



[Главная](#) >> [Инструкции](#) >> Использование Docker для чайников

# Использование Docker для чайников

Обновлено: 08 апреля 2021 Опубликовано: 29 июня, 2020 от [admin](#), 17 комментариев, время чтения: 12 минут

Обнаружили ошибку в тексте? Сообщите мне об этом. Выделите текст с ошибкой и нажмите **Ctrl+Enter**.

В одной из прошлых статей мы рассматривали [как запустить контейнер Docker](#) из образа. Основная идея Docker в том, что для каждого отдельного процесса должен быть создан отдельный контейнер Docker с окружением, нужным этому процессу. Но для сложных приложений здесь кроется проблема.

Например, для веб-приложения уже нужна база данных, веб-сервер, и возможно ещё интерпретатор PHP. Это уже три контейнера, настраивать и запускать их вручную не удобно, поэтому была придумана утилита docker-compose, которая позволяет управлять группами контейнеров, создавать их, настраивать, а также удалять одной командой. В этой статье мы разберемся как пользоваться docker для чайников. Подробно рассмотрим docker-compose, а также реальное применение утилиты.

## Содержание статьи

- [Использование Docker для чайников](#)
  - [1. Установка docker-compose](#)
  - [2. Создание проекта](#)

Конфиденциальность · Условия использования

Privacy

- [3. Добавление контейнеров](#)
- [4. Запуск контейнеров](#)
- [5. Порты контейнера](#)
- [6. Монтирование папок](#)
- [7. Настройка хранилищ](#)
- [8. Настройка сети](#)
- [9. Модификация контейнера](#)
- [10. Подключение к контейнеру](#)
- [Выводы](#)

## Использование Docker для чайников

Поскольку эта инструкция про Docker для начинающих, я рекомендую сразу прочитать статью про [запуск контейнера](#), вы поймете некоторые основы Docker, которые могут здесь вам пригодится. Там рассказано как всё делать вручную, а здесь мы уже поговорим как автоматизировать этот процесс.

### 1. Установка docker-compose

Если программа ещё не установлена, её необходимо установить. Для этого надо просто скачать исполняемый файл из официального сайта разработчиков:

```
$ sudo curl -L  
"https://github.com/docker/compose/releases/download/1.25.0/docker-  
compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
```

И дать ему права на выполнение:

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

После этого вы сможете посмотреть её версию:

```
$ docker-compose --version
```

### 2. Создание проекта

Если вы уже видели проекты, использующие Docker, то, наверное, замечали, что в с проектом лежит файл под названием `docker-compose.yaml`. Именно в этом файле настраиваются контейнеры, которые надо создать для вашего проекта, потом о

Privacy

созданы автоматически с помощью docker-compose. Файл использует синтаксис YAML и должен содержать такие данные:

```
version: 'версия'
```

```
networks:
```

```
    сети
```

```
volumes:
```

```
    хранилища
```

```
services:
```

```
    контейнеры
```

Версия указывает на версию синтаксиса файла, в разных версиях доступны разные ключевые слова, это сделано для обратной совместимости. Мы будем использовать версию 3.5. Далее нужно перечислить хранилища (volumes), сети (networks) и сами контейнеры.

Синтаксис YAML похож на JSON, здесь тоже есть пары ключ: значение, разделенные двоеточием, только тут значение может быть вообще нулевым, может содержать другие ключи, а также оно может быть массивом значений, тогда каждый элемент массива начинается с чёрточки "-". Но в отличие от JSON, здесь очень важны отступы, чтобы показать вложенность значений, поэтому не теряйте их.

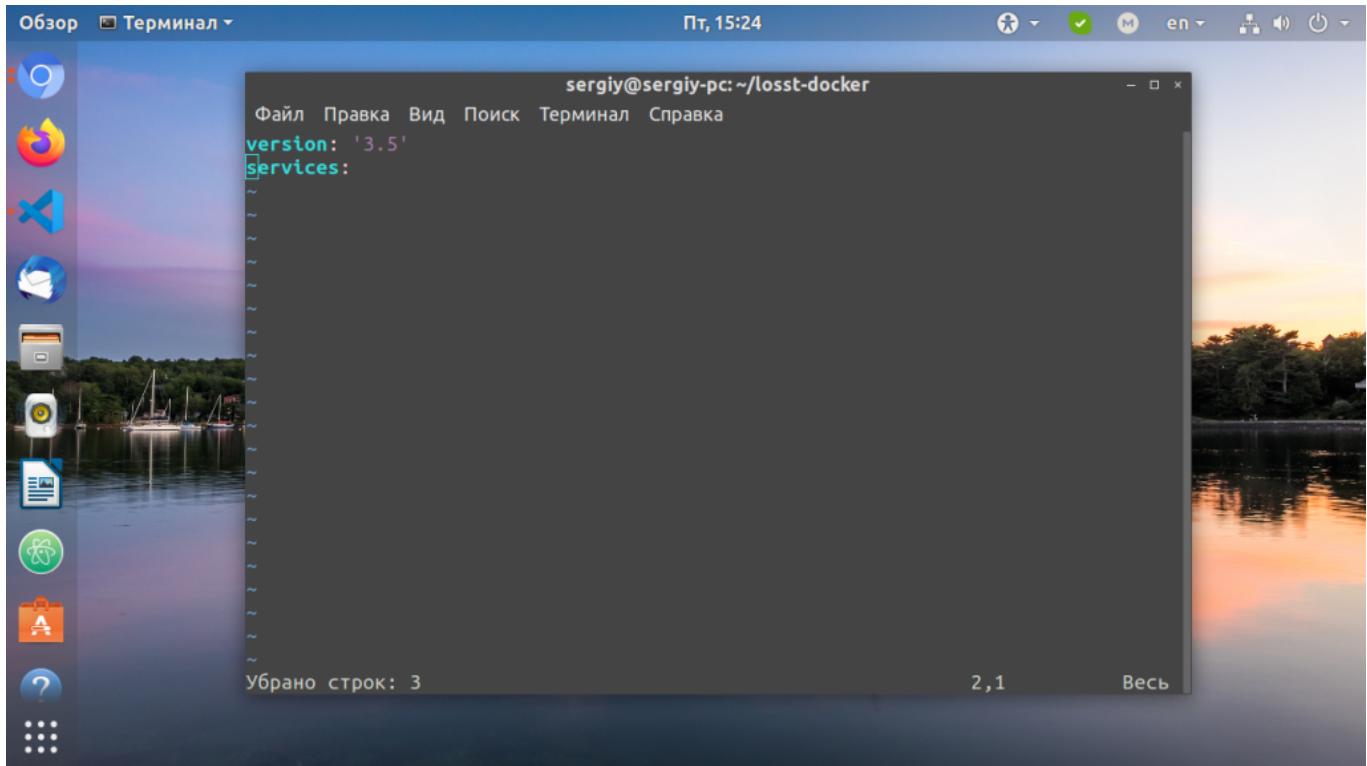
Давайте создадим папку losst-docker и создадим в ней файл docker-compose.yaml:

```
$ mkdir losst-docker
```

```
$ vi losst-docker/docker-compose.yaml
```

```
version: '3.5'
```

```
services:
```



### 3. Добавление контейнеров

Рассмотрим содержимое самого простого пункта настройки контейнера:

**имя\_контейнера:**

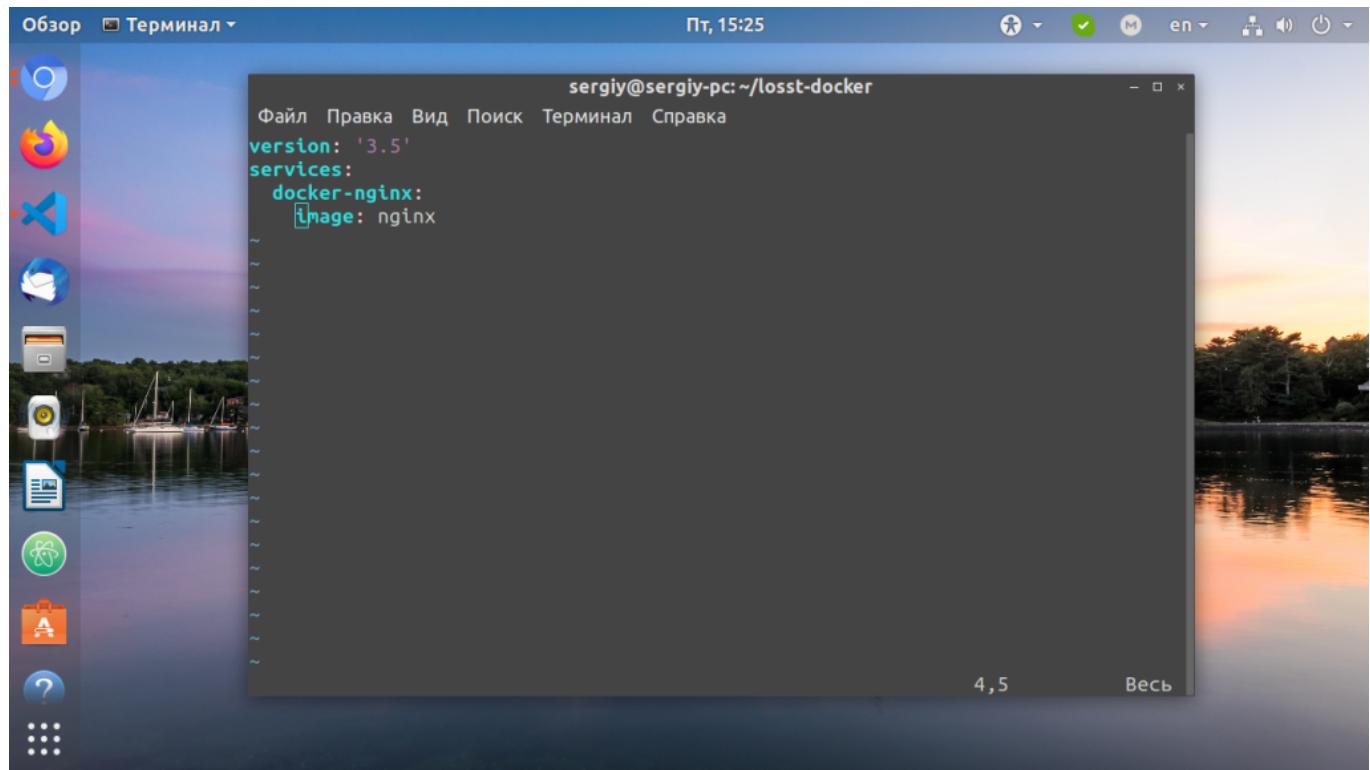
**image: образ:версия**

Здесь нам обязательно надо указать имя будущего контейнера, а также образ, на основании которого он будет создан. Через двоеточие можно указывать версию контейнера. Версии можно посмотреть на [Dockerhub](#) они там отмечены как tags. Если версия не указана используется **latest**, последняя.

