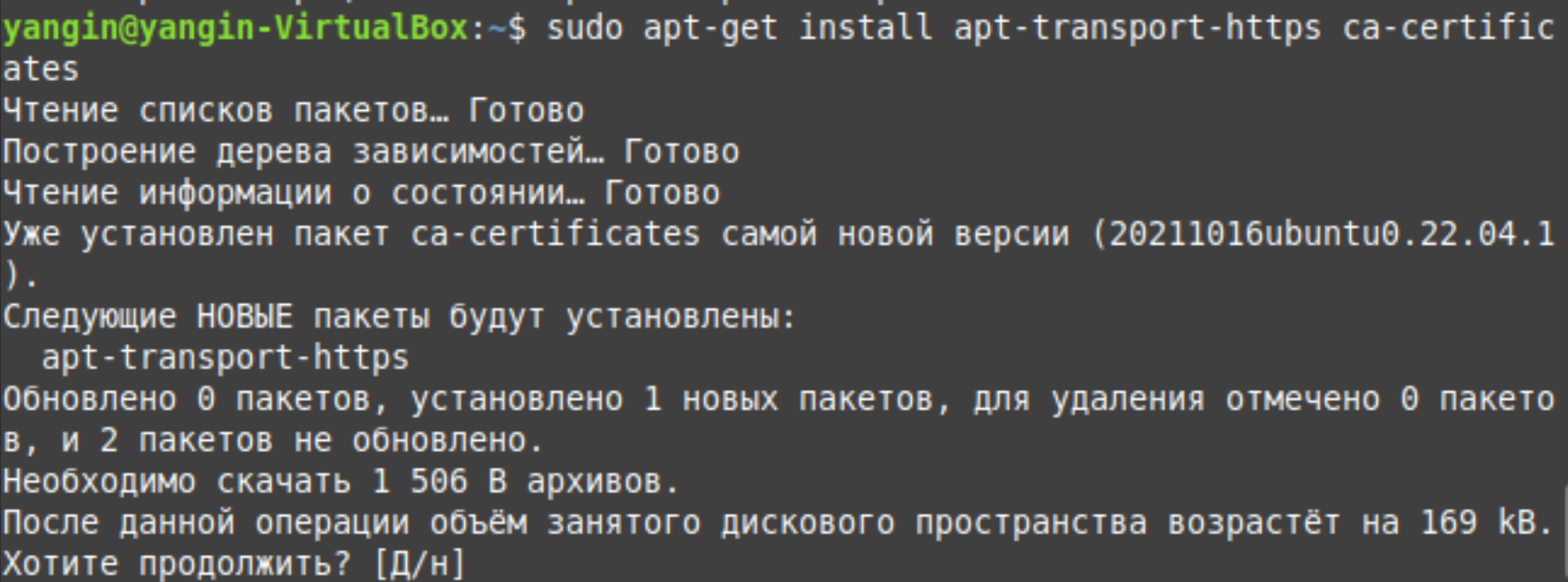
**Sudo apt-get upgrade**

При каждом обновлении пакета у вас будет запрашиваться пароль и подтверждение обновления пакетов. Затем дождитесь полного обновления пакетов.



Следующим шагом является установка необходимых сертификатов, которые потребуются для работы с сайтом Docker в дальнейшем для загрузки необходимых пакетов Docker. Это можно сделать с помощью следующей команды.

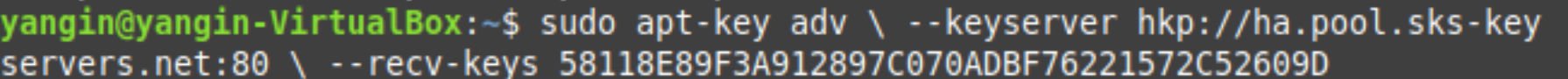
**sudo apt-get install apt-transport-https ca-certificates**



Подтвердите действие.

Следующим шагом будет добавление нового ключа GPG. Этот ключ необходим для того, чтобы все данные были зашифрованы при загрузке необходимых пакетов для Docker.

