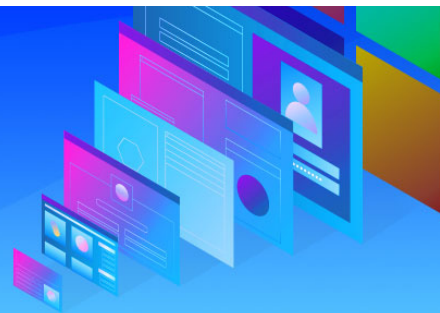


[КАК СТАТЬ АВТОРОМ](#)[Где работать в следующем году](#)

Лучшее предложение на рынке
VPS хостинг с Windows от **523** ₽/месяц

Лицензия на ОС включена в стоимость

**1923.28**

Рейтинг

RUVDS.comVDS/VPS-хостинг. Скидка 15% по коду **HABR15****ru_vds**

21 фев 2019 в 16:00

Изучаем Docker, часть 5: команды

9 мин 170K

Блог компании RUVDS.com, Разработка веб-сайтов*, Виртуализация*

[Тutorial](#)[Перевод](#)

Автор оригинала: Jeff Hale

Сегодняшняя часть цикла материалов по Docker, перевод которого мы публикуем, посвящена командам Docker. Документация Docker содержит подробнейшее описание великого множества команд, но тот, кто только начинает работу с этой платформой, может в них и потеряться, поэтому здесь приведены почти два десятка самых важных команд для работы с Docker. Продолжая сложившуюся традицию, мы сравним команды с россыпью ягод.

- [Часть 1: основы](#)
- [Часть 2: термины и концепции](#)
- [Часть 3: файлы Dockerfile](#)
- [Часть 4: уменьшение размеров образов и ускорение их сборки](#)
- [Часть 5: команды](#)
- [Часть 6: работа с данными](#)



Обзор

Давайте вспомним о том, что образы Docker создают на основе файлов `Dockerfile`, описывающих всё то, что нужно для сборки образов. Кроме того, не будем забывать и о том, что контейнер – это образ Docker, вызванный к жизни. Для того чтобы эффективно пользоваться командами Docker, в первую очередь нужно выяснить – с чем вы имеете дело – с образом или с контейнером. Если подумать об образах и контейнерах, то можно понять, что образ Docker может либо существовать, либо не существовать. То же самое можно сказать и о контейнерах Docker. Существующий контейнер Docker, кроме того, может пребывать либо в работающем, либо в неработающем состоянии.

После того, как вы выяснили, с чем именно вам нужно работать, вы можете найти подходящую команду.

Общие сведения о командах Docker

Вот кое-что, о чём полезно знать тем, кто хочет работать с Docker:

- Команды интерфейса командной строки Docker, используемые для управления чем-либо, начинаются с ключевого слова `docker`, за которым идёт пробел, затем идёт указание на то, на что именно будет направлена некая команда, потом ещё один

пробел, а потом следует сама команда. Например, именно так построена такая команда: `docker container stop` .

- Если команда направлена на конкретный образ или контейнер, то в ней используется имя или идентификатор такого образа или контейнера.

Например, команда `docker container run my_app` – это команда для создания и запуска контейнера с именем `my_app` . В примерах, которые будут приведены ниже, контейнеры мы будем называть `my_container` , образы – `my_image` , теги – `my_tag` , и так далее.

Сначала мы будем рассматривать саму команду, потом – флаги, которые можно с ней использовать, если такие флаги существуют. Если перед флагом стоит два тире – то это его полная форма, флаг с одним тире – это сокращённый вариант некоего флага. Действуют они одинаково. Например, `-p` – это сокращённая форма флага `--port` .

Цель этого материала заключается в том, чтобы дать вам общие сведения о командах Docker. Так вы, имея общее представление о них и зная о возможностях платформы, доступных благодаря этим командам, сможете, при необходимости, найти подробные сведения о них. Команды, о которых пойдёт речь, испытаны на ОС семейства Linux с использованием движка Docker версии 18.09.1 и API версии 1.39.

Примечание о командах, поддерживаемых Docker CLI 1.13

В интерфейсе командной строки Docker версии 1.13 представлены обновлённые, логически сгруппированные команды. При этом старые команды всё ещё работают, но новыми пользоваться легче, особенно – начинающим. Речь идёт, например, о том, что в версии 1.12 использовалась команда вида `docker create` , а в версии 1.13 стала доступна команда `docker container create` . Сведения о соответствии старых и новых команд можно найти [здесь](#) .