Например, добавим контейнер для веб-сервера Nginx:

```
docker-nginx:
```

```
  image: nginx
```



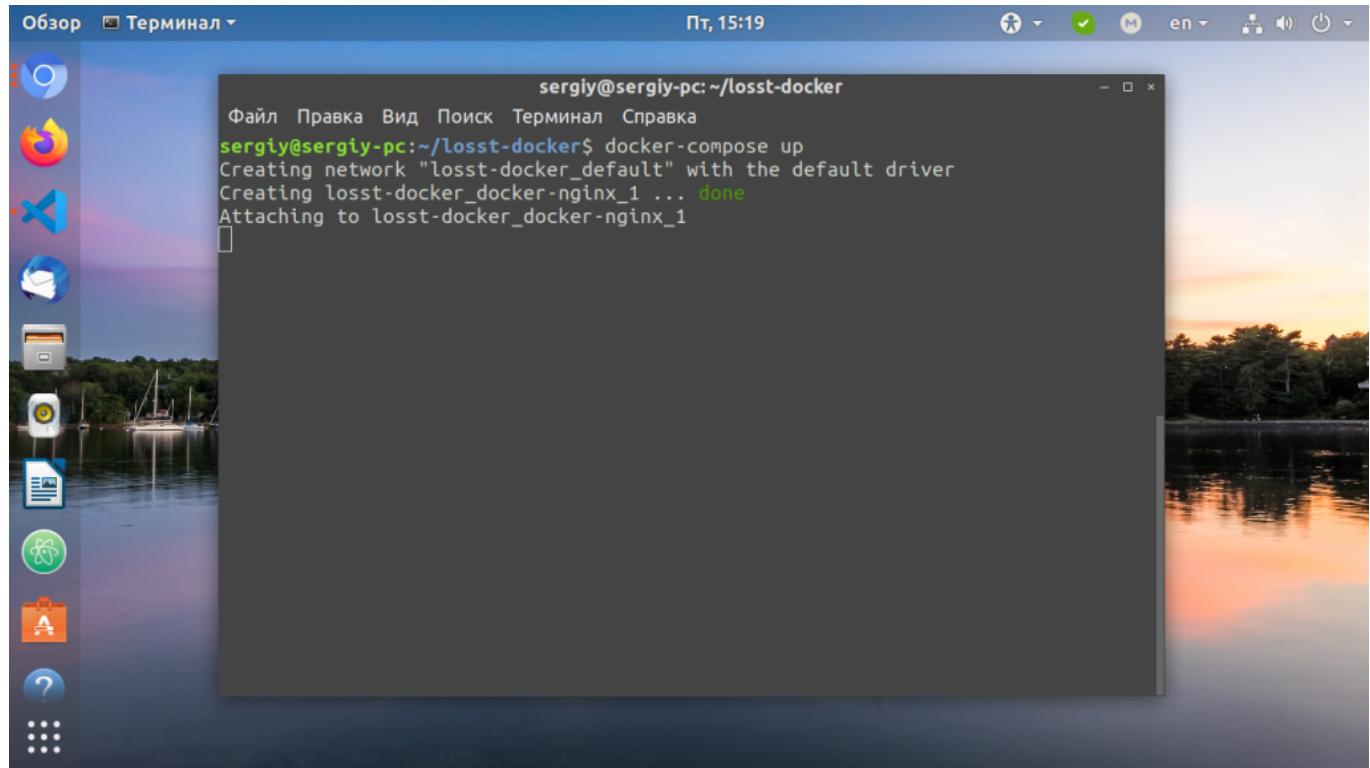
Уже имея эти данные в конфигурационном файле можно запускать контейнер.

## 4. Запуск контейнеров

Когда настройка docker завершена, надо запускать полученные контейнеры. Чтобы запустить группу контейнеров, настроенную в docker-compose.yaml необходимо перейти в папку, где находится этот файл конфигурации и выполнить там команду docker-compose up. Например:

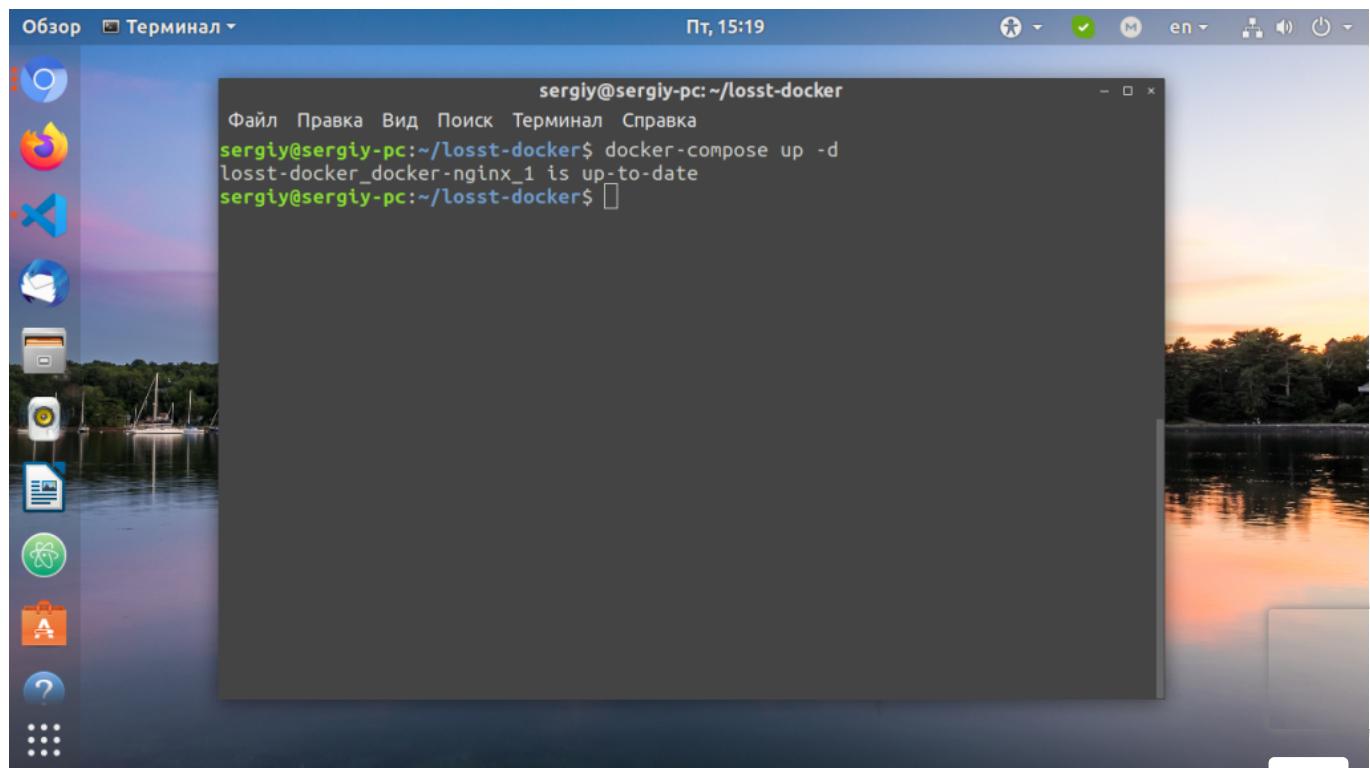
```
$ cd losst-docker
```

```
$ docker-compose up
```



После этого контейнеры будут запущены, все их потоки вывода будут объединены в один и вам будет выводится информация в терминал. Чтобы остановить контейнеры достаточно нажать **Ctrl+C**. Если вы хотите запустить контейнеры в фоновом режиме используйте опцию **-d**:

```
$ docker-compose up -d
```



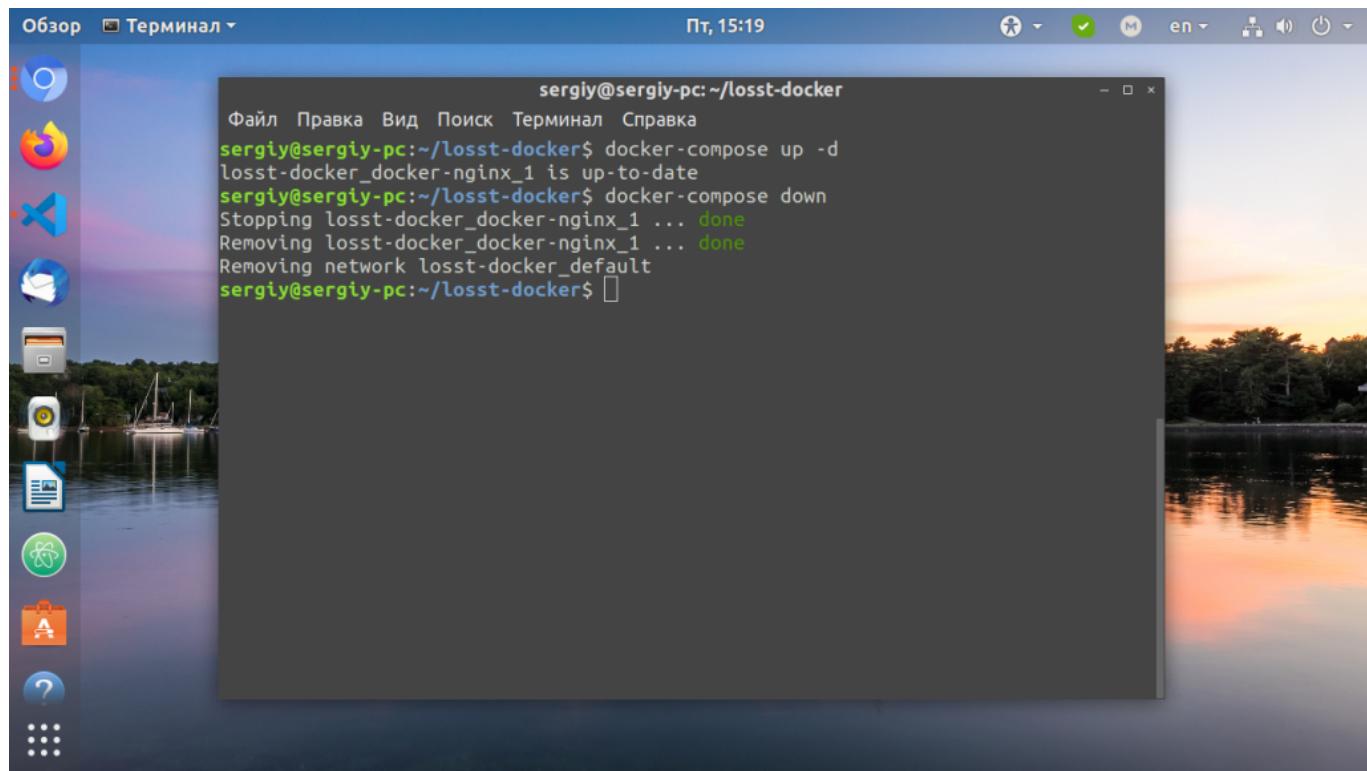
Остановить контейнеры, запущенные в фоновом режиме можно командой **stop**:

Privacy

```
$ docker-compose stop
```

Команда **down** не просто останавливает все запущенные контейнеры, но и удаляет их:

```
$ docker-compose down
```



Остановите пока этот контейнер, мы продолжим его настройку.

