

[ГЛАВНАЯ](#) » [МАНУАЛ](#)

## Где хранятся образы, контейнеры и тома Docker в хост-системе Linux?



АВТОР

cryptoparty



НА ЧТЕНИЕ

5 мин



ОПУБЛИКОВАНО

25.10.2021

Хотите узнать, где находятся образы, контейнеры и тома Docker?

В типичной среде Linux образы Docker и данные контейнеров можно найти в:

```
/var/lib/docker/
```

Если на вашем сервере не хватает места, вам обязательно следует заглянуть в этот каталог.

В основном, все связанные с Docker сущности находятся в `/var/lib/docker`.



Но давайте рассмотрим его более конкретно, на примере образа и контейнера Alpine.



Примечание: Пожалуйста, обратите внимание, что эта информация предназначена только для образовательных целей. Манипуляции с каталогами/файлами Docker хост-системы никогда не рекомендуются. Команды `docker` и `docker-compose` всегда должны быть предпочтительным методом. Доступ к физически расположенным каталогам/файлам Docker должен осуществляться только в крайнем случае в чрезвычайных ситуациях.

## Содержание ^

1. Расположение образов Docker
2. Определенные местоположения образов
3. Расположение томов Docker
4. `/var/lib/docker/volumes/`
5. Тома Docker
6. Монтирование на хосте
7. Заключение

## Расположение образов Docker

Всякий раз, когда вы используете команду `docker pull` или запускаете `docker-compose up -d` для подготовки запуска приложений, именно здесь хранятся образы на сервере Ubuntu:

```
/var/lib/docker/overlay2
```

Здесь Overlay2 является драйвером хранилища Docker по умолчанию в Ubuntu.

Вы можете подтвердить это, выполнив команду `docker info` и поискав Storage Driver:





```
...  
    "GraphDriver": {  
        "Data": {  
            "MergedDir":  
"/var/lib/docker/overlay2/64c9c0cf8c9cfb0e2168071df0652a317d49f58a68fe86e4a  
9a9a525ab9e365e/merged",  
            "UpperDir":  
"/var/lib/docker/overlay2/64c9c0cf8c9cfb0e2168071df0652a317d49f58a68fe86e4a  
9a9a525ab9e365e/diff",  
            "WorkDir":  
"/var/lib/docker/overlay2/64c9c0cf8c9cfb0e2168071df0652a317d49f58a68fe86e4a  
9a9a525ab9e365e/work"  
        },  
    },  
...  

```

Вышеуказанные каталоги – это физическое расположение данных вашего контейнера внутри хост-системы.

Помните большую директорию с хэш-именем:


64c9c0cf8c9cfb0e2168071df0652a317d49f58a68fe86e4a9a525ab9e365e из раздела Docker Images?

Каталог под ним называется diff, как вы можете видеть в разделе LowerDir после :, который теперь является точкой монтирования контейнера – полученной из базового образа UpperDir!

Эти каталоги будут оставаться доступными даже после остановки контейнера.

Итак, полный путь, который является общим между образом (MergedDir, UpperDir и WorkDir) и контейнером (точка монтирования LowerDir), следующий:

```
/var/lib/docker/overlay2/64c9c0cf8c9cfb0e2168071df0652a317d49f58a68fe86e4a9  
a9a525ab9e365e/
```

После присвоения контейнеру до уровня LowerDir цель образа является смежной, поэтому четыре поля выделяются и основываются на другом каталоге (с новым хэшем)  из-за динамической природы последнего, который становится:

```
/var/lib/docker/overlay2/d734685e284c92bdc6063ac292a48813f30f4b0b2dd6fa2882279c569e506a3/
```



Примечание: Процесс монтирования каталога из base-image в контейнер очень похож на то, как вы монтируете тома в Docker!

## Расположение томов Docker

В отличие от образов и контейнеров Docker, физическое расположение томов довольно простое.

Тома располагаются по адресу:

```
/var/lib/docker/volumes/
```

Конкретные места расположения томов

В этом случае существует два типа томов.

Первый – обычные тома Docker, а второй – bind mounts.

## Тома Docker

Если вы ищете местоположение определенных томов, вы можете сначала использовать команду `docker volume ls` и проверить имя или ID тома.

Например, я запустил контейнер `alpine` с помощью следующей команды с томом:

```
$ docker run -ti -d --name alpine-container -v test-data:/var/lib/app/content alpine
```

Теперь автоматически будет создан том с именем `test-data`.

Теперь давайте создадим файл `test.md` в этом месте:



```
~$ docker exec alpine-container sh -c "touch /var/lib/app/content/test.md"
```

Убедитесь, что файл действительно был создан:

```
$ docker exec -ti alpine-container sh
/ # ls /var/lib/app/content/
test.md
/ # exit
```

Когда вы запустите `docker volume ls`, в списке появится том с именем `test-data`:

```
$ docker volume ls
DRIVER      VOLUME NAME
local       d502589845f7ae7775474bc01d8295d9492a6c26db2ee2c941c27f3cac4449d1
local       e71ee3960cfef0a133d323d146a1382f3e25856480a727c037b5c81b5022cb1b
local       test-data
```

Наконец, вы можете подтвердить фактическое расположение файла на вашей хост-системе:

```
$ sudo ls -l /var/lib/docker/volumes/test-data/_data
total 0
-rw-r--r-- 1 root root 0 Oct  6 23:20 test.md
```

Поэтому путь для смонтированного тома всегда находится в каталоге с именем `_data` внутри соответствующего каталога тома.