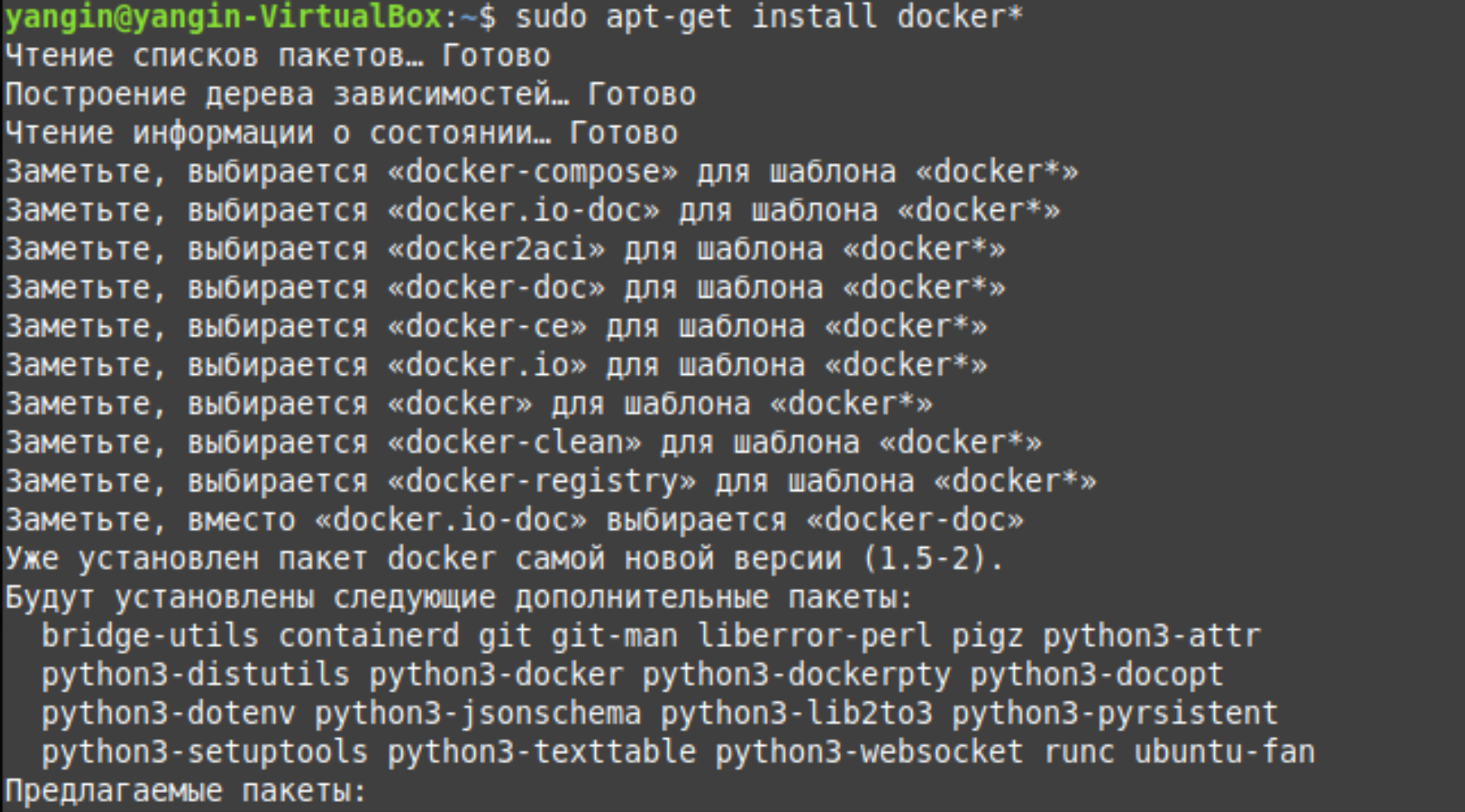
Следующая команда загрузит ключ с идентификатором 58118E89F3A912897C070ADBF76221572C52609D с сервера ключей hkp://ha.pool.sks-keyservers.net:80 и добавит его в цепочку ключей adv. Обратите внимание, что именно этот ключ требуется для загрузки необходимых пакетов Docker.



Далее устанавливаем сам Docker, через выполнение команды:

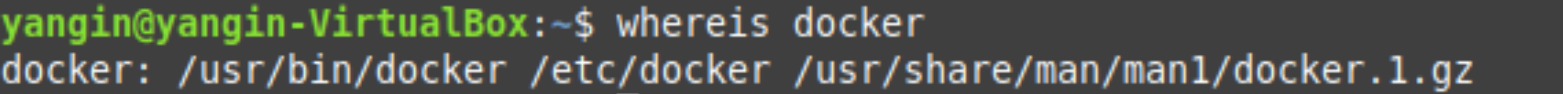
**Sudo apt-get install docker\***

Подтвердите действие.