## 5. Порты контейнера

Контейнер работает, но толку пока нам от него мало. С помощью `docker` мы можем пробросить порт 80 контейнера в основную операционную систему и получить к нему доступ. Для этого используйте директиву `ports`. Синтаксис такой:

### `ports:`

- `внешний_порт:внутренний порт`

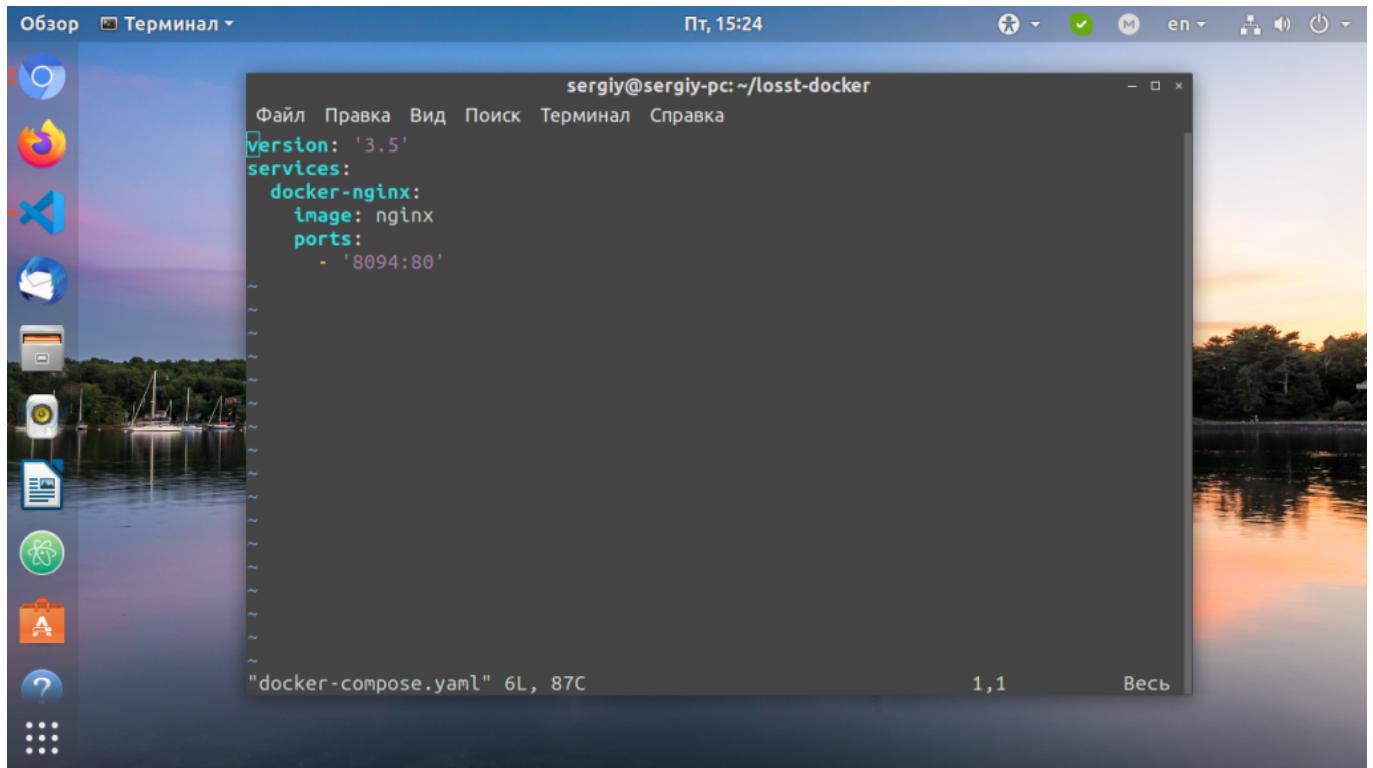
Например, пробросим порт 80 как 8094:

```
docker-compose:
  services:
    nginx:
      image: nginx
```

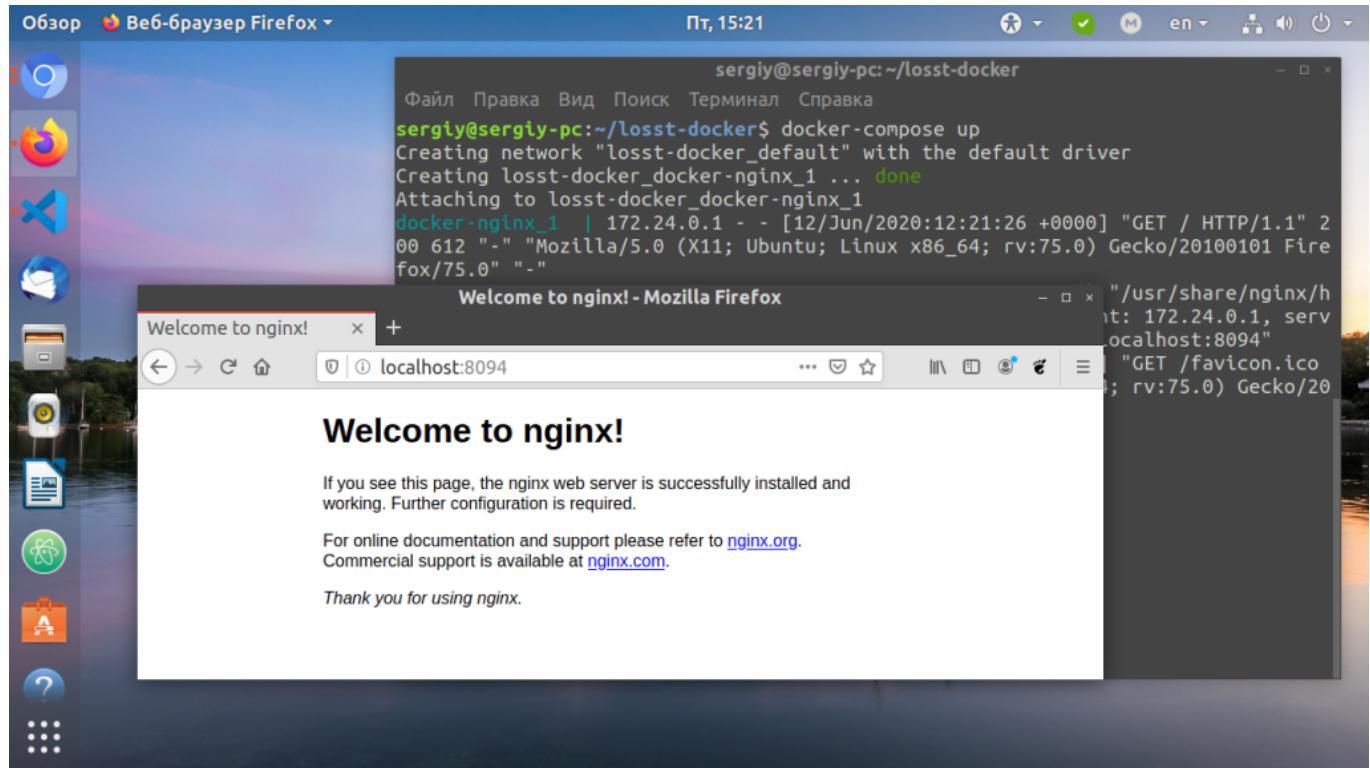
Privacy

```
ports:
```

```
- '8094:80'
```



Теперь вы можете перезапустить контейнеры и увидеть в браузере страницу, сообщающую о том, что Nginx работает по адресу <http://localhost:8094>:



Но это все ещё не интересно, потому что мы не можем размещать там свои файлы. Сейчас это исправим.