Сначала мы посмотрим на команды, предназначенные для управления контейнерами, затем обсудим управление образами.

Команды для управления контейнерами

Общая схема команд для управления контейнерами выглядит так:

```
docker container my_command
```

Вот команды, которые могут быть подставлены туда, где мы использовали `my_command` :

- `create` – создание контейнера из образа.
- `start` – запуск существующего контейнера.
- `run` – создание контейнера и его запуск.
- `ls` – вывод списка работающих контейнеров.
- `inspect` – вывод подробной информации о контейнере.
- `logs` – вывод логов.
- `stop` – остановка работающего контейнера с отправкой главному процессу контейнера сигнала `SIGTERM` , и, через некоторое время, `SIGKILL` .
- `kill` – остановка работающего контейнера с отправкой главному процессу контейнера сигнала `SIGKILL` .
- `rm` – удаление остановленного контейнера.

Команды для управления образами

Для управления образами используются команды, которые выглядят так:

```
docker image my_command
```

Вот некоторые из команд этой группы:

- `build` – сборка образа.
- `push` – отправка образа в удалённый реестр.
- `ls` – вывод списка образов.
- `history` – вывод сведений о слоях образа.
- `inspect` – вывод подробной информации об образе, в том числе – сведений о слоях.
- `rm` – удаление образа.

Разные команды

- `docker version` – вывод сведений о версиях клиента и сервера Docker.
- `docker login` – вход в реестр Docker.
- `docker system prune` – удаление неиспользуемых контейнеров, сетей и образов, которым не назначено имя и тег.

Теперь рассмотрим эти команды подробнее.

Контейнеры

Начало существования контейнера

На начальном этапе работы с контейнерами используются команды `create`, `start` и `run`. Они применяются, соответственно, для создания контейнера, для его запуска, и для его создания и запуска.

Вот команда для создания контейнера из образа:

```
docker container create my_repo/my_image:my_tag
```

В следующих примерах конструкция `my_repo/my_image:my_tag` будет сокращена до `my_image`.

Команда `create` принимает множество флагов. Например, её можно записать в таком виде:

```
docker container create -a STDIN my_image
```

Флаг `-a` представляет собой краткую форму флага `--attach`. Этот флаг позволяет подключить контейнер к `STDIN`, `STDOUT` или `STDERR`.

После того, как контейнер создан, его можно запустить следующей командой:

```
docker container start my_container
```

Обратите внимание на то, что сослаться на контейнер в команде можно либо используя его `ID`, либо имя.

Теперь взглянем на команду, которая позволяет создать и запустить контейнер:

```
docker container run my_image
```

Эта команда тоже способна принимать множество аргументов командной строки. Рассмотрим некоторые из них на примере такой конструкции:

```
docker container run -i -t -p 1000:8000 --rm my_image
```

Флаг `-i` — это сокращение для `--interactive`. Благодаря этому флагу поток `STDIN` поддерживается в открытом состоянии даже если контейнер к `STDIN` не подключён.

Флаг `-t` — это сокращение для `--tty`. Благодаря этому флагу выделяется псевдотерминал, который соединяет используемый терминал с потоками `STDIN` и `STDOUT` контейнера.

Для того чтобы получить возможность взаимодействия с контейнером через терминал нужно совместно использовать флаги `-i` и `-t`.

Флаг `-p` представляет собой сокращение для `--port`. Порт — это интерфейс, благодаря которому контейнер взаимодействует с внешним миром. Конструкция `1000:8000` перенаправляет порт Docker 8000 на порт 1000 компьютера, на котором выполняется контейнер. Если в контейнере работает некое приложение, способное выводить что-то в браузер, то, для того, чтобы к нему обратиться, в нашем случае можно перейти в браузере по адресу `localhost:1000`.

Флаг `--rm` автоматически удаляет контейнер после того, как его выполнение завершится.

Рассмотрим ещё некоторые примеры команды `run`:

```
docker container run -it my_image my_command
```

В подобной конструкции может применяться команда `sh`, которая создаст сессию терминала в контейнере, с которой можно взаимодействовать через ваш терминал. При работе с образами, основанными на Alpine, лучше ориентироваться на использование `sh`

а не `bash`, так как в этих образах, по умолчанию, оболочка `bash` не установлена. Для выхода из интерактивной сессии воспользуйтесь командой `exit`.

Обратите внимание на то, что здесь мы скомбинировали флаги `-i` и `-t` в `-it`.

Вот ещё один пример работы с командой `run`:

```
docker container run -d my_image
```

Флаг `-d` – это сокращение для `--detach`. Эта команда запускает контейнер в фоновом режиме. Это позволяет использовать терминал, из которого запущен контейнер, для выполнения других команд во время работы контейнера.

