

RuVDS (<https://ruvds.com/ru/>) / Справочник ([/ru/helpcenter/](https://ruvds.com/ru/helpcenter/)) / Развертывание ПО на VPS сервере (<https://ruvds.com/ru/help/4-programms-install/>) / Как развернуть свое Docker хранилище в Ubuntu 20.04

🔍 Запрос

ИСКАТЬ



# Как развернуть свое Docker хранилище в Ubuntu 20.04

👁 11964 просмотров 👍 13 📅 2021-06-08 📅 2021-06-16

В данном руководстве мы изучим, как развернуть своё собственное Docker хранилище на сервере (<https://ruvds.com/ru-rub/my/servers/>), работающем под управлением Ubuntu 20.04.


Хранилище Docker (<https://docs.docker.com/registry/>) представляет из себя приложение, которое управляет хранением и доставкой образов контейнеров Docker. Хранилища собирают образа контейнеров и снижают время сборки для разработчиков. Образы Docker обеспечивают создание одной и той же среды выполнения благодаря виртуализации, при том, что создание образа может потребовать значительных временных затрат. Другими словами, разработчики имеют возможность загрузить из хранилища сжатый образ, содержащий все необходимые компоненты, вместо того, чтобы устанавливать зависимости и пакеты отдельно для использования Docker.

У Docker есть бесплатное общедоступное хранилище, Docker Hub, которое может припарковать ваши образы Docker. Тонкость состоит в том, что не всегда есть желание хранить свои образы там, где они будут общедоступны. Образы обычно содержат весь необходимый для запуска код, поэтому использование собственного реестра предпочтительнее для хранения проприетарного программного обеспечения.

Мы будем использовать Docker Compose, чтобы определять конфигурации запуска ваших контейнеров Docker. А также, мы будем использовать веб-сервер Nginx для передачи трафика сервера из интернета в работающий контейнер Docker. Изучив содержимое данного руководства, вы сможете поместить образ Docker в своё личное хранилище, а также, безопасно извлечь его с удалённого сервера.

## Подготовка серверов

В данном мануале мы будем использовать два сервера. Один из них будет выступать в качестве хоста для вашего частного Docker хранилища, а другой – в качестве сервера-клиента. Оба VPS (<https://ruvds.com/ru-rub/my/servers/>) работают под управлением операционной системы Ubuntu 20.04. Все работы на серверах мы будем производить под учётной записью, не являющейся root-ом, но имеющей привилегии sudo. Для настройки межсетевого экрана на серверах мы будем использовать интерфейс UFW.

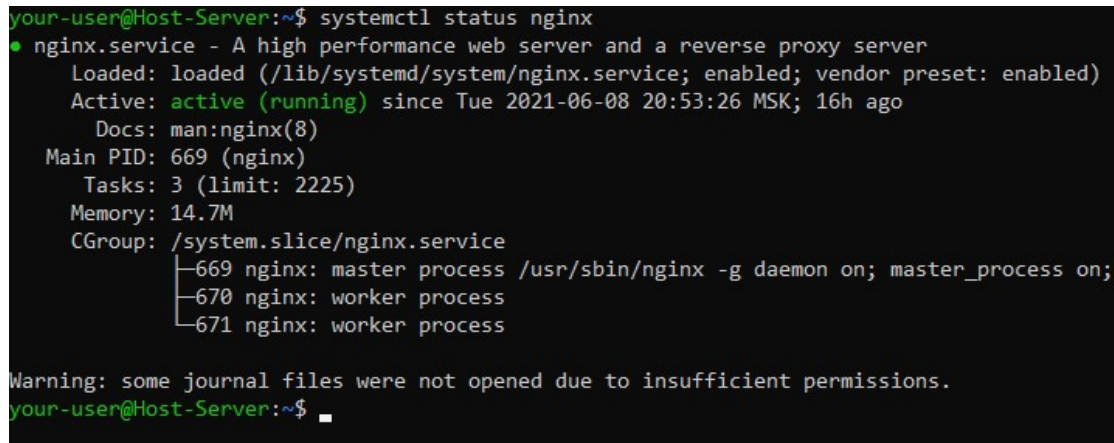
Так как в качестве веб-сервера мы планируем использовать Nginx, то давайте перейдём к его настройке. Действия по настройке Nginx нужно будет произвести на сервере, выступающем в качестве хоста для 

нашего хранилища. Сначала необходимо установить Nginx:

```
$ sudo apt update  
$ sudo apt install nginx
```