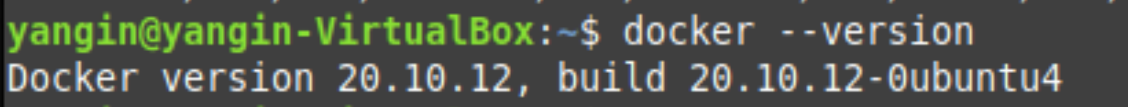


Убедимся что Docker действительно установлен в нашу систему с помощью команды:

**Whereis docker**



Посмотрим актуальную версию Docker



**Работа с Docker**

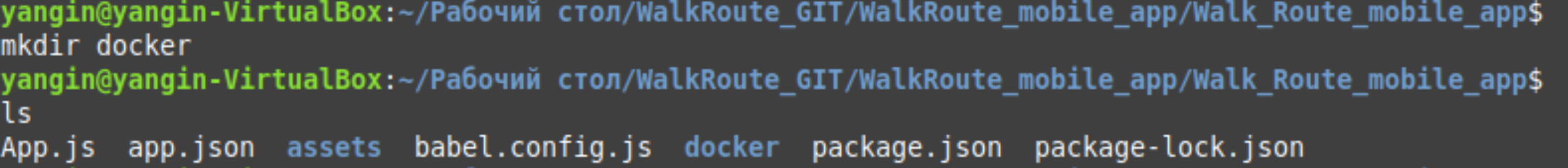
**Что такое Docker**

**Docker** — это платформа, которая позволяет упаковать в контейнер приложение со всем окружением и зависимостями, а затем доставить и запустить его в целевой системе. Приложение, упакованное в контейнер, изолируется от операционной системы и других приложений.  
Внутри контейнера находится среда, необходимая для работы. Простыми словами **контейнер — это некая изолированная песочница для запуска ваших приложений**.

Для того чтобы работать с кодом приложения установите в систему Git актуальной версии и воспользуйтесь документом **«Методические указания по работе с GIT».** Там описаны основные моменты работы с GIT, а также приведены примеры работы команд.

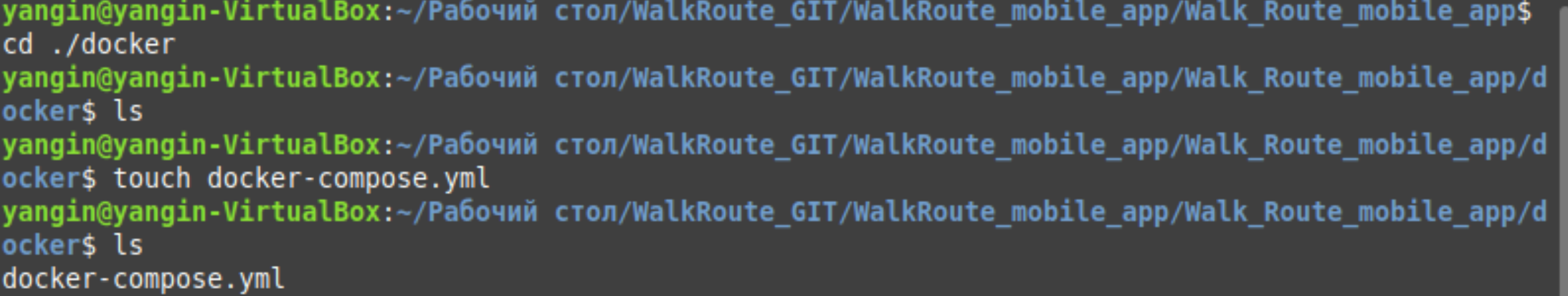
После клонирования репозитория, создаем папку внутри нашего проекта под названием docker через команду

**Mkdir <название\_папки>**



Внутри папки docker создаем файл docker-compose.yml через команду

**Touch docker-compose.yml**



Через любой текстовый редактор открываем наш ранее созданный файл и вводим следующие значения:

version: '3.8'

services:

api:

build: ./api

container\_name: api\_backend

ports:

- '4000:4000'

