В нашей реализации будем использовать пример из руководства с сайта:

<https://github.com/YuriyCherniy/how-to-dockerize-django-app>

Для того, чтобы развернуть приложение на удаленном сервере, кроме непосредственного запуска контейнера Docker потребуется настройка реверс-прокси (в нашем случае nginx) и SSL.

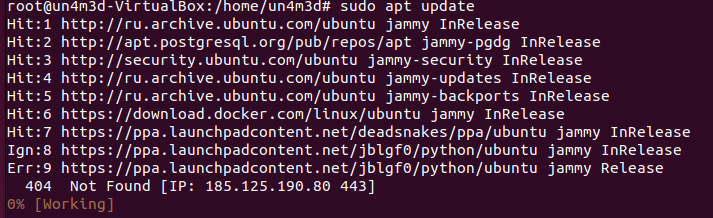
Подключаемся по ssh к удаленному серверу: ssh root@<server\_ip>

<server\_ip> - это IP-адрес удаленного сервера.

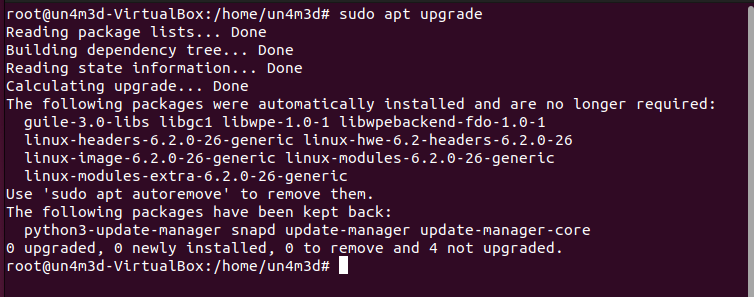
После ввода команды необходимо авторизоваться на сервере по паролю.

Обновление репозиториев и установленных пакетов в системе на удаленном сервере:

sudo apt update

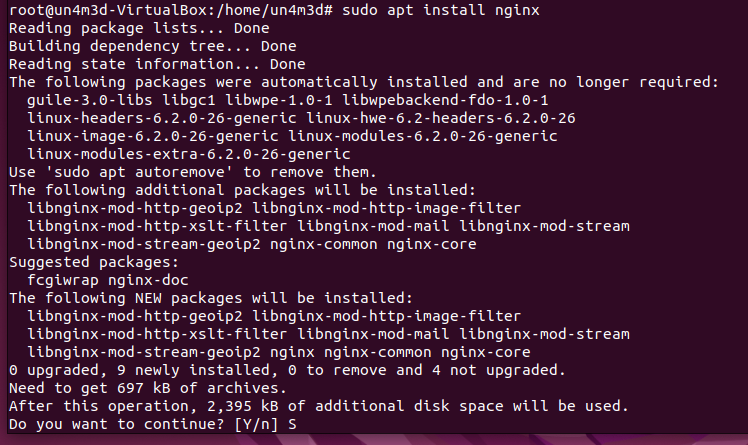


sudo apt upgrade



Установка Nginx:

sudo apt install nginx



Установка Postgresql:

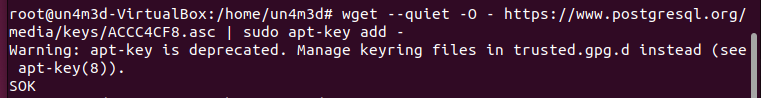
Внимание! Установка Postgresql на Ubuntu 20.04 и 22.04 имеет различия.

1. Установка Postgresql на Ubuntu 22.04:

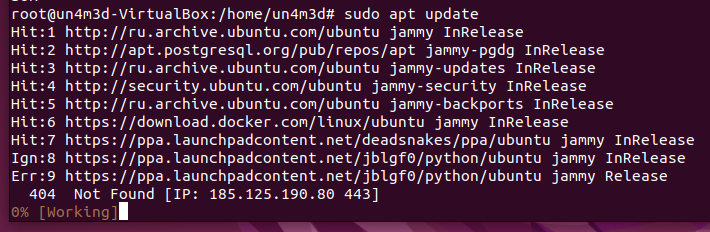
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb\_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'