## 6. Монтирование папок

Для монтирования хранилищ или внешних папок хоста используется пункт `volumes`. Синтаксис очень похож на работу с портами:

### `volumes:`

- /путь/к/внешней/папке:/путь/к/внутренней/папке

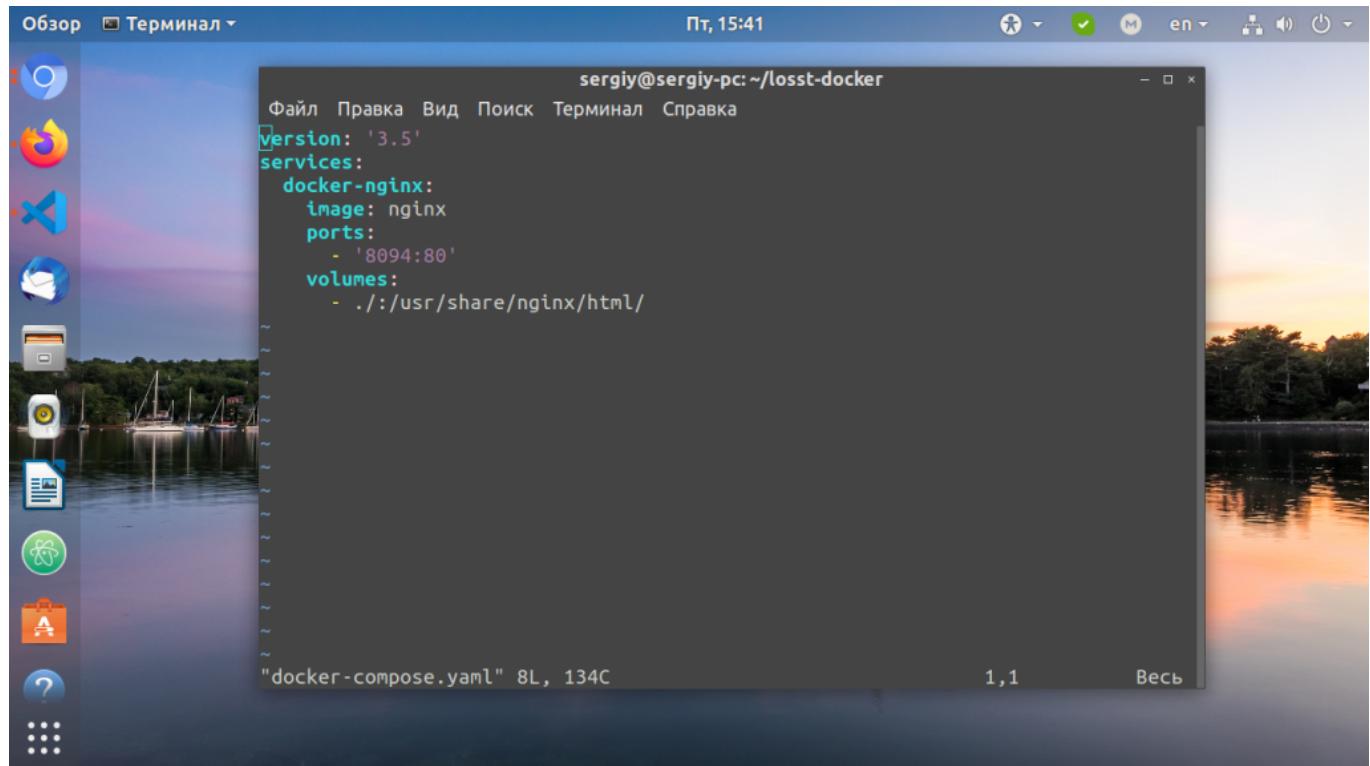
Например, давайте создадим в текущей папке проекта файл `index.html` и смонтируем эту папку вместо папки `/usr/share/nginx/html/` контейнера:

```
$ vi index.html
```

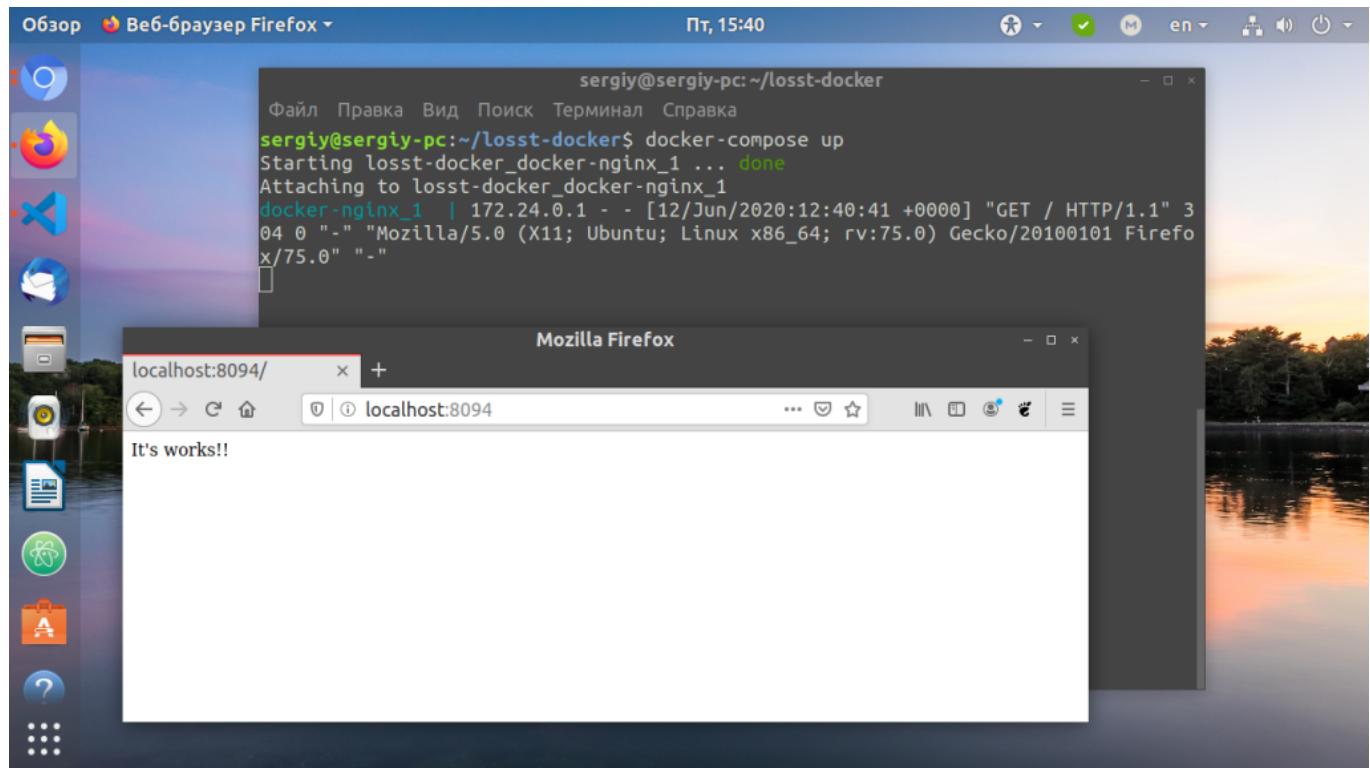
```
<html>It's works!</html>
```

```
$ vi docker-compose.yaml
```

```
docker-nginx:  
  image: nginx  
  ports:  
    - '8094:80'  
  volumes:  
    - ./:/usr/share/nginx/html/
```



После перезапуска контейнера вы увидите в браузере уже свою страницу. Это очень мощный инструмент, с помощью которого вы можете пробрасывать в контейнер всё, начиная от исходников и заканчивая конфигурационными файлами. Очень удобно.



## 7. Настройка хранилищ

Мы можем монтировать к контейнеру не только внешние папки, но и хранилища, создаваемые в docker. Для этого сначала надо добавить хранилище в главную секцию volumes. Например **losst-vl**:

```
volumes:  
  losst-vl:
```

Большинству веб приложений необходима база данных, например, MySQL. Добавим ещё один контейнер для этой базы данных и добавим в него наше хранилище. Хранилище добавляется также, как и внешняя папка, только вместо папки указывается название хранилища:

```
docker-mysql:  
  image: mysql  
  volumes:  
    - losst-vl:/var/lib/mysql  
  environment:  
    - MYSQL_ROOT_PASSWORD=password  
    - MYSQL_DATABASE=database
```

- MYSQL\_USER=user
- MYSQL\_PASSWORD=password

```
version: '3.5'
volumes:
  losst-vl:
services:
  docker-nginx:
    image: nginx
    ports:
      - '8094:80'
    volumes:
      - ./:/usr/share/nginx/html/
  docker-mariadb:
    image: mariadb
    volumes:
      - losst-vl:/var/lib/mysql
  environment:
    - MYSQL_ROOT_PASSWORD=password
    - MYSQL_DATABASE=database
    - MYSQL_USER=user
    - MYSQL_PASSWORD=password
```

Здесь мы ещё добавили пункт `environment`, в котором задали переменные окружения для контейнера. Они необходимы, для того, чтобы указать имя базы данных и пароль `root`, которые будут использоваться по умолчанию. Как видите, с переменными окружения нет ничего сложного.

```

sergiy@sergiy-pc:~/losst-docker
Файл Правка Вид Поиск Терминал Справка
now 12 MB.
docker-mariadb_1 | 2020-06-12 13:06:59 0 [Note] InnoDB: 10.4.11 started; log sequence number 60972; transaction id 21
docker-mariadb_1 | 2020-06-12 13:06:59 0 [Note] InnoDB: Loading buffer pool(s) from /var/lib/mysql/ib_buffer_pool
docker-mariadb_1 | 2020-06-12 13:06:59 0 [Note] Plugin 'FEEDBACK' is disabled.
docker-mariadb_1 | 2020-06-12 13:06:59 0 [Warning] 'user' entry 'root@e8a3b172a071' ignored in --skip-name-resolve mode.
docker-mariadb_1 | 2020-06-12 13:06:59 0 [Warning] 'user' entry '@e8a3b172a071' ignored in --skip-name-resolve mode.
docker-mariadb_1 | 2020-06-12 13:06:59 0 [Warning] 'proxies_priv' entry '%@ root@e8a3b172a071' ignored in --skip-name-resolve mode.
docker-mariadb_1 | 2020-06-12 13:06:59 0 [Note] InnoDB: Buffer pool(s) load completed at 200612 13:06:59
docker-mariadb_1 | 2020-06-12 13:06:59 0 [Note] Reading of all Master_info entries succeeded
docker-mariadb_1 | 2020-06-12 13:06:59 0 [Note] Added new Master_info '' to hash table
docker-mariadb_1 | 2020-06-12 13:06:59 0 [Note] mysqld: ready for connections.
docker-mariadb_1 | Version: '10.4.11-MariaDB-1:10.4.11+maria~bionic' socket: '/var/run/mysqld/mysqld.sock' port: 0 mariadb.org binary distribution
docker-mariadb_1 | 2020-06-12 13:07:00+00:00 [Note] [Entrypoint]: Temporary server started.