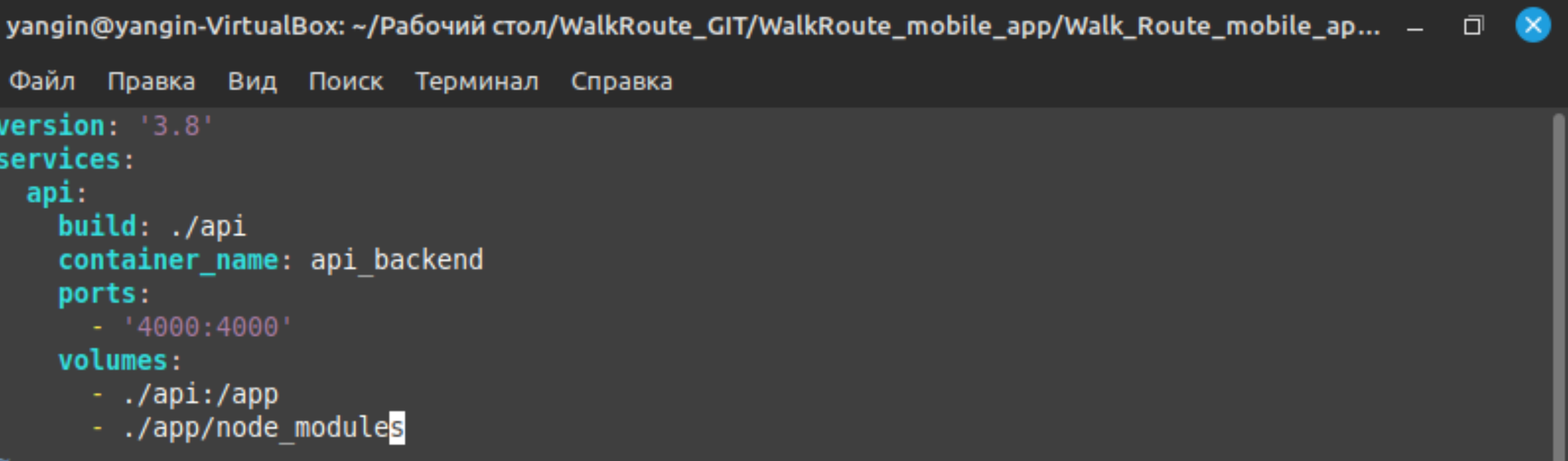
volumes:

- ./api:/app

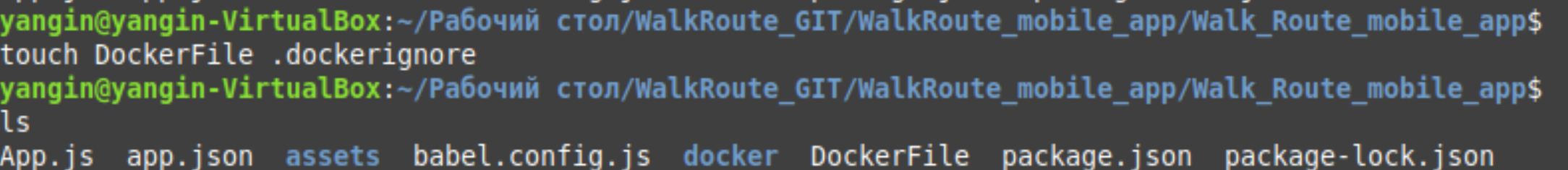
- ./app/node\_modules

Если текстового редактора нет, его можно установить с помощью команды

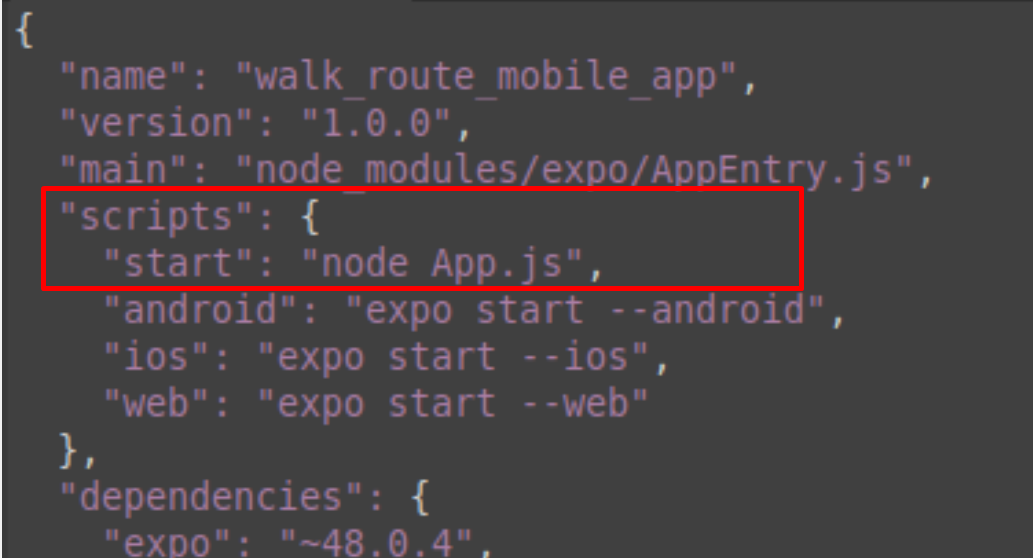
**sudo apt install vim**



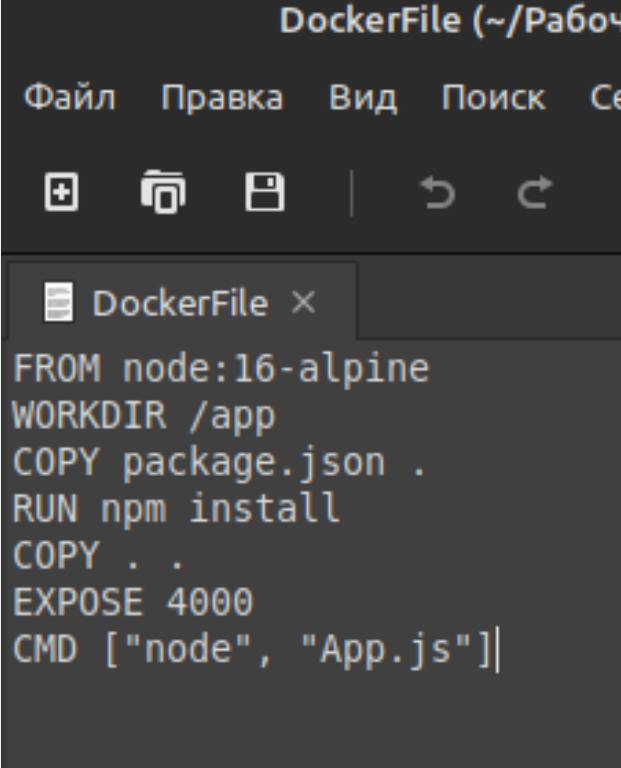
Создаем еще два файла для самого приложения:



В файле **package.json:**



Этот код – в файл **Dockerfile:**



Далее детальней описано что делает программа внутри файла **Dockerfile:**

FROM node:16-alpine

WORKDIR /app – установление рабочей папки в системе docker

COPY package.json . – копирование файла package.json

RUN npm install – установка пакета

COPY . .

EXPOSE 4000 – обозначение порта для сервера Docker на котором будет развернуто приложение

CMD ["node", "App.js"]– написание в консоли команды выполнения запуска файла App.js через библиотеку node

Эту строку – в файл **.dockerignore :**

node\_modules

В файл **App.js** вводим следующие команды:

**const** express = require('express');

**const** app = express();

**const** port = process.env.PORT || 4000;

app.get('/', (req, res) => {

res.send('Home Route');

});