Проверка состояния контейнера

Если у вас имеются запущенные контейнеры Docker и вы хотите узнать о том, что это за контейнеры, вам понадобится вывести их список. Сделать это можно такой командой:

```
docker container ls
```

Эта команда выводит список выполняющихся контейнеров и снабжает этот список некоторыми полезными сведениями о них. Вот ещё один пример этой команды:

```
docker container ls -a -s
```

Ключ `-a` этой команды – это сокращение для `--all`. Благодаря использованию этого ключа можно вывести сведения обо всех контейнерах, а не только о выполняющихся.

Ключ `-s` – это сокращение для `--size`. Он позволяет вывести размеры контейнеров.

Вот команда, которая выводит подробные сведения о контейнере:

```
docker container inspect my_container
```

Вот команда, выводящая логи контейнера:

```
docker container logs my_container
```

| Завершение работы контейнера

Иногда работающий контейнер надо остановить. Для этого используется такая команда:

```
docker container stop my_container
```

Она позволяет останавливать работающие контейнеры, позволяя им корректно завершить работу. У контейнера есть, по умолчанию, 10 секунд, на то, чтобы завершить работу.

Если же контейнер нужно остановить быстро, не заботясь о корректном завершении его работы, можно воспользоваться такой командой:

```
docker container kill my_container
```

Команда `kill`, если сравнить работающий контейнер с включенным телевизором, напоминает выключение телевизора путём отключения его от электричества. Поэтому, в большинстве ситуаций, для остановки контейнеров рекомендуется использовать команду `stop`.

Вот команда, которая позволяет быстро остановить все работающие контейнеры:

```
docker container kill $(docker ps -q)
```

Для удаления остановленного контейнера можно воспользоваться такой командой:

```
docker container rm my_container
```

Вот команда, которая позволяет удалить все контейнеры, которые на момент вызова этой команды не выполняются:

```
docker container rm $(docker ps -a -q)
```


Подведём итоги этого раздела. Сначала контейнер создают, потом его запускают, или комбинируют эти два шага, используя команду вида `docker run my_container`. После этого запускается контейнеризированное приложение.

Потом контейнер останавливают командой `docker stop my_container`. Для удаления контейнера используется команда `docker rm my_container`.

Поговорим теперь о командах, используемых для работы с образами, с теми самими шаблонами, из которых создают контейнеры.

Образы

| Создание образов

Вот команда, которая позволяет собирать образы Docker:

```
docker image build -t my_repo/my_image:my_tag .
```

В данном случае создаётся образ с именем `my_image`, при его сборке используется файл `Dockerfile`, находящийся по указанному пути или URL.

Флаг `-t` — это сокращение для `--tag`. Он указывает Docker на то, что создаваемому образу надо назначить предоставленный в команде тег. В данном случае это `my_tag`.

Точка в конце команды указывает на то, что образ надо собрать с использованием файла `Dockerfile`, находящегося в текущей рабочей директории.

После того, как образ собран, его можно отправить в удалённый реестр. Благодаря этому им смогут воспользоваться другие люди, его можно будет загрузить и запустить на другом компьютере. Предположим, вы хотите использовать `Docker Hub`. Если так — вам понадобится завести там учётную запись. Пользоваться этим ресурсом можно бесплатно.

После того, как вы зарегистрируетесь на `Docker Hub`, вам нужно войти в систему. И хотя команда, которая для этого используется, напрямую к командам, предназначенным для работы с образами, не относится, её полезно будет рассмотреть именно здесь. Речь идёт о следующей команде:

```
docker login
```

Она позволяет войти в учётную запись на Docker Hub. Для входа в систему вам понадобится ввести имя пользователя и пароль.

После входа в систему можно будет отправлять образы в реестр. Делается это так:

```
docker image push my_repo/my_image:my_tag
```

Теперь, когда у вас наберётся несколько образов, вы можете их исследовать с помощью специальных команд.

| Исследование образов

Вот команда, которая выводит список образов, выводя, в том числе, и сведения об их размере:

```
docker image ls
```

Следующая команда позволяет вывести сведения о промежуточных образах, входящих в состав образа, в частности – данные об их размерах и о том, как они были созданы:

```
docker image history my_image
```

Вот команда, которая выводит подробные сведения об образе, в том числе – данные о слоях, из которых состоит образ:

```
docker image inspect my_image
```

Если вы создадите очень много образов, может случиться так, что некоторые из них понадобится удалить.

| Удаление образов

Вот команда, которая позволяет удалить указанный образ:

```
docker image rm my_image
```

Если образ хранится в удалённом репозитории, он оттуда удалён не будет.

