Как установить Docker на Ubuntu

**20 Февраля 202511 мин64**

Docker представляет собой открытую платформу, предназначенную для разработки, доставки и развертывания приложений в контейнерах. Контейнеризация позволяет объединять программное обеспечение вместе со всеми его зависимостями в единую автономную единицу, что гарантирует одинаковую работу приложения в любых средах — начиная  разработкой и заканчивая тестированием и эксплуатацией. Это усложненный процесс интеграции и развертывания становится более простым, заметно уменьшаются затраты на инфраструктуру, а масштабировать приложения становится удобнее.

Основные требования для установки:

* Процессор с 64-битным ядром и поддержкой виртуализации через KVM.
* Минимум 4 ГБ RAM.

Существуют разные способы установки Docker на Ubuntu:

* Настройка и инсталляция Docker Engine с использованием apt репозитория.
* Ручная установка с помощью загружаемого deb-файла.
* Установка с применением официального скрипта (рекомендуется исключительно для использования в тестовой среде и разработки).

Далее в статье мы будем использовать метод через репозиторий apt. Он считается оптимальным для работы с виртуальными  серверами.

[Настройка репозитория apt Docker](https://firstvds.ru/technology/kak-ustanovit-docker-na-ubuntu#nastroika-repozitoriya-apt-doc)

[Использование Docker-команд](https://firstvds.ru/technology/kak-ustanovit-docker-na-ubuntu#ispolzovanie-docker-komand)

[Работа с Docker-образами](https://firstvds.ru/technology/kak-ustanovit-docker-na-ubuntu#rabota-s-docker-obrazami)

[Запуск Docker-контейнеров](https://firstvds.ru/technology/kak-ustanovit-docker-na-ubuntu#zapusk-docker-konteinerov)

[Управление Docker-контейнерами](https://firstvds.ru/technology/kak-ustanovit-docker-na-ubuntu#upravlenie-docker-konteinerami)

[Создание собственного Docker-образа](https://firstvds.ru/technology/kak-ustanovit-docker-na-ubuntu#sozdanie-sobstvennogo-docker-o)

[Загрузка Docker-образа в Docker-репозиторий](https://firstvds.ru/technology/kak-ustanovit-docker-na-ubuntu#zagruzka-docker-obraza-v-docke)

Настройка репозитория apt Docker

Перед установкой Docker Engine на сервер необходимо настроить репозиторий apt Docker. После этого установка и обновление Docker станет простым и автоматизированным процессом.

**Шаги по настройке репозитория:**

1. Обновите список пакетов:  
   Убедитесь, что у вас есть актуальная информация о доступных пакетах.

sudo apt-get update

Copy

1. Установите необходимые утилиты для добавления репозиториев:

sudo apt-get install ca-certificates curl

Copy

1. Создайте каталог для ключей шифрования APT:

sudo install -m 0755 -d /etc/apt/keyrings

Copy

1. Скачайте официальный GPG-ключ Docker:

sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

Copy

1. Установите разрешение на ключ:

sudo chmod a+r /etc/apt/keyrings/docker.asc

Copy

1. Добавьте репозиторий Docker:
2. echo \
3. "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] https://download.docker.com/linux/ubuntu \
4. $(. /etc/os-release && echo "${UBUNTU\_CODENAME:-$VERSION\_CODENAME}") stable" | \

sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

Copy

1. Снова обновите список пакетов:

sudo apt-get update

Copy

1. И теперь поставьте нужные пакеты для Docker:

sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

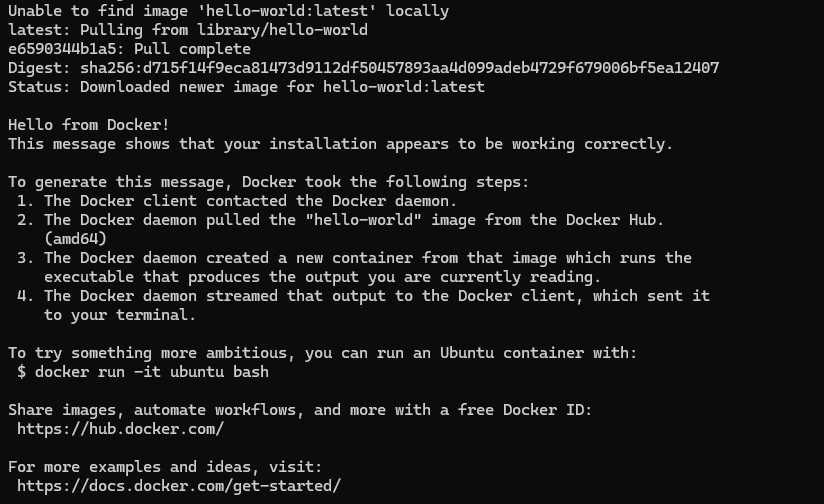
Copy

1. В конце проверьте, что всё работает, запустив тестовый контейнер. У нас это будет “hello-word”**,** где в консоль выведется информация и контейнер завершит свою работу:

sudo docker run hello-world

Copy

После запуска у вас должно отобразиться следующее в консоли:

[](https://firstvds.ru/sites/default/files/all_images/instructions/install_docker/image1.png)

Использование Docker-команд

Работа с Docker основывается на передаче ему набора опций и команд, после которых следуют необходимые аргументы. Общий формат команды выглядит следующим образом:

docker [опции] [команда] [аргументы]

Copy

Чтобы вывести полный список доступных команд, выполните:

docker

Copy

Ниже представлен перечень основных команд Docker:

**ps** – Показывает список контейнеров (по умолчанию только запущенные).

**pull**– Загружает образ из реестра (например, Docker Hub).

**build**– Собирает образ из Dockerfile.

**logs**– Выводит логи контейнера.

**run**– Создает и запускает контейнер из образа.

**stop**– Останавливает контейнер (посылает SIGTERM, затем SIGKILL).

**kill**– Мгновенно заканчивает контейнер (SIGKILL по умолчанию).

**rm**– Удаляет контейнер.

**rmi**– Удаляет образ.

**volume ls** – Показывает список доступных томов Docker.

Для просмотра дополнительных опций конкретной команды используйте:

docker <команда> --help

Copy

Чтобы получить общую системную информацию о Docker, выполните:

docker info

Copy

Работа с Docker-образами

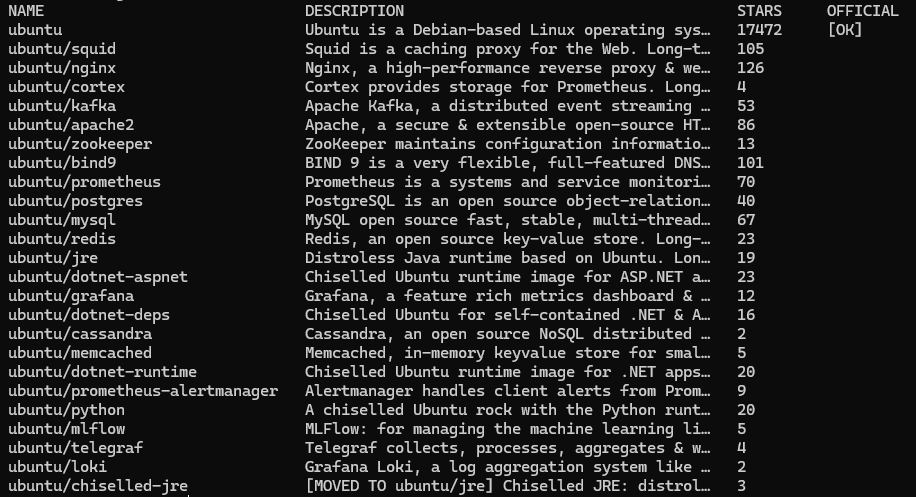
Контейнеры формируются на основе образов. По умолчанию они загружаются из [Docker Hub](https://hub.docker.com/) — публичного реестра, управляемого создателями проекта. Любой пользователь может разместить там собственные образы, поэтому большинство необходимых приложений и версий Linux вы найдете именно в этом месте.

Для поиска доступных образов на Docker Hub, воспользуйтесь командой **docker**вместе с подкомандой **search**. К примеру, если вам нужно найти образ Ubuntu, введите:

docker search ubuntu

Copy

Скрипт выполнит запрос к Docker Hub и выдаст перечень образов, названия которых совпадают с вашим поисковым запросом. Результат будет выглядеть приблизительно следующим образом:

[](https://firstvds.ru/sites/default/files/all_images/instructions/install_docker/image2.png)

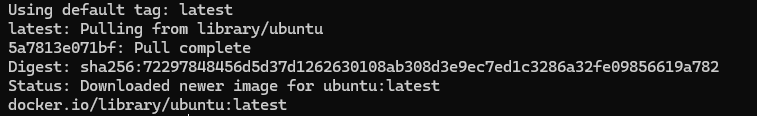
В колонке **OFFICIAL**отметки **OK**означают образы, созданные и поддерживаемые компанией-разработчиком проекта. Как только вы выберете подходящий вам образ, его можно будет загрузить на ваш сервер с помощью подкоманды **pull**.

Чтобы скачать официальный образ Ubuntu на вашем сервере, введите следующую команду:

docker pull ubuntu

Copy

После успешной загрузки вы увидите подтверждающее сообщение.