app.listen(port, () =>

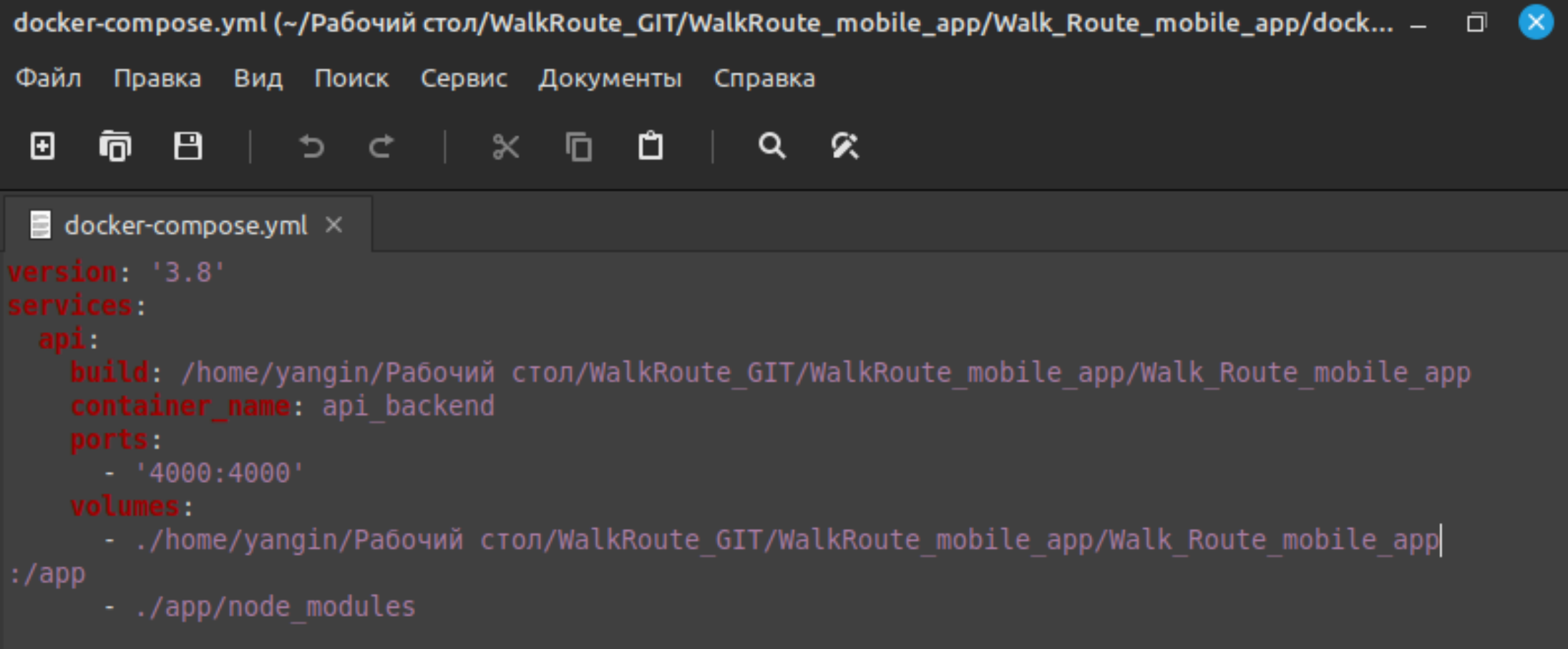
console.log(`Server running on port ${port}, http://localhost:${port}`)

);