Вот команда, которая позволяет удалить все локальные образы:

```
docker image rm $(docker images -a -q)
```

Пользоваться этой командой стоит с осторожностью, но надо заметить, что при её использовании образы, хранящиеся в удалённом репозитории, удалены не будут. В этом заключается одно из преимуществ хранения образов в репозиториях.

Мы рассмотрели основные команды, используемые для управления контейнерами и образами. Поговорим теперь ещё о некоторых командах.

Разные команды

Вот команда, которая выводит сведения о версиях клиента и сервера Docker:

```
docker version
```

Эта, уже известная вам команда, применяется для входа в реестр Docker:

```
docker login
```

Такая команда позволяет удалить неиспользуемые контейнеры, сети и образы, которым не назначено имя и тег:

```
docker system prune
```

Вот пример её использования:

```
docker system prune -a --volumes
```

Ключ `-a` – сокращение для `--all`, позволяет удалить неиспользуемые образы, а не только те, которым не назначено имя и тег.

Ключ `--volumes` позволяет удалить неиспользуемые тома.

Итоги

В этом материале мы рассмотрели полезные команды Docker. Если вы только начинаете работать с Docker, то вам стоит обратить внимание на три следующих важнейших команды:

Создание и запуск контейнера:

```
docker container run my_image
```

Сборка образа:

```
docker image build -t my_repo/my_image:my_tag .
```

Отправка образа в удалённый репозиторий:

```
docker image push my_repo/my_image:my_tag
```

Для того чтобы посмотреть справку по командам Docker, можете выполнить в терминале команду `docker`. Здесь можно найти справочные материалы по интерфейсу командной строки Docker.

В следующий раз мы поговорим о работе с данными в Docker.

Уважаемые читатели! Если вы работаете с Docker, то у вас, наверняка, есть собственный список часто используемых команд. Если это так – просим вас этим списком поделиться.

Habrahabr10

Промо-код для скидки в 10% на наши виртуальные сервера


Теги: Docker, разработка

Хабы: Блог компании RUVDS.com, Разработка веб-сайтов, Виртуализация

Редакторский дайджест

Присылаем лучшие статьи раз в месяц


Электронпочта



RUVDS.com

VDS/VPS-хостинг. Скидка 15% по коду **HABR15**

Telegram ВКонтакте Twitter



409

349.3

Карма

Рейтинг

@ru_vds
Пользователь


Комментарии 6


Публикации


ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ


22 часа назад


Герои напильника и паяльника: итоги сезона DIY


 8 мин


 8.5K


 Сезон DIY

 Спецпроект

 +44


 42


 6


 **spiritus_sancti**


23 часа назад

RGB-усилители. Особенности, проблемы, выбор

 Простой

 6 мин

 5.6K

 Тutorial

 +40 24 11

ru_vds

19 часов назад

Профилирование Python — почему и где тормозит ваш код



Средний



10 мин



3.5K

Тutorial

Перевод

 +32 93 4

ViktorSergeev

16 часов назад

WebOne: даём жизнь старым браузерам



6 мин



3.3K

 +30 29 11

zatiм

1 час назад

Видеокарта VGA для микроконтроллера



Средний



17 мин



893

Тutorial

 +21 16 6

it_union

2 часа назад

ЗАО Гейм Инсайт Труп



Простой



4 мин



3.4K

 +21 7 7

AnnaVIMorozova

19 часов назад

Как повысить эффективность коммуникаций в команде: учимся решать конфликты



7 мин



2.3K

+19

43

2

Yu-Leo
вчера в 15:22

Обзор электронной книги Meebook P10 Pro

Простой 9 мин 6.3K

Обзор

+18

18

8

Sagidullin
6 часов назад

Всего два месяца — и новый релиз ядра Linux. Что появилось в ядре 6.5, что изменилось и что удалили. Новые возможности

5 мин 5.3K

+15

6

5

mr-pickles
22 часа назад

Архетипы программных архитекторов. Часть 2

Простой 9 мин 2.5K

Перевод

+15

32

0

Показать еще

ИНФОРМАЦИЯ			
Ваш аккаунт	Разделы	Информация	Услуги
Войти	Статьи	Устройство сайта	Корпоративный блог
Регистрация	Новости	Для авторов	Медийная реклама
	Хабы	Для компаний	Нативные проекты
	Компании	Документы	Образовательные программы
	Авторы	Соглашение	



- Настройка языка
- Техническая поддержка
- Вернуться на старую версию

© 2006–2023, Habr

.....