Вы также можете проверить такие пути используя команду `docker volume inspect` с последующим указанием имени или ID тома и просмотрев параметр `Mountpoint` в выводе:



```
$ docker volume inspect test-data
[
  {
    "CreatedAt": "2021-10-06T23:20:20+05:30",
    "Driver": "local",
    "Labels": null,
    "Mountpoint": "/var/lib/docker/volumes/test-data/_data",
    "Name": "test-data",
    "Options": null,
    "Scope": "local"
  }
]
```

## Монтирование на хосте

Расположение привязанных монтирований довольно очевидно и даже более просто, поскольку вы монтируете том непосредственно из расположения на стороне хоста:

```
$ mkdir /home/avimanyu/test-data
$ docker run -ti -d --name alpine-container -v /home/avimanyu/test-data:/var/lib/app/content alpine
```

В этом случае смонтированный том с именем test-data станет доступен на стороне контейнера как /var/lib/app/content.

## Заключение

В этом кратком руководстве я использовал общий подход на базе Linux, чтобы показать физическое расположение образов Docker, контейнеров и томов, расположенных на вашем Linux-сервере (в данном случае Ubuntu 20.04) на уровне хоста.

[#Alpine](#) [#docker](#) [#linux](#)

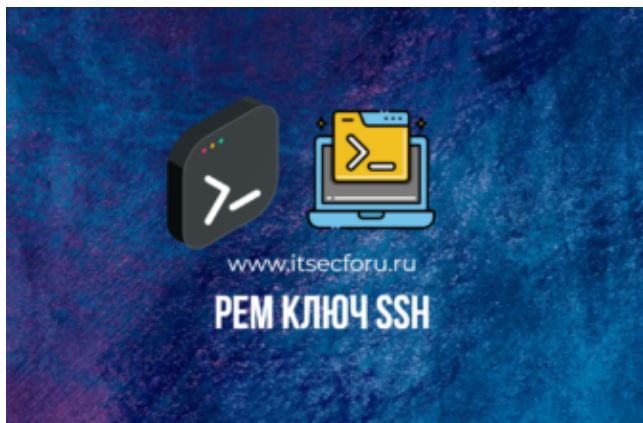
Пожалуйста, не спамьте и никого не оскорбляйте. Это поле для комментариев, а не спамбокс. Рекламные ссылки не индексируются!



## Добавить комментарий

Отправить комментарий

## Вам также может понравиться



### Как сгенерировать ключи SSH в формате PEM

Ключи SSH (Secure Shell) – это пара криптографических

 0



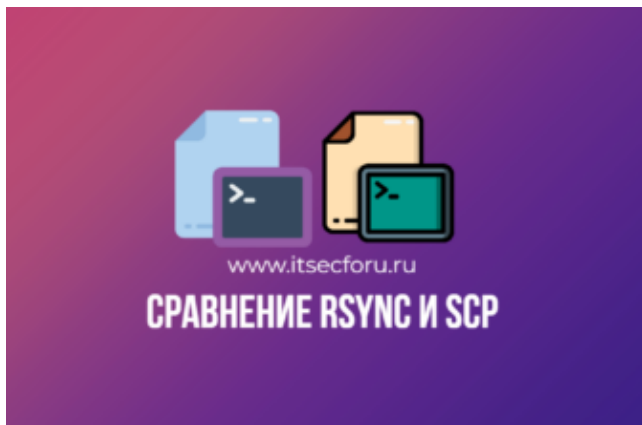
### Как предотвратить случайное удаление записей Crontab на Linux

Случалось ли вам переживать тот душераздирающий момент

 0







### 🐧 SCP vs RSYNC – что из них использовать в продакшене?

В этом руководстве рассматриваются основы и различия

🗨 0



### 🔑 Как проверить конфигурацию клиента SSH

Проверим конфигурацию SSH-клиента (разрешения и типы

🗨 0



### Как правильно откладывать денежные средства: советы и стратегии

Откладывание денежных средств является важной частью

🗨 0



### Тонкости решения финансовых вопросов при использовании Apple account Developer

Девайсы «яблочного» бренда продолжают удерживать первые

🗨 0



## Интеллектуальная обработка документов: автоматизация документооборота и повышение эффективности бизнеса

В современном мире, где информация является  
ключом

 0

## Настройка прокси-сервера APT на Debian Linux

Настройка прокси-сервера APT в среде Debian  
Linux может

 0

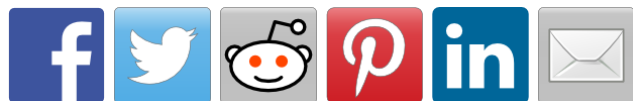
### Свежие комментарии

Сергей к записи [✓ Как установить самую последнюю версию OpenSSL на Windows 10](#)  
сгугторарту к записи [XERXES – самый мощный инструмент для DoS-атаки с использованием Kali Linux](#)  
Максим к записи [XERXES – самый мощный инструмент для DoS-атаки с использованием Kali Linux](#)  
сгугторарту к записи [🌐 Как заблокировать .git в Apache, Nginx и Cloudflare?](#)  
Павел к записи [🌐 Как заблокировать .git в Apache, Nginx и Cloudflare?](#)

### Свежие записи

[🌱 Kali Linux 2024 уже доступна. Скачайте ее прямо сейчас.](#)  
[🔑 Как сгенерировать ключи SSH в формате PEM](#)  
[🐧 Как предотвратить случайное удаление записей Crontab на Linux](#)  
[⚓ GitFive Инструмент OSINT](#)  
[🖨 Интерфейс Loopback в протоколах маршрутизации](#)

### Поделиться



© 2024 \$ information Security Squad