```

## 8. Настройка сети

Контейнеры должны взаимодействовать между собой. У нас уже есть Nginx и MySQL, им пока не нужно обращаться друг к другу, но как только у нас появится контейнер для PhpMyAdmin, которому надо обращаться к MariaDB ситуация поменяется. Для взаимодействия между контейнерами используются виртуальные сети, они добавляются похожим образом на хранилища. Сначала добавьте сеть в глобальную секцию networks:

`networks:`

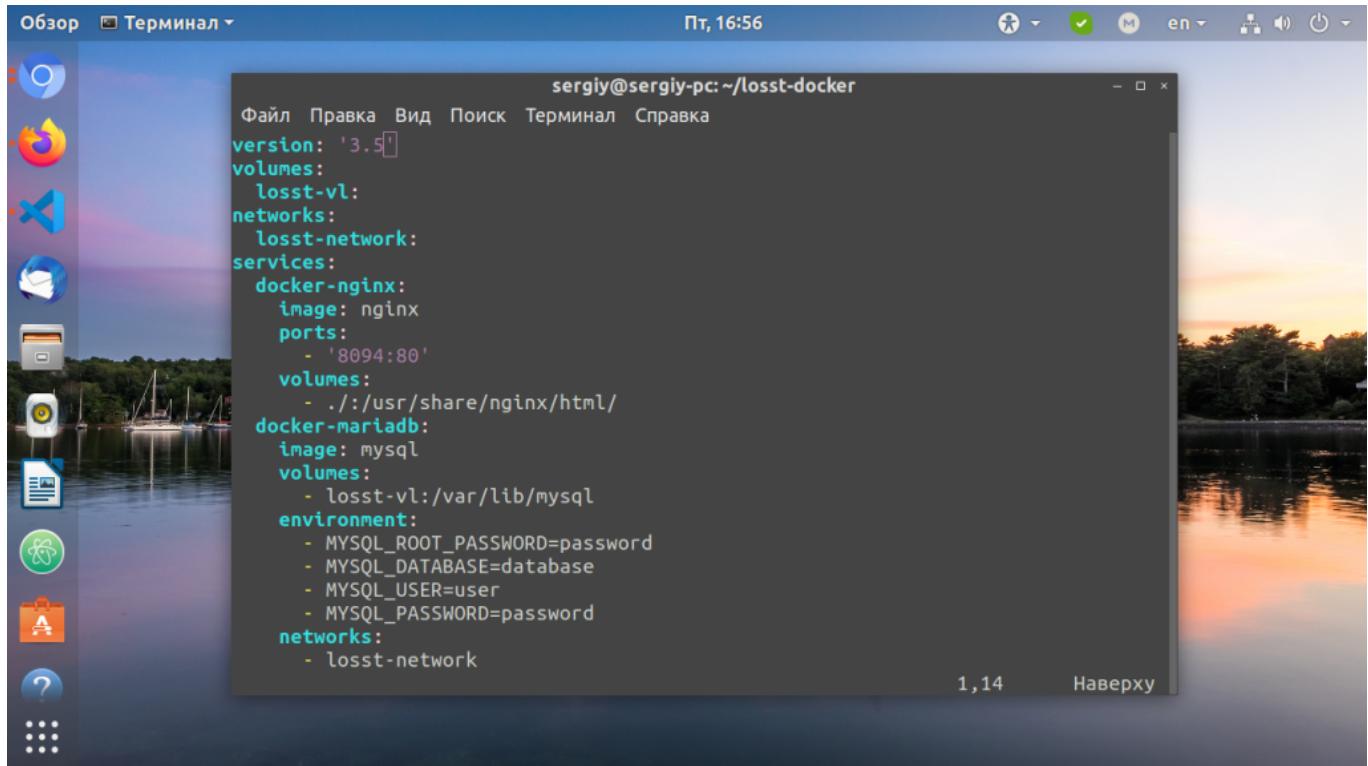
`losst-network:`

Затем для каждого контейнера, которые будут иметь доступ к этой сети, надо добавить сеть в раздел `networks`.

```
Файл Правка Вид Поиск Терминал Справка
- ./:/usr/share/nginx/html/
docker-mariadb:
  image: mysql
  volumes:
    - losst-vl:/var/lib/mysql
  environment:
    - MYSQL_ROOT_PASSWORD=password
    - MYSQL_DATABASE=database
    - MYSQL_USER=user
    - MYSQL_PASSWORD=password
  networks:
    - losst-network
docker-phpmyadmin:
  image: phpmyadmin/phpmyadmin:latest
  ports:
    - "8095:80"
  environment:
    - PMA_HOST=docker-mariadb
  networks:
    - losst-network
```

Далее контейнеры будут доступны по сети по их имени. Например, добавим PhpMyAdmin и дадим ему доступ к базе данных:

```
docker-phpmyadmin:
  image: phpmyadmin/phpmyadmin:latest
  ports:
    - "8095:80"
  environment:
    - PMA_HOST=docker-mariadb
  networks:
    - losst-network
```

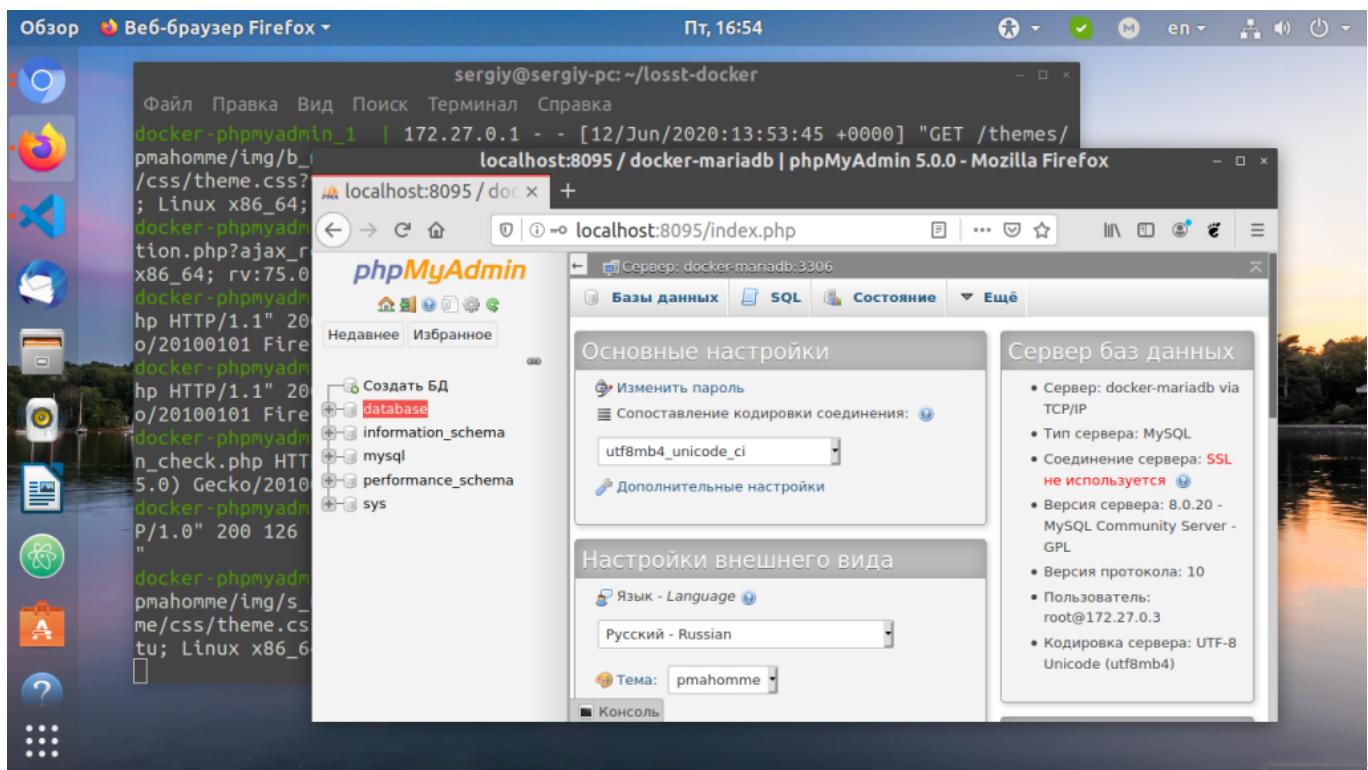


```

sergiy@sergiy-pc: ~/losst-docker
version: '3.5'
volumes:
  losst-vl:
networks:
  losst-network:
services:
  docker-nginx:
    image: nginx
    ports:
      - '8094:80'
    volumes:
      - ./:/usr/share/nginx/html/
  docker-mariadb:
    image: mysql
    volumes:
      - losst-vl:/var/lib/mysql
    environment:
      - MYSQL_ROOT_PASSWORD=password
      - MYSQL_DATABASE=database
      - MYSQL_USER=user
      - MYSQL_PASSWORD=password
  networks:
    - losst-network

```

Здесь в переменной PMA\_HOST мы ссылаемся на хост docker-mariadb, который благодаря общей сети этих контейнеров доступен. Аналогично в контейнере docker-mariadb, наш контейнер с PhpMyAdmin доступен как docker-phpmyadmin. Вы можете открыть адрес <http://localhost:8095> и авторизоваться чтобы проверить, что база данных работает:



## 9. Модификация контейнера

Это уже более сложные вещи, но зато вы разберетесь с Docker на практике. В большинстве случаев можно обойтись без модификации контейнера, иногда туда

[Privacy](#)