VPS Windows от 523 рублей в месяц. Бесплатный тестовый период 3 дня.
ruvds.com


VDS в Цюрихе. Дата-центр TIER III – швейцарское качество по низкой цене.
ruvds.com

Антивирусная защита виртуального сервера. Легкий агент для VPS.
ruvds.com

VPS в Лондоне. Дата-центр TIER III – английская точность за рубли.
ruvds.com

VPS с видеокартой на мощных серверах 3,4ГГц
ruvds.com

ПРИЛОЖЕНИЯ

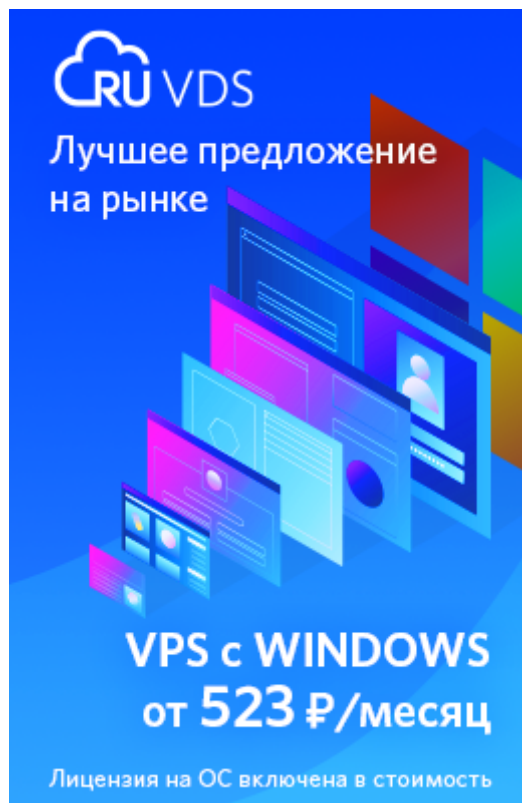


RUVDS Client

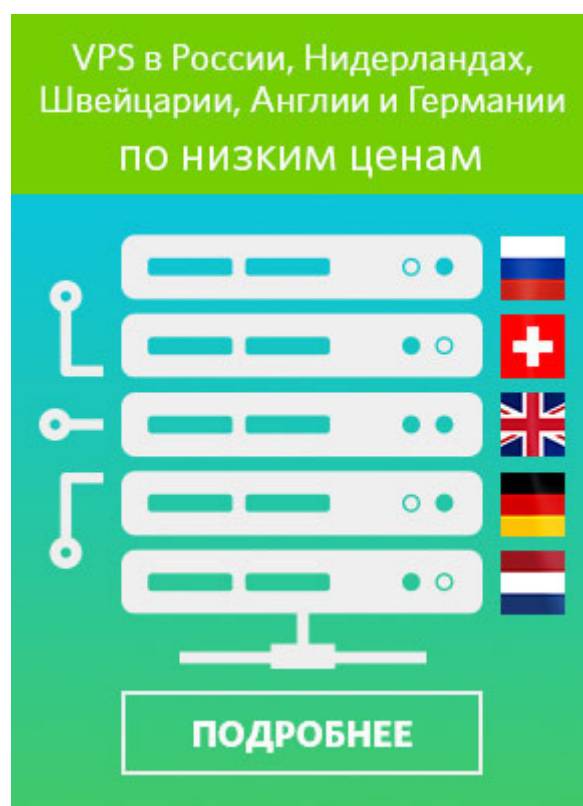
Приложение для мониторинга и управления виртуальными серверами RUVDS с мобильных устройств.

Android iOS

ВИДЖЕТ



ВИДЖЕТ



БЛОГ НА ХАБРЕ

19 часов назад

<https://habr.com/ru/companies/ruvds/articles/440660/>

Профилирование Python — почему и где тормозит ваш код

 3.5K  4

23 часа назад

RGB-усилители. Особенности, проблемы, выбор

 5.6K  11

27 авг в 17:00

Интернет 90-х: когда после 20 часов в онлайн тебе пишет президент ISP

 15K  40

26 авг в 17:00

История компьютерных стратегий. Часть 8. «Age of Empires»: шедевр геймдева, от которого бомбит у любителей истории

 13K  12

25 авг в 16:00

Xbox is a new Dreamcast. Зачем покупать консоль от Microsoft в 2023 году и во что играть

 6.5K  8