Теперь добавьте Nginx в список приложений, доступ для которых разрешён в нашем брандмауэре, и проверьте статус службы Nginx:

```
$ sudo ufw allow 'Nginx HTTP'  
$ systemctl status nginx
```



```
your-user@Host-Server:~$ systemctl status nginx  
● nginx.service - A high performance web server and a reverse proxy server  
   Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)  
   Active: active (running) since Tue 2021-06-08 20:53:26 MSK; 16h ago  
     Docs: man:nginx(8)  
  Main PID: 669 (nginx)  
    Tasks: 3 (limit: 2225)  
   Memory: 14.7M  
    CGroup: /system.slice/nginx.service  
            └─669 nginx: master process /usr/sbin/nginx -g daemon on; master_process on;  
              └─670 nginx: worker process  
                └─671 nginx: worker process  
  
Warning: some journal files were not opened due to insufficient permissions.  
your-user@Host-Server:~$
```

Если сервис запущен вы можете проверить его доступность через веб-интерфейс. Для чего в браузере наберите IP-адрес вашего сервера-хоста:

# Welcome to nginx!

If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to [nginx.org](https://nginx.org).  
Commercial support is available at [nginx.com](https://nginx.com).

*Thank you for using nginx.*

Далее, вам необходимо будет добавить виртуальный хост для вашего домена. В нашем мануале мы будем использовать домен `my-domain.host`. Обратите внимание, что А-запись домена, с которым вы будете производить дальнейшие действия, должна соответствовать IP-адресу вашего сервера-хоста.

Итак, создайте каталог для вашего сайта и предоставьте ему соответствующие права:

```
$ sudo mkdir -p /var/www/my-domain.host/html  
$ sudo chown -R $USER:$USER /var/www/my-domain.host/html  
$ sudo chmod -R 755 /var/www/my-domain.host
```

Теперь в созданном каталоге создайте файл страницы сайта `index.html` :



```
$ sudo nano /var/www/my-domain.host/html/index.html
```

Наполните его содержимым:

```
<html>
  <head>
    <title>
      Domain on Nginx
    </title>
  </head>
  <body>
    <h1>
      Connection to my-domain.host is created successfully!!!
    </h1>
  </body>
</html>
```

Закройте файл, сохранив изменения (Ctrl+X, после чего Y и Enter).

В директории /etc/nginx/sites-available/ создайте файл my-domain.host :

```
$ sudo nano /etc/nginx/sites-available/my-domain.host
```

Скопируйте в него следующие строки:

```
server
{
    listen 80;
    listen [::]:80;
    root /var/www/my-domain.host/html;
    index index.html index.htm index.nginx-debian.html;
    server_name my-domain.host www.my-domain.host;
    location /
    {
        try_files $uri $uri/ =404;
    }
}
```

Закройте файл с сохранением изменений и создайте ссылку в /etc/nginx/sites-enabled/ :

```
$ sudo ln -s /etc/nginx/sites-available/my-domain.host /etc/nginx/sites-enabled/
```

Отредактируйте файл /etc/nginx/nginx.conf :

```
$ sudo nano /etc/nginx/nginx.conf
```

В этом файле вам нужно будет раскомментировать строку, которая содержит server\_names\_hash\_bucket\_size .

Для проверки корректности синтаксиса Nginx наберите:

```
$ sudo nginx -t
```

Должно быть:

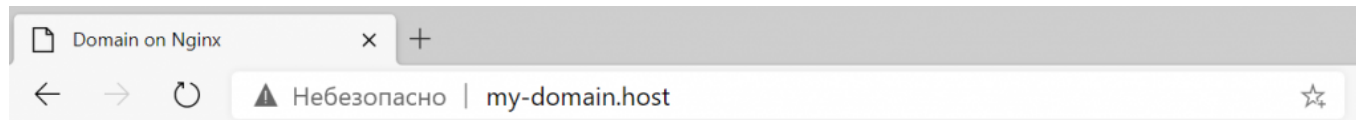


```
your-user@Host-Server:~$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
your-user@Host-Server:~$
```

Перезапустите Nginx:

```
$ sudo systemctl restart nginx
```

После этого ваш домен должен стать доступным по своему доменному имени:



## Connection to my-domain.host is created successfully!!!

И нам останется только настроить редирект всего трафика с протокола HTTP на HTTPS. Мы осуществим это при помощи центра сертификации Let's Encrypt.