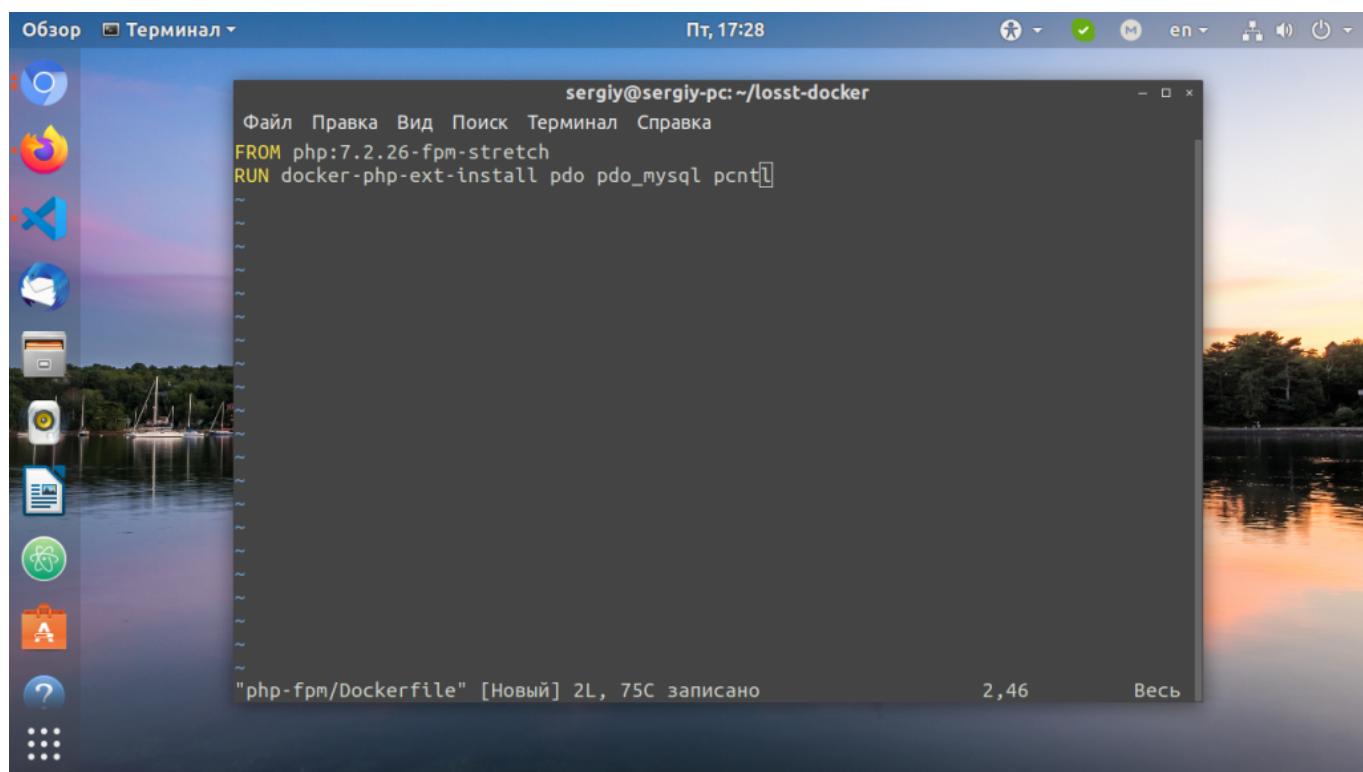
скопировать специфические конфигурационные файлы либо установить дополнительное программное обеспечение, для таких случаев docker-compose позволяет создавать свои контейнеры на основе уже существующих образов. Для этого надо создать файл Dockerfile на основе которого будет создаваться контейнер. Давайте добавим контейнер php-fpm и установим в него несколько расширений php с помощью Dockerfile.

Сначала создадим папку для файлов контейнера:

```
$ mkdir php-fpm
```

```
$ vi php-fpm/Dockerfile
```

```
FROM php:7.2.26-fpm-stretch
RUN docker-php-ext-install pdo pdo_mysql pcntl
```



Вот основные директивы, которые можно использовать в этом файле:

- **FROM** - образ, на основе которого будет создаваться наш образ;
- **RUN** - выполнить команду в окружении образа;
- **COPY** - скопировать файл в образ;
- **WORKDIR** - задать рабочую папку для образа;
- **ENV** - задать переменную окружения образа;
- **CMD** - задать основной процесс образа;

[Privacy](#)

Теперь надо добавить новую секцию в наш docker-compose.yaml. Здесь вместо image мы используем директиву build, которой надо передать путь к папке с конфигурацией образа:

```
docker-php-fpm:  
  build:  
    context: ./php-fpm  
  volumes:  
    - .:/var/www/  
  networks:  
    - losst-network
```

Дальше, раз мы уже добавили php-fpm надо прмонтировать для Nginx верный конфиг, который будет поддерживать php-fpm. Создайте папку nginx и поместите в неё такой конфигурационный файл:

```
$ mkdir nginx
```

```
$ vi nginx/site.conf
```

```
server {  
listen 80;  
server_name _ !default;  
root /var/www/;  
add_header X-Frame-Options "SAMEORIGIN";  
add_header X-XSS-Protection "1; mode=block";  
add_header X-Content-Type-Options "nosniff";  
index index.html index.htm index.php;  
charset utf-8;  
location / {  
try_files $uri $uri/ /index.php?$query_string;  
}  
error_page 404 /index.php;  
location ~ \.php$ {  
fastcgi_pass docker-php-fpm:9000;  
fastcgi_index index.php;  
fastcgi_param SCRIPT_FILENAME $realpath_root$fastcgi_script_name;
```

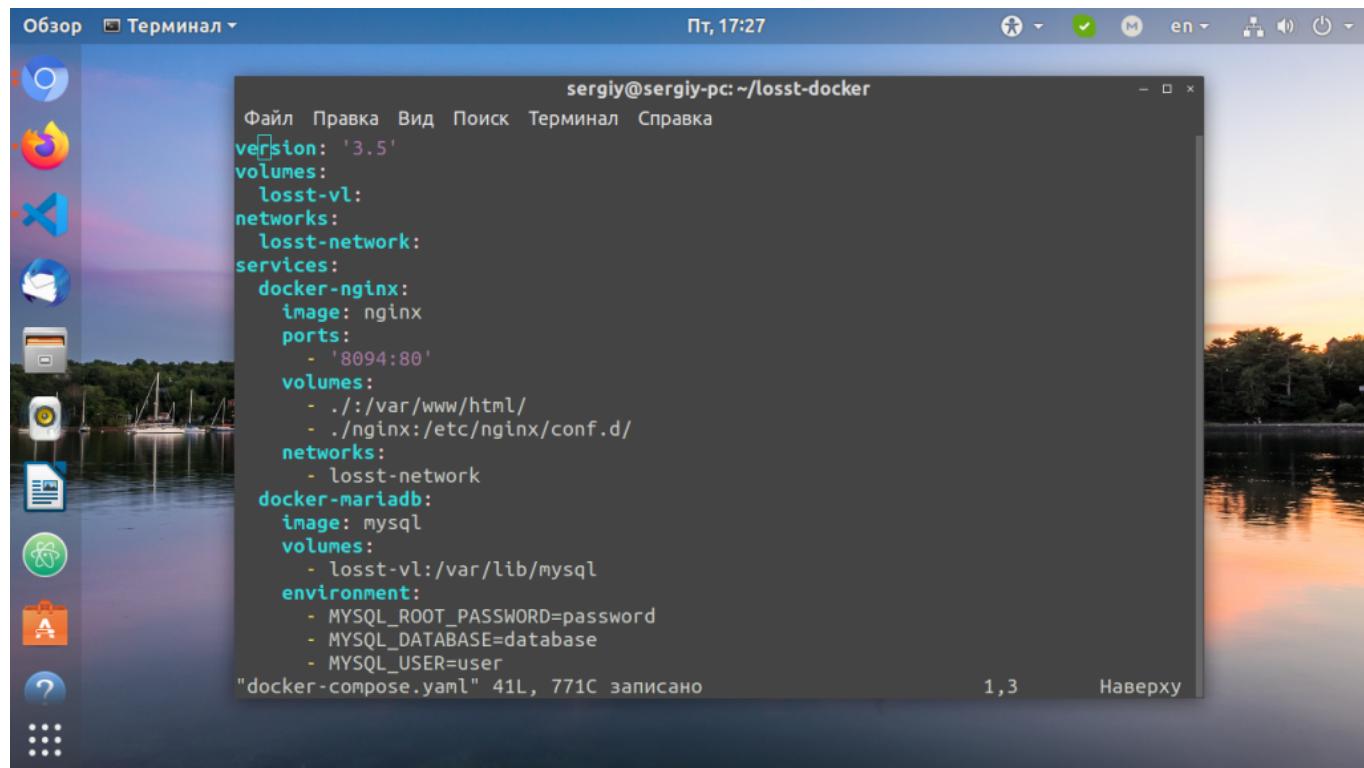
Privacy

```
include fastcgi_params;
}
}
```

Осталось немного поправить конфигурацию `nginx`. Примонтировать этот конфигурационный файл и поправить папку для веб-документов по умолчанию:

```
docker-nginx:
  image: nginx
  ports:
    - '8094:80'
  volumes:
    - ./:/var/www/html/
    - ./nginx:/etc/nginx/conf.d
  networks:
    - losst-network

```



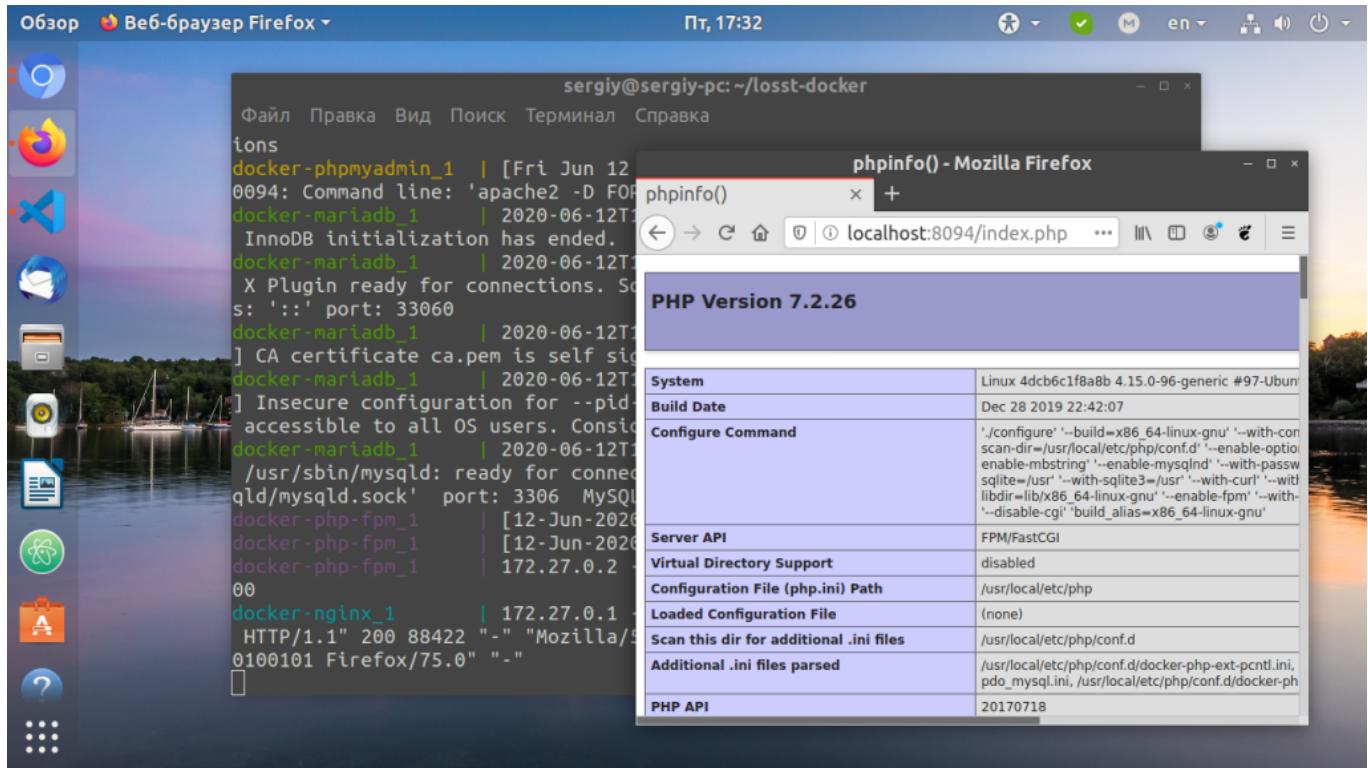
Осталось создать файл `index.php` и можно тестировать:

```
$ vi index.php
```

```
<?php phpinfo(); ?>
```