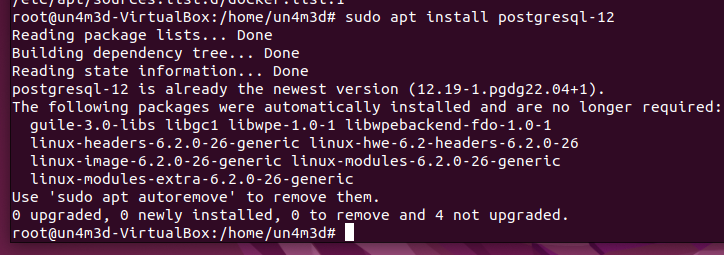
wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add -



sudo apt update



sudo apt install postgresql-12



1. Установка Postgresql на Ubuntu 20.04:

sudo apt install postgresql-12

Установка Docker и Compose:

sudo apt-get install \

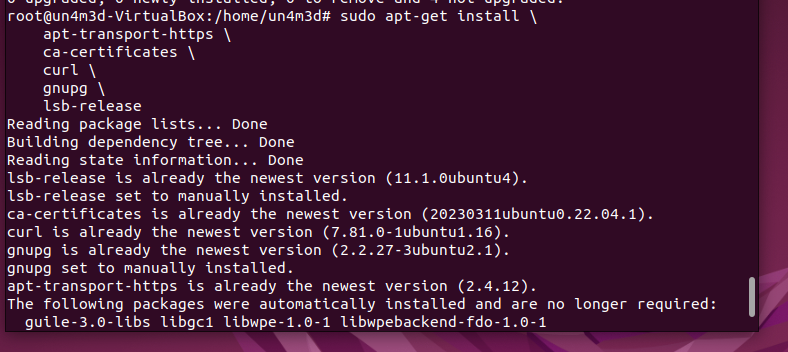
apt-transport-https \

ca-certificates \

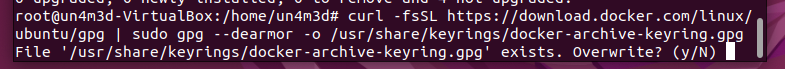
curl \

gnupg \

lsb-release



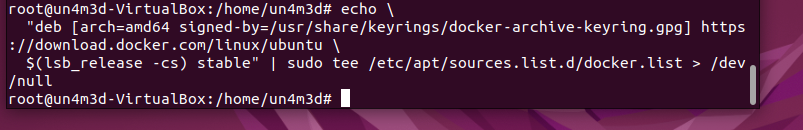
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o /usr/share/keyrings/docker-archive-keyring.gpg



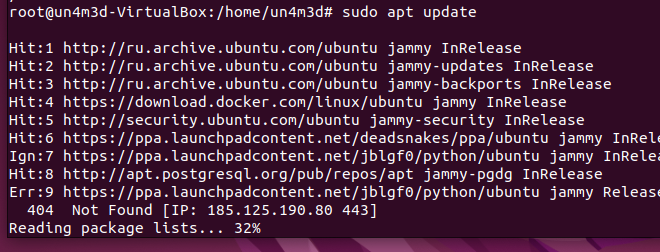
echo \

"deb [arch=amd64 signed-by=/usr/share/keyrings/docker-archive-keyring.gpg] https://download.docker.com/linux/ubuntu \

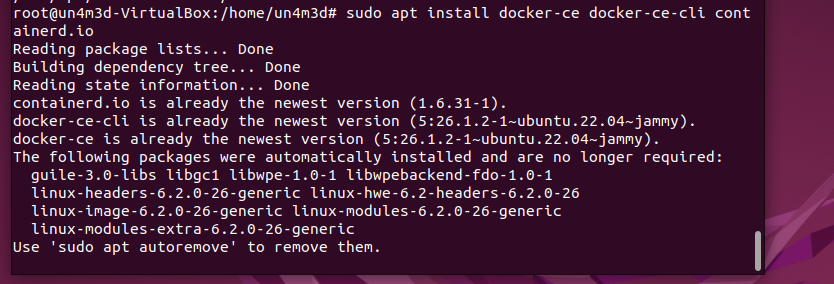
$(lsb\_release -cs) stable" | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null



sudo apt update



sudo apt install docker-ce docker-ce-cli containerd.io



Проверить установку Docker:

docker --version

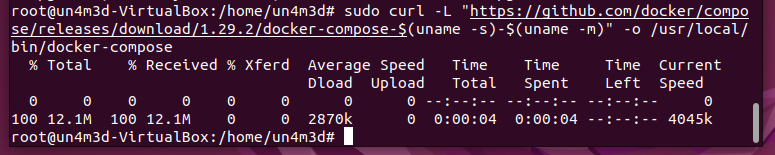
За подробностями обращайтесь к официальной документации:

<https://docs.docker.com/engine/install/ubuntu/#install-using-the-repository>

Для установки Compose выполнить следующие команды:

sudo curl -L "https://github.com/docker/compose/releases/download/1.29.2/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose

sudo chmod +x /usr/local/bin/docker-compose



Проверить установку Compose:

docker-compose –version

За подробностями обращайтесь к официальной документации:

<https://docs.docker.com/compose/install/#install-compose>

Создание и настройка базы данных:

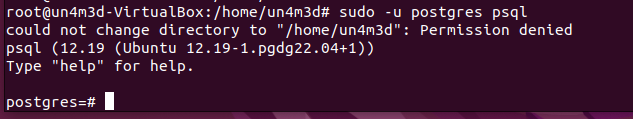
- запустить консоль psql: sudo -u postgres psql

- создать базу данных: CREATE DATABASE <db\_name>;

- создать пользователя: CREATE USER <username> WITH PASSWORD '<password>';

- предоставить пользователю административный доступ к базе данных: GRANT ALL PRIVILEGES ON DATABASE <db\_name> TO <username>;

Замените <db\_name> на test\_db, <username> на user\_postgres, <password> на 0000.







Настройка Docker-контейнера:

Файл Dockerfile содержит инструкции по созданию Docker образа на основе которого будет создаваться и запускаться контейнер с Django приложением. Вот так выглядит наш:

FROM python:3.9.5-slim

ENV PYTHONUNBUFFERED=1

WORKDIR /app

COPY requirements.txt requirements.txt

RUN pip3 install -r requirements.txt

COPY . .

RUN ["chmod", "+x", "docker-entrypoint.sh"]

ENTRYPOINT [ "./docker-entrypoint.sh" ]

Образ будет создан на основе официального образа python:3.9.5-slim, туда будет скопирован файл requirements.txt, затем установятся все зависимости. Далее будут скопированны файлы нашего приложения и выполнены инструкции из файла docker-entrypoint.sh. Вот его содержимое:

#!/bin/bash

# Collect static files

echo "Collect static files"

python manage.py collectstatic --noinput

# Apply database migrations

echo "Apply database migrations"

python3 manage.py migrate

# Start server

echo "Starting server"

gunicorn app\_to\_dockerize.wsgi:application --bind 0.0.0.0:8000 --workers 3

Последняя строка запустит сервер приложений gunicorn для обслуживания нашего Django проекта. Управлять Docker будем с помощью docker-compose. Инструкции для docker-compose описываются в файле docker-compose.yml. Вот содержимое нашего:

version: "3.9"

services:

web:

build: .

volumes:

- ./static:/app/static

network\_mode: "host"