Сначала необходимо будет установить пакет snap:

```
$ sudo apt install snapd
```

Затем с его помощью проинсталируйте Certbot:

```
$ sudo snap install --classic certbot
$ sudo ln -s /snap/bin/certbot /usr/bin/certbot
```

Далее, в брандмауэре необходимо разрешить трафик по протоколу HTTPS, закрыв трафик по HTTP:


```
$ sudo ufw allow 'Nginx Full'
$ sudo ufw delete allow 'Nginx HTTP'
$ sudo ufw status
```

```
your-user@Host-Server:~$ sudo ufw status
Status: active

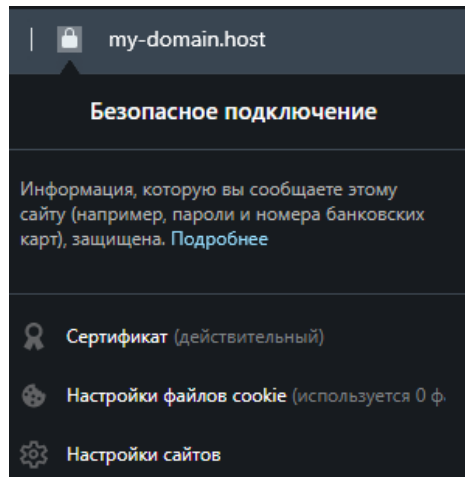
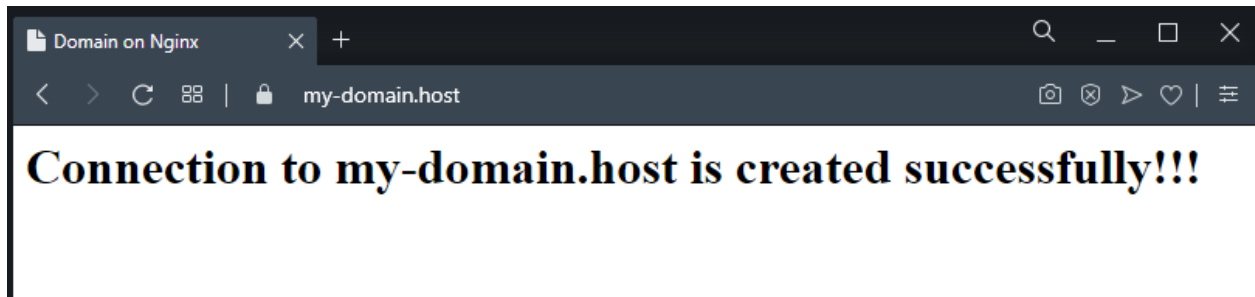
To Action From
--
OpenSSH ALLOW Anywhere
Nginx Full ALLOW Anywhere
OpenSSH (v6) ALLOW Anywhere (v6)
Nginx Full (v6) ALLOW Anywhere (v6)
```

Теперь, необходимо запустить Certbot с помощью плагина `nginx` для указания доменов, которые будут использовать сертификаты:

```
$ sudo certbot --nginx
```

Здесь вам нужно будет указать свой E-mail, согласиться с предоставляемыми условиями и указать домен, для которого требуется активировать протокол HTTPS. 

С этого момента подключение к вашему домену защищено:



Далее, переходим с настройке Docker.

## Установка Docker

Пакет Docker будем устанавливать из официального репозитория Docker. Чтобы сделать это, мы добавим новый источник пакетов и ключ GPG от Docker, чтобы быть уверенными в валидности загрузки пакета. После чего установим пакет.

Проинсталлируйте несколько необходимых пакетов, которые позволят установщику использовать пакеты обновлений по протоколу HTTPS:

```
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
```

Теперь, добавьте в систему ключ GPG для официального репозитория Docker:

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

Далее, добавьте репозиторий Docker и запустите обновление базы данных пакетов обновлений:

```
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu focal stable"
$ sudo apt update
```

Убедитесь, что дальнейшая установка будет произведена из репозитория Docker, а не из дефолтного репозитория Ubuntu:

```
$ apt-cache policy docker-ce
```



```
your-user@Host-Server:~$ apt-cache policy docker-ce
docker-ce:
  Installed: (none)
  Candidate: 5:20.10.7~3-0~ubuntu-focal
  Version table:
     5:20.10.7~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:20.10.6~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:20.10.5~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:20.10.4~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:20.10.3~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:20.10.2~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:20.10.1~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:20.10.0~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:19.03.15~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:19.03.14~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:19.03.13~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:19.03.12~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:19.03.11~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:19.03.10~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
     5:19.03.9~3-0~ubuntu-focal 500
        500 https://download.docker.com/linux/ubuntu focal/stable amd64 Packages
your-user@Host-Server:~$
```

Обратите внимание, что `docker-ce` не установлен, но готов к установке из репозитория Ubuntu 20.04.