[](https://firstvds.ru/sites/default/files/all_images/instructions/install_docker/image3.png)

Вы можете использовать загруженный образ для запуска контейнера, применяя подкоманду **run**. Как показано выше на примере с hello-world, если в момент выполнения команды **run**нужный образ отсутствует на вашем сервере, Docker автоматически загрузит его, а затем запустит контейнер. Для просмотра списка образов, уже находящихся на вашем сервере, введите:

docker images

Copy

Результат будет выглядеть следующим образом:

[Просмотр списка образов](https://firstvds.ru/sites/default/files/all_images/instructions/install_docker/image4.png)

Запуск Docker-контейнеров

Контейнер hello-world, который вы запускали ранее, служит демонстрацией контейнера, выполняющего свои задачи, и завершает работу после вывода тестового сообщения.Однако контейнеры обладают гораздо большей функциональностью и интерактивностью. По сути, они напоминают виртуальные машины, но более экономичны в использовании ресурсов.

Например, давайте запустим контейнер на основе последней версии образа Ubuntu. Использование ключей -i (интерактивный режим) и -t (выделение терминала) позволит вам получить доступ к интерактивной командной оболочке внутри контейнера:

docker run -it ubuntu

Copy

* **-i (interactive)**: Позволяет поддерживать стандартный ввод (STDIN) контейнера открытым, что необходимо для взаимодействия с оболочкой.
* **-t (tty)**: Создаёт виртуальный терминал, обеспечивая удобный интерфейс для ввода команд и отображения вывода.

Комбинация этих ключей (-it) обеспечивает полноценный интерактивный сеанс внутри контейнера, позволяя вам выполнять команды так, как если бы вы работали в обычной терминальной сессии.

Ваш консоль должна измениться, показывая, что вы теперь находитесь внутри контейнера, и будет выглядеть так:

[ Убеждаемся, что находимся внутри контейнера](https://firstvds.ru/sites/default/files/all_images/instructions/install_docker/image5.png)

Обратите внимание на идентификатор контейнера в командной строке. В данном примере это **c4cfc0823039**. Этот идентификатор пригодится вам позже, если вы захотите найти и удалить контейнер. Теперь у вас есть возможность выполнять любые команды внутри контейнера. Например, вы можете обновить базу данных пакетов в контейнере. Дополнительно указывать sudo не требуется, так как вы работаете с правами пользователя root.

Чтобы выйти из контейнера, используйте команду для выхода:

exit

Copy

Управление Docker-контейнерами

С течением времени на вашем компьютере может скопиться большое количество работающих и остановленных контейнеров в Docker. Чтобы увидеть текущий список активных контейнеров, введите команду:

docker ps

Copy

Результат будет выглядеть примерно так:

[](https://firstvds.ru/sites/default/files/all_images/instructions/install_docker/image6.png)

В этом процессе мы запустили два контейнера: один демонстрационный — hello-world, и один из образа ubuntu. Оба контейнера завершили свою работу, но остались в системе. Чтобы увидеть все контейнеры, как активные, так и завершенные, используйте команду **docker ps** с параметром -a:

docker ps -a

Copy

Вывод будет следующим:

[Вывод всех контейнеров](https://firstvds.ru/sites/default/files/all_images/instructions/install_docker/image7.png)

Для повторного запуска остановленного контейнера используйте команду**docker start**, указав его ID или имя. Например, давайте запустим контейнер на основе Ubuntu с ID c4cfc0823039:

docker start 1c08a7a0d0e4

Copy

После выполнения **docker ps** мы увидим его в списке активных контейнеров:

[](https://firstvds.ru/sites/default/files/all_images/instructions/install_docker/image8.png)

Чтобы остановить работающий контейнер, используйте команду **docker stop**, указав его ID или имя. В данном случае мы воспользуемся именем контейнера, которое Docker назначил контейнеру, например, kind\_panini:

docker stop kind\_panini

Copy

Когда контейнер больше не нужен, вы можете удалить его командой **docker rm**, указав ID контейнера или его имя. Для примера удалим тестовый контейнер, который мы запускали в начале — Hello World, используйте **docker ps -a** для поиска его id или name, затем удалите контейнер:

docker rm <имя\_контейнра|id\_контейнар>

Copy

Вы можете запустить новый контейнер, присвоив ему имя при помощи параметра --name. Опция --rm позволяет создать контейнер, который будет автоматически удален при остановке. Для получения более детальной информации о возможностях и параметрах команд, используйте:

docker run help

Copy

Контейнеры также можно преобразовать в образы, которые затем могут служить для создания новых контейнеров. Давайте разберемся, как это сделать.

Создание собственного Docker-образа

При запуске Docker-образа вы можете создавать, изменять или удалять файлы, аналогично работе с виртуальной машиной. Однако все изменения, что вы вносите, будут применимы только к конкретному контейнеру. Вы можете запускать и останавливать контейнер, но как только удалите его с помощью команды **docker rm**, все внесенные изменения и данные будут утеряны навсегда.