Теперь можно запускать контейнеры:

```
$ docker-compose up
```



В отличие от предыдущего раза, теперь перед запуском будет собран новый контейнер на основе файла `Dockerfile`. Такие контейнеры собираются только первый раз, если они не существуют. Чтобы их пересобрать используйте опцию `--build`:

```
$ docker-compose up --build
```

Теперь у вас есть полноценное веб-приложение, которое доступно на вашем компьютере, может использовать базу данных, PHP и много чего другого. Вы можете добавить любые недостающие вам контейнеры.

## 10. Подключение к контейнеру

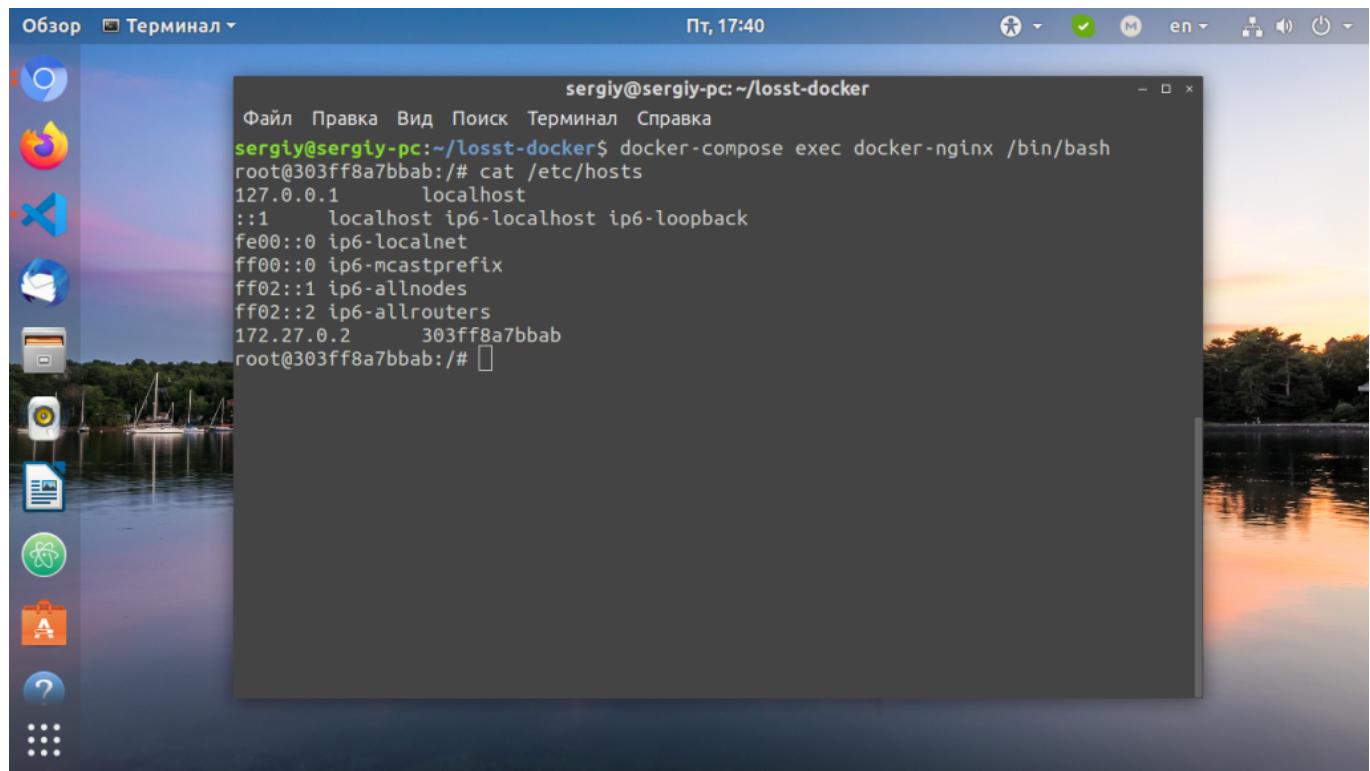
С помощью `docker-compose` вы можете подключиться к любому контейнеру из группы. Для этого просто используйте команду `exec`. Например, запустите проект в фоновом режиме:

[Privacy](#)

```
$ docker-compose up -d
```

И используйте docker-compose exec. Подключимся к контейнеру с Nginx:

```
$ docker-compose exec docker-nginx /bin/bash
```



Перед вами откроется оболочка Bash этого контейнера. Устанавливать здесь что-то вручную не рекомендуется, так как всё сотрётся после удаления контейнера, но для тестирования работы чего либо такая возможность будет очень кстати.

## Выводы

В этой статье мы разобрали использование docker для чайников. Тема довольно обширная и сложная, но очень интересная. Развернув таким образом проект на одной машине, вы сможете развернуть его там, где вам нужно просто скопировав туда конфигурацию. Поэтому эта технология завоевала такую большую популярность среди разработчиков и DevOps.

Была ли эта информация полезной для вас?

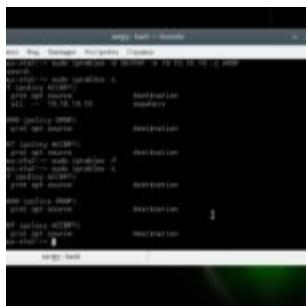
Да

Нет

X

## Похожие записи

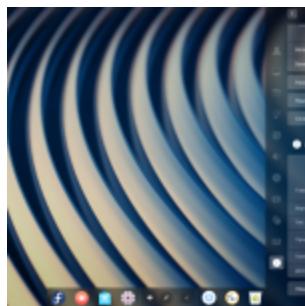
[Privacy](#)



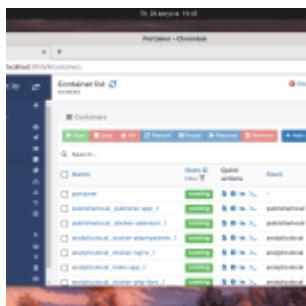
Настройка iptables для чайников



Использование Elasticsearch



Установка и использование



Установка Docker Portainer

## Оцените статью

(10 оценок, среднее: 4,60 из 5)



Статья распространяется под лицензией Creative Commons ShareAlike 4.0 при копировании материала ссылка на источник обязательна .

[Инструкции](#)

## Об авторе



ADMIN

Основатель и администратор сайта losst.ru, увлекаюсь открытым программным обеспечением и операционной системой Linux. В качестве основной ОС сейчас использую Ubuntu. Кроме Linux, интересуюсь всем, что связано с информационными технологиями и современной наукой.

## 17 комментариев к “Использование Docker для чайников”



Bogdan

8 июля, 2020 в 8:27 пп

Заметил только один странный момент ( Изначально в mariadb, image был mariadb, а потом mysql )

Может я чего то не понимаю или упустил момент

Очень крутая статья, благодаря некоторым ошибкам ( со своей стороны ) очень много разобрал.

[Ответить](#)



DeGun

12 июля, 2020 в 2:01 пп

Поясните пожалуйста: в FreeNAS используется jail - это тот же докер по сути?

[Ответить](#)



Alexander

21 октября, 2020 в 4:25 пп

Никак не могу разобраться с таким моментом:

нужно поменять имя хоста, т.е. вместо localhost прописать какое-нибудь другое.

Пробовал писать нужный хост в настройках pgsql в поле server\_name - не помогает

[Privacy](#)

[Ответить](#)**Slava**11 февраля, 2021 в 8:10 pp

Кто-то ещё получает при запуске в 8 разделе такие логи контейнера?

[ERROR] [MY-012224] [InnoDB] Tablespace flags are invalid in datafile:

./ibdata1, Space ID:0, Flags: 21. Please refer to

<http://dev.mysql.com/doc/refman/8.0/en/innodb-troubleshooting-dict.html>

for how to resolve the issue.

docker-mariadb\_1 | [ERROR] [MY-012237] [InnoDB] Corrupted page [page id: space=0, page number=0] of datafile './ibdata1' could not be found in the doublewrite buffer.

docker-mariadb\_1 | [ERROR] [MY-012930] [InnoDB] Plugin initialization aborted with error Data structure corruption.

docker-mariadb\_1 | [ERROR] [MY-011013] [Server] Failed to initialize DD Storage Engine.

docker-mariadb\_1 | [ERROR] [MY-010020] [Server] Data Dictionary initialization failed.

docker-mariadb\_1 | [ERROR] [MY-010119] [Server] Aborting

Думаю из-за них и не получается залогиниться в БД на localhost:8095 на шаге

[Ответить](#)**Anon**13 апреля, 2021 в 12:52 pp

И могу поздравить -- ни\*я не работает. Очередной крап в сети, скопипашенный с каких то е\*ней

[Ответить](#)**Pol**16 августа, 2021 в 1:31 дп[Privacy](#)

Привет! Подскажи такой момент плиз:

есть контейнер с приложением Django, который использует SQLite. в ручном режиме контейнер собирается, предпоследним шагом прогоняются миграции - все гуд. Но билд контейнера происходит из docker-compose.yml. На одном уровне с ним лежит папка database - в которую я хочу положить БД чтобы не хранить ее в контейнере, для чего в docker-compose.yml перед билдом создаю volumes, который монтируется в папку контейнера - в которой происходят миграции. Насколько я понял логику как ни крути volumes монтируется при docker-compose up и соответственно он банально перемонтирует папку системы и я своего файла с БД уже не вижу. Вопрос в том - КАК мне сделать миграцию, чтобы build положил БД уже в примонтированную папку? Я понимаю, что можно использовать именованные хранилища - но они же создаются где то в глубине системы, типа /var/lib/docker и тд, а хотелось бы в папку database рядом с docker-compose.yml

[Ответить](#)



Андрей

[18 августа, 2021 в 8:11 дп](#)

Перечитайте раздел «Монтирование папок» в статье, там ответ на ваш вопрос.