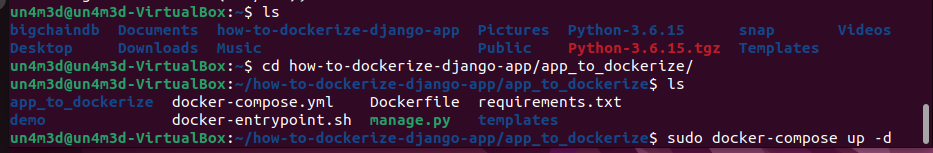
restart: unless-stopped

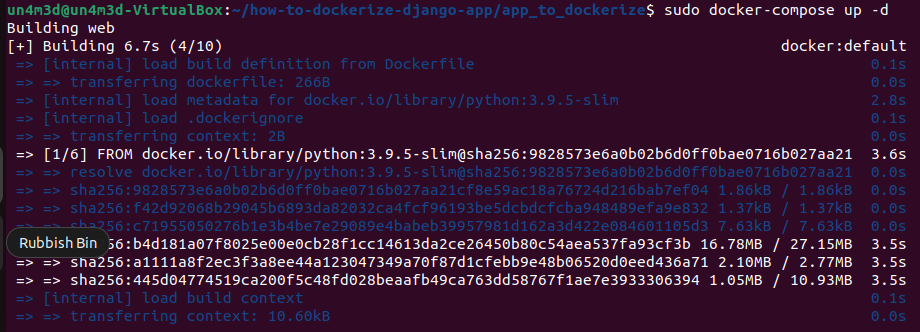
Compose смотрит в Dockerfile, строит на его основе образ, а также определяет папку на хосте, куда будут собраны статические файлы. network\_mode: "host" говорит о том, что сеть Docker контейнера будет открыта локальному хосту, без этого Nginx не увидит сеть контейнера. restart: unless-stopped говорит о том, что при перезагрузке сервера Docker контейнер будет стартовать автоматически, пока не будет остановлен намеренно.

Создание и старт контейнера

* перейти в домашнюю директорию: cd ~
* скачать Django приложение на сервер: git clone <https://github.com/YuriyCherniy/how-to-dockerize-django-app.git>
* перейти в папку содержащую docker-compose.yml файл: cd how-to-dockerize-django-app/app\_to\_dockerize/
* запустить создание и запуск контейнера: sudo docker-compose up -d

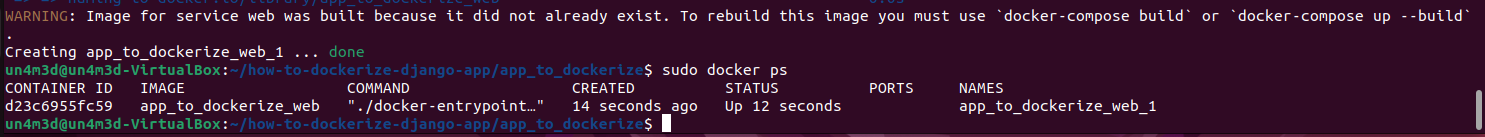
Ключь -d говорит о том, что контейнер будет запущен в detached mode. Это значит, что после создания образа и запуска контейнера, консоль будет освобождена. Если хотите видеть подробности работы приложения опустите ключ.





Проверка:

sudo docker ps



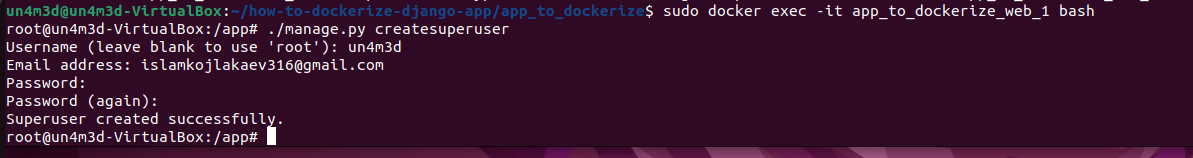
Для полноценной работы Django приложения необходимо создать супер пользователя. Для этого необходимо запустить следующую команду:

sudo docker exec -it <container\_name> bash

<container\_name> необходимо заменить на имя контейнера. Контейнер должен быть в запущенном состоянии. Имя контейнера можно узнать с помощью команды sudo docker ps. Теперь можно выполнять стандартные административные команды Django находясь на одном уровне с файлом manage.py:

- создать супер пользователя: ./manage.py createsuperuser

- вернуться в консоль сервера: exit



Настройка Nginx в качестве прокси сервере

- открыть файл:

sudo nano /etc/nginx/nginx.conf



- удалить всё и поместить следующее содержимое:

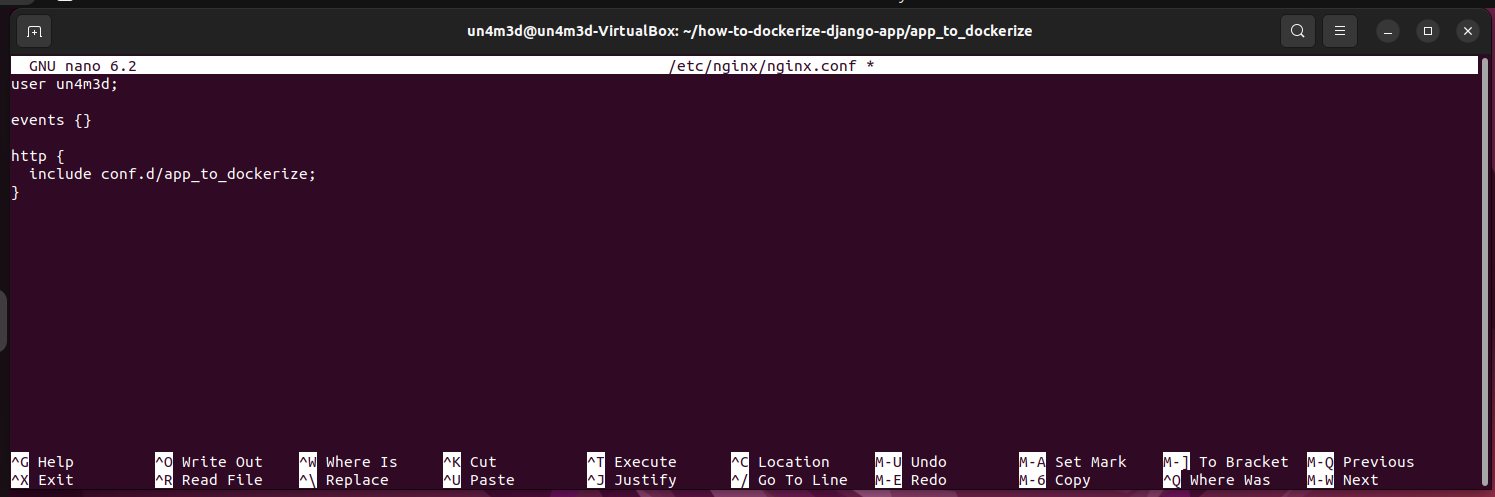
user <username>;

events {}

http {

include conf.d/app\_to\_dockerize;

}



<username> замените на имя пользователя операционной системы, без этой строки приложение работать будет, но в Ubuntu 22.04 Nginx не сможет обрабатывать статические файлы, необходимые для отображения административной панели Django.

- открыть файл:

sudo nano /etc/nginx/conf.d/app\_to\_dockerize



- поместить следующее содержимое:

include /etc/nginx/mime.types;

upstream web\_app {

server localhost:8000;

}

server {

listen 80;

server\_name <your\_domain.ru>;

location / {

proxy\_pass http://web\_app;

proxy\_set\_header X-Forwarded-For $proxy\_add\_x\_forwarded\_for;

proxy\_set\_header Host $host;

proxy\_set\_header X-Forwarded-Proto $scheme;

proxy\_redirect off;