И наконец, запустите установку Docker:

```
$ sudo apt install docker-ce
```

Теперь Docker установлен, служба запущена и процесс доступен для старта при загрузке системы. Проверьте:

```
$ sudo systemctl status docker
```





```

your-user@Host-Server:~$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2021-06-08 21:24:18 MSK; 15min ago
     TriggeredBy: ● docker.socket
       Docs: https://docs.docker.com
    Main PID: 8852 (dockerd)
      Tasks: 10
     Memory: 41.0M
    CGroup: /system.slice/docker.service
            └─8852 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock

Jun 08 21:24:18 Host-Server dockerd[8852]: time="2021-06-08T21:24:18.014329806+03:00" level=warni
Jun 08 21:24:18 Host-Server dockerd[8852]: time="2021-06-08T21:24:18.014423107+03:00" level=warni
Jun 08 21:24:18 Host-Server dockerd[8852]: time="2021-06-08T21:24:18.014551008+03:00" level=warni
Jun 08 21:24:18 Host-Server dockerd[8852]: time="2021-06-08T21:24:18.014795110+03:00" level=info
Jun 08 21:24:18 Host-Server dockerd[8852]: time="2021-06-08T21:24:18.170085596+03:00" level=info
Jun 08 21:24:18 Host-Server dockerd[8852]: time="2021-06-08T21:24:18.243073647+03:00" level=info
Jun 08 21:24:18 Host-Server dockerd[8852]: time="2021-06-08T21:24:18.269326981+03:00" level=info
Jun 08 21:24:18 Host-Server dockerd[8852]: time="2021-06-08T21:24:18.269454182+03:00" level=info
Jun 08 21:24:18 Host-Server systemd[1]: Started Docker Application Container Engine.
Jun 08 21:24:18 Host-Server dockerd[8852]: time="2021-06-08T21:24:18.300420658+03:00" level=info
lines 1-21/21 (END)

```

Также, нам нужно будет установить инструмент, который позволяет осуществлять запуск сред выполнения приложений с несколькими контейнерами. Это – Docker Compose. Чтобы загрузить самую свежую Docker Compose, посмотрите её номер на странице официального репозитория Github (<https://github.com/docker/compose>). Наберите в командой строке команду, которая загрузит исполняемый файл:

```
$ 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
```

```

your-user@Host-Server:~$ 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
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           % Dload  % Upload   Dload  Upload  Total  Spent  Left  Speed
100 633 100 633 0 0 3459 0 --:--:-- --:--:-- --:--:-- 3478
100 12.1M 100 12.1M 0 0 8608k 0 0:00:01 0:00:01 --:--:-- 14.6M
your-user@Host-Server:~$

```

После этого задайте загруженному файлу соответствующие права. Это позволит сделать `docker-compose` исполняемым файлом:

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

Проверьте версию установленного компонента:

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

```

your-user@Host-Server:~$ docker-compose --version
docker-compose version 1.29.2, build 5becea4c
your-user@Host-Server:~$

```

Теперь всё готово для установки и настройки Docker хранилища.



# Установка и настройка Docker хранилища

В командной строке Docker очень полезен когда вы запускаете и тестируете контейнеры. Но для больших развёртываний, включающих несколько параллельно выполняющихся контейнеров, она оказывается очень громоздкой.

С помощью Docker Compose вы можете создать один файл `.yaml` для настройки конфигурации каждого контейнера и данных, которые нужны контейнерам для взаимодействия друг с другом. Вы можете использовать `docker-compose` как инструмент командной строки для выдачи команд всем компонентам, составляющих ваше приложение, и управления ими как группой.

Docker хранилище само по себе является приложением с несколькими компонентами, поэтому для управления им вы будете использовать Docker Compose. Для запуска хранилища, необходимо будет создать файл `docker-compose.yaml`, который определит как хранилище, так и расположение на диске, где оно будет хранить данные.

На сервере-хосте создайте каталог `docker-registry` для хранения настроек и перейдите в него:

```
$ mkdir ~/docker-registry
$ cd ~/docker-registry
```

Там создайте каталог `data`, где будут храниться образы:

```
$ mkdir data
```

Создайте файл `docker-compose.yaml`:

```
$ sudo nano docker-compose.yaml
```

Добавьте в него следующие строки, которые определяют основной экземпляр хранилища:

```
version: '3'

services:
  registry:
    image: registry:2
    ports:
      - "5000:5000"
    environment:
      REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY: /data
    volumes:
      - ./data:/data
```

Здесь:

- `registry` – название первой службы;
- `registry:2` – определение образа в `registry` версии 2;
- `ports` – сопоставление порта 5000 на хосте порту 5000 на контейнере;
- `REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY` – переменная, определяющая каталог для хранения данных;
- `volumes` – сопоставление каталога `/data` на сервере-хосте каталогу `/data` в контейнере.