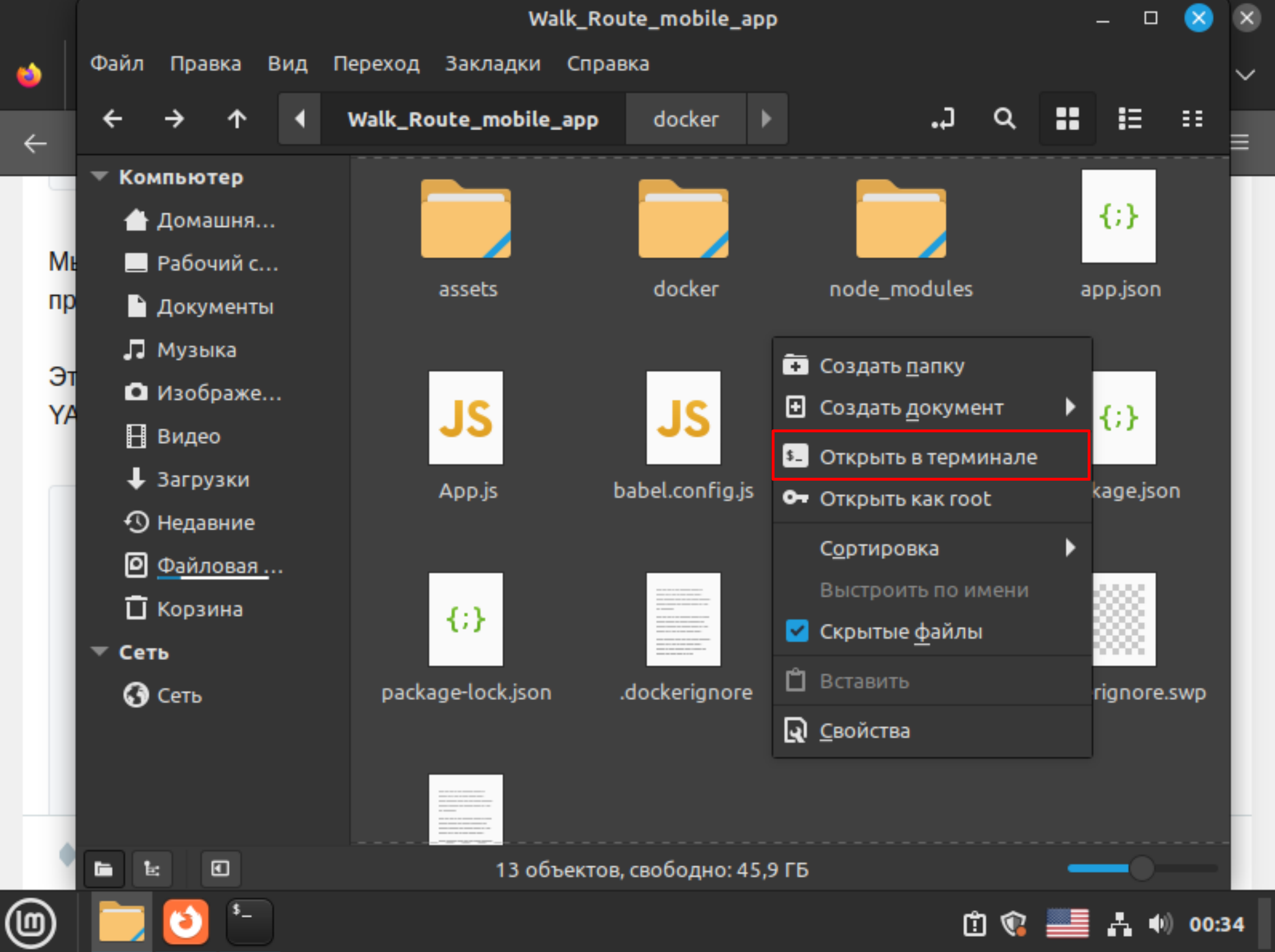
**ПРИМЕЧАНИЕ:** Если ваша папка где хранится проект называется **не api** как на скриншотах в примерах, нужно указывать **ПОЛНЫЙ ПУТЬ к** папке с проектом. Чтобы получить полный путь, достаточно ввести команду **pwd!**

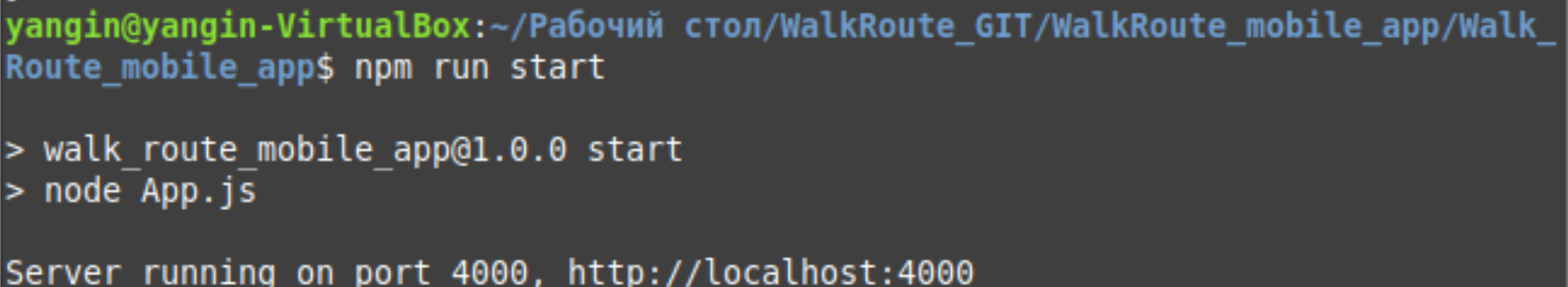
Полный путь нужно указывать в файле **docker-compose.yml**

В итоге файлы должны выглядеть примерно так:



При успешном выполнении всех пунктов, при введении в новом терминале ВНУТРИ проекта (см. скриншот) терминал выведет сообщение о запуске сервера и ссылку для просмотра:





**ПРИМЕЧАНИЕ:** не забудьте установить **npm install express**

В данном случае программа у нас выведет