В этой части мы рассмотрим, как сохранить текущее состояние контейнера, превратив его в новый Docker-образ.

Предположим, вы установили какое-либо ПО в контейнер из образа Ubuntu, и у вас есть работающий контейнер, который теперь отличается от исходного образа, с которого он был создан. И например, вы хотите сохранить изменённый образ, чтобы использовать его в будущем в качестве основы для новых образов.

Чтобы зафиксировать изменения и создать новый образ, используйте команду:

docker commit -m "Что вы сделали с образом" -a "Имя автора" container\_id repository/new\_image\_name

Copy

Параметр -m позволяет описать внесенные изменения, что поможет вам и другим пользователям понять, что было сделано. Опция -a указывает автора изменений. container\_id — это идентификатор контейнера, который вы записали ранее, работая с Docker. Если вы не создали дополнительные репозитории на Docker Hub, репозиторию будет автоматически присвоено имя пользователя.

Например, для пользователя FirstJohn с ID контейнера c4cfc0823039,  команда будет выглядеть так:

docker commit -m "добавил Python" -a "FirstJohn" c4cfc0823039 FirstJohn/ubuntu-python

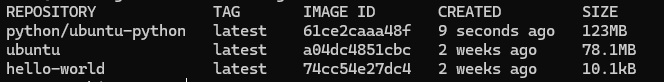
Copy

Здесь мы создаем новый образ контейнера, который называем ubuntu-python. При фиксации изменений создается этот новый образ, который затем сохраняется локально на сервере. В следующем разделе мы рассмотрим пример загрузки созданного образа на Docker-hub.

Если теперь снова вывести список Docker-образов, вы увидите новый образ и исходный, от которого он был получен:

docker images

Copy

[](https://firstvds.ru/sites/default/files/all_images/instructions/install_docker/image9.png)

В этом примере ubuntu-python — это новый образ, созданный на основе существующего образа ubuntu из Docker Hub. Разница в размере отражает изменения, такие как установка Python. Теперь, если вам потребуется контейнер с Ubuntu и предустановленным Python, вы можете просто воспользоваться новым образом.

Кроме того, образы также можно создавать с помощью файла Dockerfile, что позволяет автоматизировать установку программного обеспечения, но это тема для другой статьи.

Теперь давайте посмотрим, как мы можем поделиться новым образом, чтобы другие пользователи могли создавать контейнеры на его основе.

Загрузка Docker-образа в Docker-репозиторий

Чтобы загрузить образ на Docker Hub или иную платформу, вам потребуется учетная запись. После этого откроется возможность загружать образы в свой репозиторий или публичный, если у вас есть доступ к нему. Далее мы рассмотрим пример загрузки в собственный репозиторий на Docker Hub.

Начнем процесс загрузки образа с входа в ваш аккаунт на Docker Hub:

docker login -u ваш-логин-на-докер-хаб

Copy

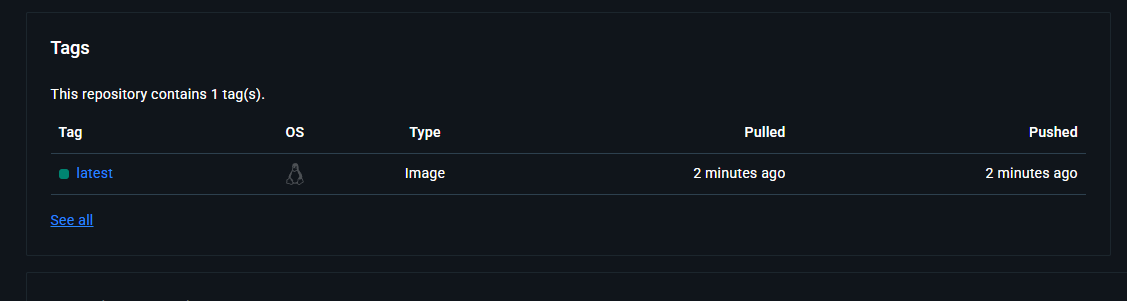
Система запросит у вас аутентификацию, требуя ввод пароля от Docker Hub. При правильном вводе данных, вы получите доступ.

Далее загрузите ваш образ командой:

docker push ваш-логин-на-докер-хаб/название-образа

Copy

После выполнения команды образ будет размещен в вашем репозитории на Docker Hub.

[](https://firstvds.ru/sites/default/files/all_images/instructions/install_docker/image10.png)

Теперь вы можете использовать команду **docker pull ваш-логин-на-докер-хаб/название-образа**, чтобы загрузить образ на другой компьютер и использовать его для запуска нового контейнера.