Теперь запустите созданную конструкцию:

```
$ docker-compose up
```





Контейнер и его зависимости будут загружены и запущены:

```
your-user@Host-Server:~/docker-registry$ docker-compose up
Recreating docker-registry_registry_1 ... done
Attaching to docker-registry_registry_1
registry_1 | time="2021-06-09T18:09:19.87124904Z" level=warning msg="No HTTP secret provided - generated random secret.
This may cause problems with uploads if multiple registries are behind a load-balancer. To provide a shared secret, fil
1 in http.secret in the configuration file or set the REGISTRY_HTTP_SECRET environment variable." go.version=go1.11.2 in
stance.id=df1b73f4-8ed0-4c01-981d-863d477caa0f service=registry version=v2.7.1
registry_1 | time="2021-06-09T18:09:19.887541346Z" level=info msg="redis not configured" go.version=go1.11.2 instance.i
d=df1b73f4-8ed0-4c01-981d-863d477caa0f service=registry version=v2.7.1
registry_1 | time="2021-06-09T18:09:19.872194152Z" level=info msg="Starting upload purge in 46m0s" go.version=go1.11.2
instance.id=df1b73f4-8ed0-4c01-981d-863d477caa0f service=registry version=v2.7.1
registry_1 | time="2021-06-09T18:09:19.887541346Z" level=info msg="using inmemory blob descriptor cache" go.version=go1
.11.2 instance.id=df1b73f4-8ed0-4c01-981d-863d477caa0f service=registry version=v2.7.1
registry_1 | time="2021-06-09T18:09:19.889851375Z" level=info msg="listening on [::]:5000" go.version=go1.11.2 instance
.id=df1b73f4-8ed0-4c01-981d-863d477caa0f service=registry version=v2.7.1
^CGracefully stopping... (press Ctrl+C again to force)
Stopping docker-registry_registry_1 ... done
your-user@Host-Server:~/docker-registry$
```

Обратите внимание на сообщение No HTTP secret provided . К нему мы ещё вернёмся в данном руководстве. Последняя строка содержит сообщение о том, что запущено прослушивание на порту 5000 .

Чтобы прервать выполнение команды, нажмите Ctrl+C .

## Настройка переадресации портов Nginx

Ранее, мы уже настроили доступ к нашему домену по протоколу HTTPS. Чтобы открыть защищённое хранилище, вам необходимо будет всего лишь настроить Nginx в части перенаправления трафика от домена к контейнеру хранилища.

Откройте для редактирования созданный ранее файл, содержащий настройки вашего сервера:

```
$ sudo nano /etc/nginx/sites-available/my-domain.host
```

Найдите там блок location :

```
location /
{
    try_files $uri $uri/ =404;
}
```

Необходимо переадресовать трафик на порт 5000 , на котором хранилище будет слушать трафик. Также, можно добавить заголовки к запросам, направленным хранилищу, которое от имени сервера предоставляет дополнительную информацию о самом запросе. Замените содержимое блока location следующими строками:

```
location / {
    # Do not allow connections from docker 1.5 and earlier
    # docker pre-1.6.0 did not properly set the user agent on ping, catch "Go *" user agents
    if ($http_user_agent ~ "^(docker\/1\.(3|4|5(?:!\.|[0-9]-dev))|Go ).*$" ) {
        return 404;
    }

    proxy_pass http://localhost:5000;
    proxy_set_header Host $http_host; # required for docker client's sake
    proxy_set_header X-Real-IP $remote_addr; # pass on real client's IP
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_read_timeout 900;
}
```



Блок `if` проверяет пользовательский агент запроса, верифицирует версию клиента Docker выше 1.5 и определяет, что это не приложение Go, которое пытается получить доступ.

Сохраните файл перед его закрытием. Для применения новых настроек перезапустите Nginx:

```
$ sudo systemctl restart nginx
```

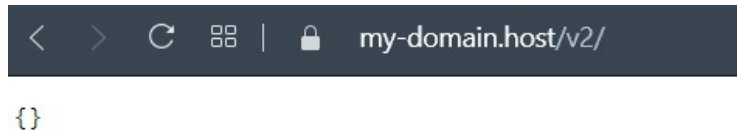
Чтобы убедиться, что Nginx правильно перенаправляет трафик в контейнер хранилища на порту 5000, запустите его:

```
$ cd ~/docker-registry
$ docker-compose up
```

И теперь, наберите в браузере:

```
https://my-domain.host/v2
```

Вы увидите пустой объект JSON:



В терминале же вы сможете видеть, что в последней строке запрос `GET` был сделан в `/v2/`. Это указывает на какую конечную точку вы отправили запрос из своего браузера. Другими словами, контейнер получил запрос, который вы сделали, и вернул ответ `{}`. Код `http.response.status=200` в последних строках означает, что контейнер справился с запросом успешно.

```
registry_1 | time="2021-06-09T20:46:42.296655363Z" level=info msg="response completed" go.version=go1.11.2 http.request
.host=my-domain.host http.request.id=8bf308a4-6fdc-4e02-86db-0e14686c6a2b http.request.method=GET http.request.remoteadd
r=176.59.45.97 http.request.uri="/v2/" http.request.useragent="Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537
.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36 OPR/76.0.4017.177" http.response.contenttype="application/jso
n; charset=utf-8" http.response.duration=2.467031ms http.response.status=200 http.response.written=2
registry_1 | 172.18.0.1 - - [09/Jun/2021:20:46:42 +0000] "GET /favicon.ico HTTP/1.0" 404 19 "https://my-domain.host/v2/
" "Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/90.0.4430.212 Safari/537.36 O
PR/76.0.4017.177"
```

Чтобы прервать выполнение команды, нажмите `Ctrl+C`.

## Настройка аутентификации

Nginx позволяет настроить аутентификацию HTTP для сайтов, которыми Nginx управляет. Именно это вы можете использовать для ограничения доступа к вашему хранилищу Docker. Чтобы этого добиться, необходимо создать файл аутентификации при помощи `htpasswd` и добавить в него комбинацию пользователь-пароль.

Утилиту `htpasswd` вы можете получить, например, установив пакет `apache2-utils`:

```
$ sudo apt install apache2-utils
```



Теперь необходимо сохранить файл проверки подлинности с учетными данными в `~/docker-registry/auth/`:

```
$ mkdir ~/docker-registry/auth
$ cd ~/docker-registry/auth
```

На следующем шаге создайте своего первого пользователя, заменив `user1` на какое-нибудь своё имя учётной записи. Здесь, опция `-B` нужна для применения алгоритма `bcrypt`:

```
$ htpasswd -Bc registry.password user1
```

Далее, введите пароль для создаваемого пользователя:

```
your-user@Host-Server:~/docker-registry/auth$ htpasswd -Bc registry.password user1
New password:
Re-type new password:
Adding password for user user1
your-user@Host-Server:~/docker-registry/auth$
```

Теперь учётная запись `user1` и её пароль добавлены в `registry.password`.

Если вам необходимо добавить других пользователей, повторите предыдущую команду, но с другим именем пользователя и без использования флага `-c`, который отвечает за создание нового файла:

```
$ htpasswd -B registry.password user2
```

Теперь нужно заставить Docker использовать созданный файл аутентификации. Для этого отредактируйте его:

```
$ sudo nano ~/docker-registry/docker-compose.yml
```

Добавьте в файл выделенные строки:

```
version: '3'

services:
  registry:
    image: registry:2
    ports:
      - "5000:5000"
    environment:
      REGISTRY_AUTH: htpasswd
      REGISTRY_AUTH_HTPASSWD_REALM: Registry
      REGISTRY_AUTH_HTPASSWD_PATH: /auth/registry.password
      REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY: /data
    volumes:
      - ./auth:/auth
      - ./data:/data
```

Таким образом, мы добавили переменные окружения, указывающие на необходимость использования аутентификации HTTP, и указали путь к созданному файлу `htpasswd`. Для `REGISTRY_AUTH` мы указали значение `htpasswd`, которое аутентифицирует используемую схему. Переменная `REGISTRY_AUTH_HTPASSWD_PATH` содержит путь к файлу аутентификации. Переменная `REGISTRY_AUTH_HTPASSWD_REALM` означает имя области `htpasswd`.

Также, мы смонтировали каталог `./auth/`, чтобы сделать файл доступным внутри контейнера хранилища.

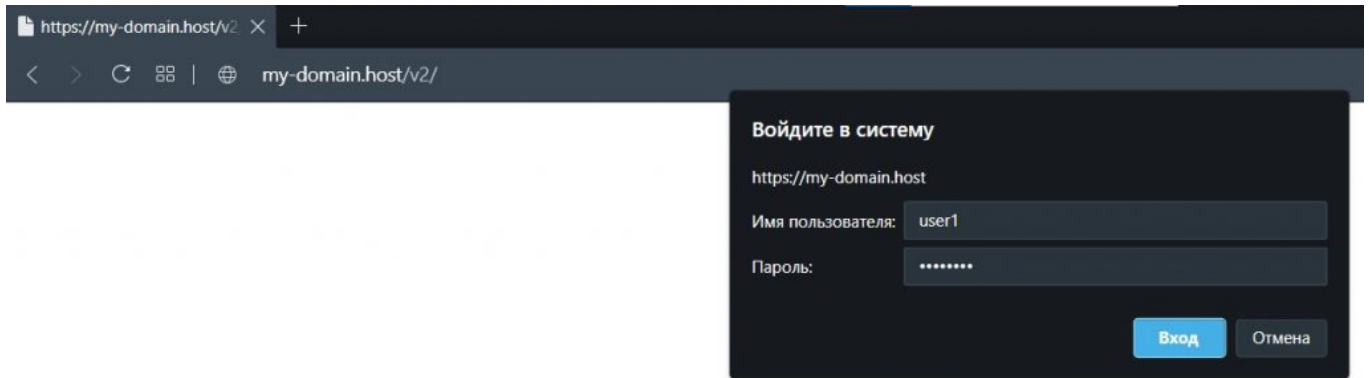


Закройте файл сохранив внесённые изменения.