[Ответить](#)



witek

[24 августа, 2021 в 8:44 дп](#)

У кого не появляется страничка php info после подключения php fpm, нужно в файле site.conf в разделе location закомментировать строку try\_files и вместо этого вставить две строчки:

```
location / {  
root /var/www/html;  
try_files $uri /index.php$is_args$args;
```

[Ответить](#)

Privacy



Виталий

25 августа, 2021 в 12:00 pp

День добрый, не совсем понятен 7 пункт, а именно "хранилища" (losst-vl). Что это и откуда берется? Это типа облако отдельное или как?

[Ответить](#)

admin

25 августа, 2021 в 9:46 pp

Да, в своём роде это облако для контейнера. Добавляется в docker-compose в секции volumes. Это не какой-то внешний сервис, туда нужно прописать просто любое имя, и хранилище с таким именем будет создано, а потом его можно монтировать к контейнерами. А нужно это для того чтобы была возможность хранить файлы, к которым нужно иметь доступ из контейнера(например, файлы базы данных mysql), так как всё что есть в самом контейнере стирается при его пересборке.

[Ответить](#)

Виталий

26 августа, 2021 в 6:45 дп

т.е. это просто для удобства, чтобы не прописывать каждый раз путь хранения файла БД с основной машины?

[Ответить](#)

admin

26 августа, 2021 в 9:31 дп

Почти так. Хранить файлы БД в какой-нибудь папке на основной машине не надежно, можно забыть, случайно удалить. А так для таких целей специально есть хранилища, которыми управляет docker.

[Privacy](#)

[Ответить](#)**Виталий**26 августа, 2021 в 10:55 дп

Понял. Спасибо!

[Ответить](#)**Иван**15 октября, 2021 в 2:13 дп

Странно, но по адресу <http://localhost:8094> пишет: "Попытка соединения не удалась". Перепробовал по-всякому. Вообще никак не работает. Докер запущен, но в браузере ничего не выдает. Может Windows 8.1 докер не совместим?

[Ответить](#)**Иван**15 октября, 2021 в 2:16 дп

В файле hosts пробовал по-разному прописать. Тоже результатов нет

[Ответить](#)**Игорь**2 февраля, 2023 в 7:36 дп

Есть докер с десятком контейнеров. Есть необходимость организовать выход каждого контейнера через свой индивидуальный прокси. Как это можно организовать. Если можно поподробнее. Можно ли в конф файле прописать прокс конкретно на контейнер, так же как мы прописываем на весь докер?

[Privacy](#)

[Ответить](#)**Проходимец**7 июля, 2023 в 6:48 pp

В конце файла `losst-docker.yaml` нужно добавить  
`networks:`  
`losst-network:`

[Ответить](#)

## Оставьте комментарий

 Имя \* Email

Я прочитал и принимаю политику конфиденциальности. Подробнее [Политика конфиденциальности](#) \*

[Privacy](#)

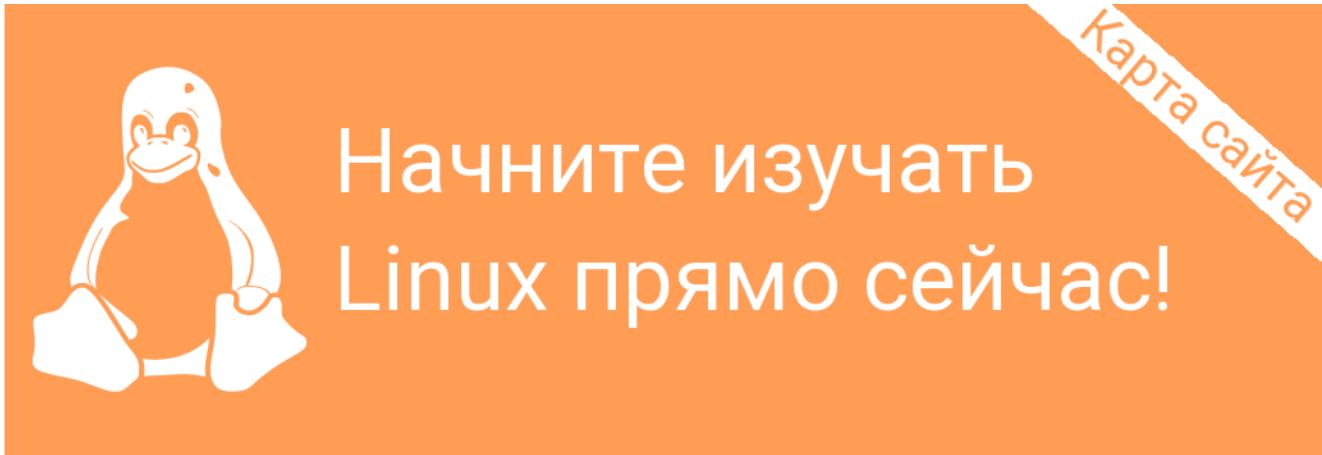
Русский

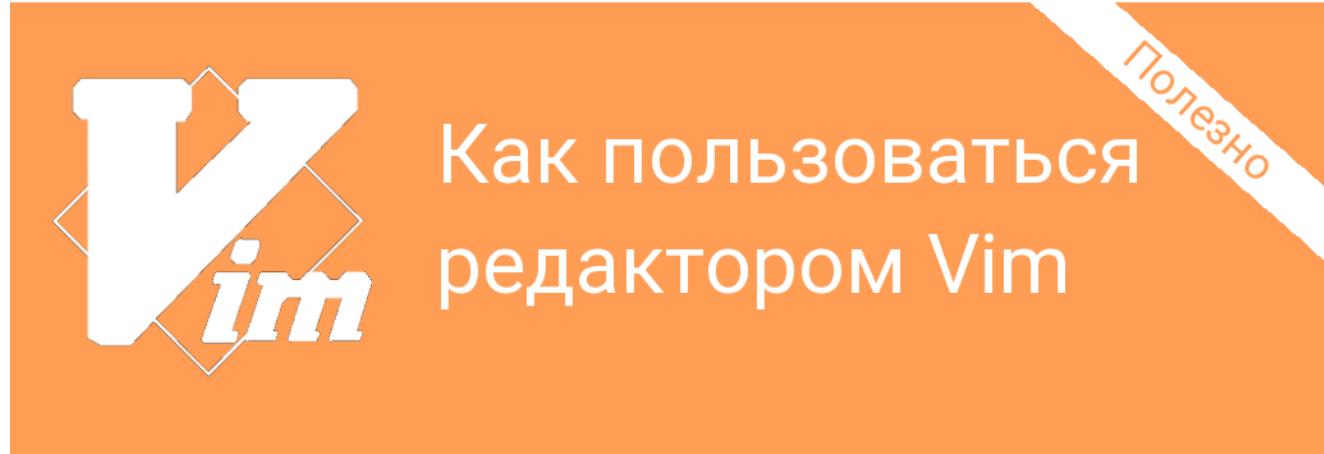
Поиск

## ПОИСК ПО КОМАНДАМ

Начните вводить команду

Поиск



[Лучшие](#)[Свежие](#)[Теги](#)[Команда chmod в Linux](#)

2020-04-13

[Команда find в Linux](#)

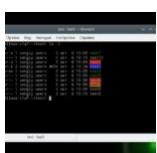
2021-10-17

[Как узнать IP-адрес в Linux](#)

2023-04-14

[Настройка Cron](#)

2021-10-01

[Права доступа к файлам в Linux](#)

2020-10-09

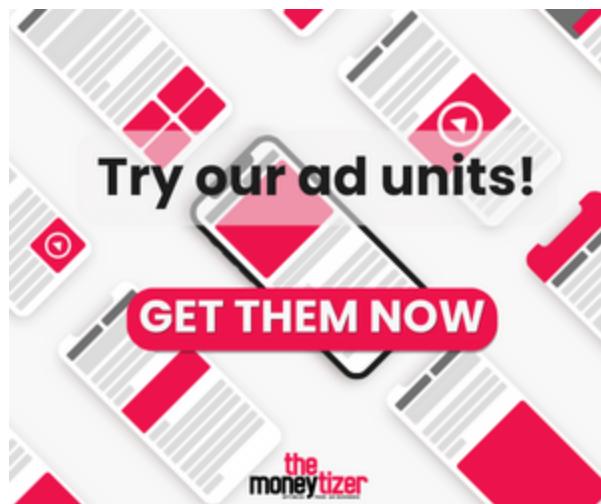
[Privacy](#)

# РАССЫЛКА

Ваш E-Mail адрес

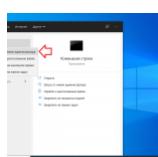
Я прочитал(а) и принимаю политику конфиденциальности

Sign up



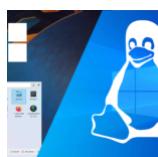
Windows

Списки



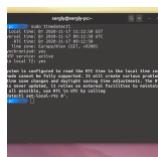
Восстановление Grub после установки Windows 10

2020-08-15



Установка Linux рядом с Windows 10 или 11

2023-02-08

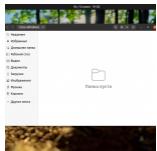


Сбивается время в Ubuntu и Windows

2023-02-18

Ошибка Ubuntu не видит сеть Windows

Privacy



2023-02-18

[Смотреть ещё](#)

## META

[Регистрация](#)[Войти](#)[Лента записей](#)[Лента комментариев](#)

## СЛЕДИТЕ ЗА НАМИ В СОЦИАЛЬНЫХ СЕТЯХ



## Интересное

[Шпаргалка по journalctl в Linux](#)

2019-03-22



Privacy



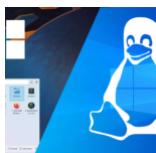
## Что такое Inode

2023-02-01



## Полезные утилиты для Linux

2021-10-12



## Установка Linux рядом с Windows 10 или 11

2023-02-08

©Losst 2024 CC-BY-SA [Политика конфиденциальности](#)