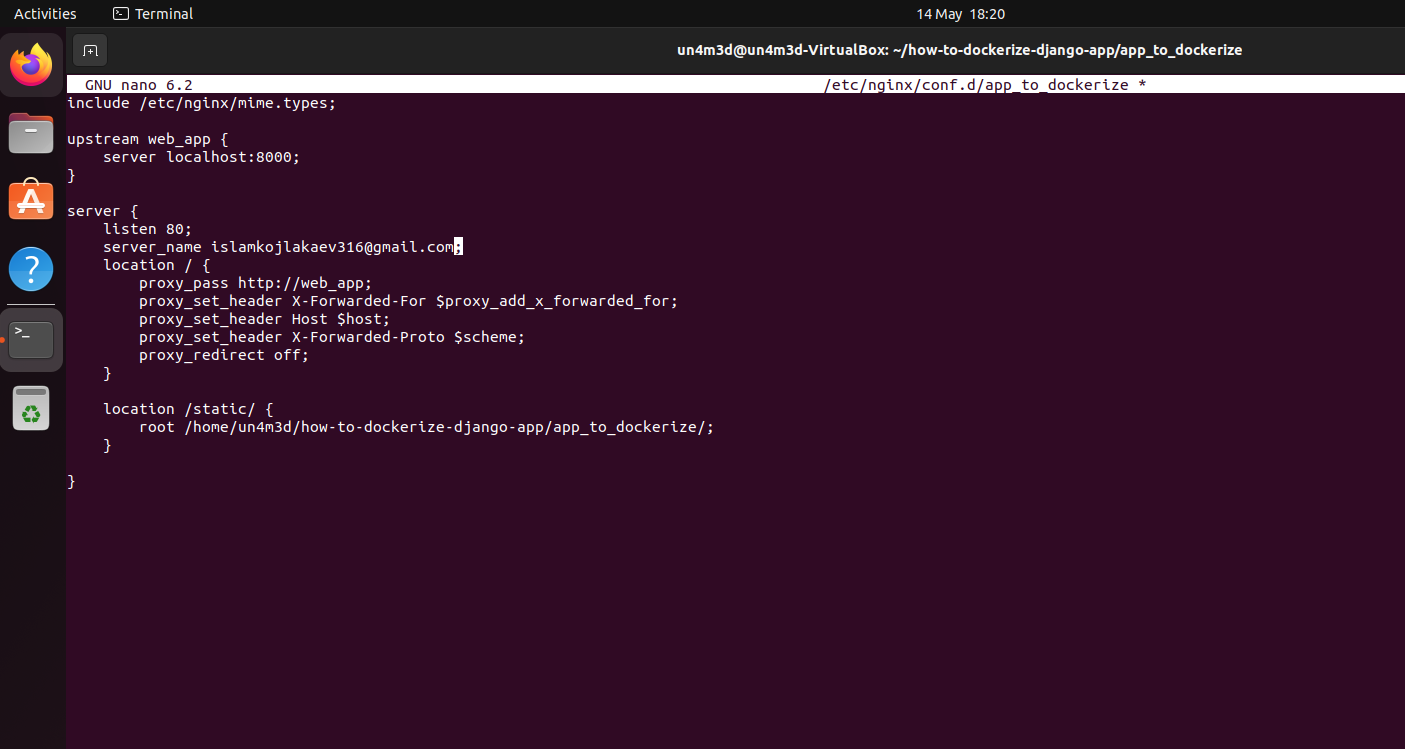
}

location /static/ {

root /home/<username>/how-to-dockerize-django-app/app\_to\_dockerize/;

}

}



<your\_domain.ru> замените на ваше доменное имя. <username> замените на имя пользователя операционной системы.

Проверить конфигурационные файлы Nginx на синтаксические ошибки:

sudo nginx -t



Перезагрузить конфигурационные файлы Nginx:

sudo nginx -s reload

Если всё было сделанно правильно сайт должен открываться на вашем домене по http соединению.

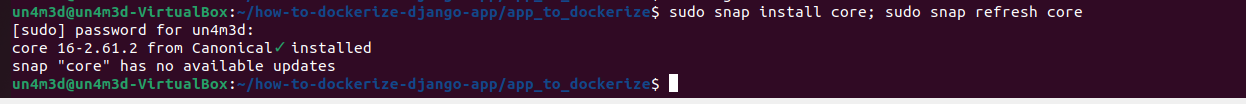


Получение SSL сертификата от Let's Encrypt

и настройка автоматического обновления

Получение и обновление сертификата будет происходить в автоматическом режиме с помощью ACME-клиента certbot, для этого выполните следующие команды:

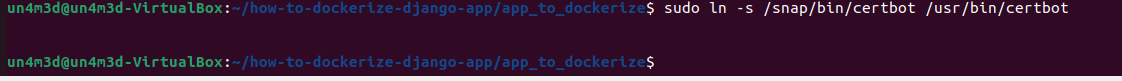
sudo snap install core; sudo snap refresh core



sudo snap install --classic certbot



sudo ln -s /snap/bin/certbot /usr/bin/certbot



sudo certbot --nginx



Ошибка на скрине выше связана с тем, что мы не зарегистрировали доменное имя (для успешного запуска тестового приложения понадобится доменное имя, также необходимо в настройках DNS Вашего домена создать запись, указывающую на IP адрес Вашего удаленного сервера).

Проверить возможность автоматического обновления сертификата:

sudo certbot renew --dry-run

Больше подробностей, в официальной инструкции:

<https://certbot.eff.org/lets-encrypt/ubuntufocal-nginx>

Если всё сделанно верно, сайт будет доступен по защищённому протоколу https:

<https://your_domain.ru>