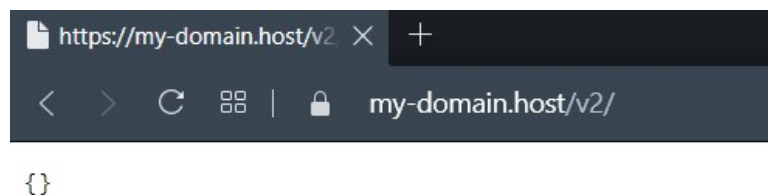
Теперь можно проверить корректность работы системы аутентификации:

```
$ cd ~/docker-registry  
$ docker-compose up
```

После этого, обновите страницу браузера, в котором открыт ваш домен. Система должна попросить вас ввести имя пользователя и пароль:



Введите учётные данные и, в случае успешной аутентификации, вы увидите пустой объект JSON:



Это означает, что вы успешно прошли проверку подлинности и получили доступ к хранилищу.

Как и раньше, выход осуществляется через `Ctrl+C`.

Теперь ваше хранилище защищено, и доступ к нему можно осуществить только после успешной проверки подлинности. Далее, мы настроим его на запуск в фоновом режиме, при этом хранилище будет устойчиво к перезагрузкам, запускаясь каждый раз автоматически.

## Запуск Docker хранилища как службы

Для того, чтобы контейнер хранилища запускался каждый раз при загрузке системы или после её сбоя, необходимо проинструктировать Docker Compose всегда поддерживать его работу.

Отредактируйте файл `docker-compose.yml` :

```
$ sudo nano ~/docker-registry/docker-compose.yml
```



Добавьте в файл выделенную строку:

```
version: '3'

services:
  registry:
    restart: always
    image: registry:2
    ports:
      - "5000:5000"
    environment:
      REGISTRY_AUTH: htpasswd
      REGISTRY_AUTH_HTPASSWD_REALM: Registry
      REGISTRY_AUTH_HTPASSWD_PATH: /auth/registry.password
      REGISTRY_STORAGE_FILESYSTEM_ROOTDIRECTORY: /data
    volumes:
      - ./auth:/auth
      - ./data:/data
```

Настройка `restart` гарантирует нам, что контейнер выдержит перезапуск системы. Сохраните изменения и закройте файл.

А теперь, запустите ваше хранилище как фоновый процесс при помощи опции `-d`:

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

С этого момента ваше хранилище работает в фоновом режиме, и поэтому, вы можете спокойно закрыть сессию SSH, и даже перезагрузить сервер. Это не окажет на хранилище никакого эффекта.

## Увеличение размера загружаемых файлов для Nginx

Перед тем, как помещать образ в хранилище, необходимо убедиться, что наше хранилище сможет обрабатывать загрузку файлов большого объема.

По умолчанию предельный размер загружаемого файла в Nginx составляет 1MB. Этого явно не достаточно для образа Docker. Чтобы это исправить, необходимо внести изменения в главный конфигурационный файл Nginx. Он расположен в директории `/etc/nginx/`. Откройте его для редактирования, набрав в командной строке:

```
$ cd /etc/nginx
$ sudo nano nginx.conf
```

Найдите там секцию `http` и добавьте в неё выделенную строку:

```
http {
    ##
    # Basic Settings
    ##
    client_max_body_size 16384m;
    sendfile on;
    tcp_nopush on;
    tcp_nodelay on;
    keepalive_timeout 65;
    types_hash_max_size 2048;
```

Значение параметра `client_max_body_size` теперь установлено в 16384 MB. Это означает, что максимальный размер загружаемого файла равно 16GB.





Закройте файл с сохранением изменений и перезапустите Nginx:

```
$ sudo systemctl restart nginx
```

## Публикация в Docker хранилище

Настало время попробовать загрузить образ на наше хранилище. Так как у нас пока нет доступных образов, мы можем, в качестве теста, использовать образ `ubuntu`, который можно скачать с общедоступного Docker хранилища.

Действия по публикации образа в наше хранилище мы будем производить на нашем втором сервере, который мы обозначили как сервер-клиент.

Для начала, чтобы на этом сервере запускать команду `docker` не используя полномочия `sudo`, необходимо добавить вашего пользователя в группу `docker`:

```
$ sudo usermod -aG docker your-user
```

Теперь, для того, чтобы загрузить образ `ubuntu`, запустить его и получить доступ к его оболочке, в командной строке вашего сервера-клиента наберите:

```
$ docker run -t -i ubuntu /bin/bash
```

Опции `-i` и `-t` позволяют вам получить интерактивный доступ к оболочке внутри контейнера.

Теперь, когда вы подключились к оболочке, в корне системы создайте файл `my.IMAGE`:

```
root@2b8dcf17a0db:/# touch /my.IMAGE
```

Создав такой файл, мы изменили первоначальный контейнер. Наличие этого файла позже позволит нам понять, что мы имеем дело именно с точно таким же контейнером.

Выход из оболочки контейнера осуществляется при помощи `CNTL D`, либо командой:

```
root@2b8dcf17a0db:/# exit
```


Далее, создайте новый образ из контейнера, в который вы только что внесли изменения:

```
$ docker commit $(docker ps -lq) new-ubuntu-image
```

```
your-user@Client-Server:~$ docker commit $(docker ps -lq) new-ubuntu-image  
sha256:c9dba412ae64a8decf495b977915856012da42b455683eae839e8850bcfb7d7d  
your-user@Client-Server:~$
```

Новый образ теперь доступен локально, и, значит, вы можете передать его в ваше хранилище. Сначала подключитесь к нему:

```
$ docker login https://my-domain.host
```

Введите имя пользователя и пароль для получения доступа к контейнеру. В нашем случае, это данные 

учётной записи `user1`.

В случае успешной аутентификации система выведет подобное сообщение:

```
your-user@Client-Server:~$ docker login https://my-domain.host
Username: user1
Password:
WARNING! Your password will be stored unencrypted in /home/your-user/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
your-user@Client-Server:~$
```

Теперь, когда вы подключились, переименуйте созданный образ:

```
$ docker tag new-ubuntu-image my-domain.host/new-ubuntu-image
```

И наконец, передайте только что отмеченный образ в ваше хранилище:

```
$ docker push my-domain.host/new-ubuntu-image
```

Вы получите вывод, аналогичный следующему:

```
your-user@Client-Server:~$ docker push my-domain.host/new-ubuntu-image
Using default tag: latest
The push refers to repository [my-domain.host/new-ubuntu-image]
2f140462f3bc: Pushed
63c99163f472: Pushed
ccdbb80308cc: Pushed
latest: digest: sha256:86ac87f73641c920fb42cc9612d4fb57b5626b56ea2a19b894d0673fd5b4f2e9 size: 943
your-user@Client-Server:~$
```

После того, как мы отправили образ в хранилище, давайте лишние образы удалим. Сначала необходимо вывести на экран список локально доступных образов:

```
$ docker images
```

```
your-user@Client-Server:~$ docker images
REPOSITORY              TAG         IMAGE ID      CREATED        SIZE
new-ubuntu-image        latest     c9dba412ae64  2 minutes ago  72.7MB
my-domain.host/new-ubuntu-image  latest     c9dba412ae64  2 minutes ago  72.7MB
ubuntu                  latest     7e0aa2d69a15  6 weeks ago   72.7MB
```

Мы удалим все образы, кроме первоначально загруженного образа `ubuntu`:

```
$ docker rmi new-ubuntu-image
$ docker rmi my-domain.host/new-ubuntu-image
```

Если ещё раз вывести список образов, то вы увидите, что в списке остался только образ `ubuntu`:

```
$ docker images
```



```
your-user@Client-Server:~$ docker images
REPOSITORY    TAG       IMAGE ID       CREATED        SIZE
ubuntu        latest    7e0aa2d69a15   6 weeks ago    72.7MB
```

Таким образом, мы проверили, как хранилище обрабатывает аутентификацию пользователей при их подключении и как позволяет пользователям, прошедшим проверку, передавать образ в хранилище. Теперь попробуем забрать образ из нашего хранилища.

## Извлечение из Docker хранилища

Для загрузки образа из хранилища подключитесь к нему набрав на вашем сервере-клиенте команду:

```
$ docker login https://my-domain.host
```

Не забывайте использовать домен своего сервера-хоста вместо `my-domain.host`.

Для того, чтобы принять файл образа `new-ubuntu-image`, выполните следующую команду:

```
$ docker pull my-domain.host/new-ubuntu-image
```

После того, как образ загрузится, выполните команду для просмотра списка имеющихся на сервере-клиенте образов:

```
$ docker images
```

```
your-user@Client-Server:~$ docker images
REPOSITORY          TAG       IMAGE ID       CREATED        SIZE
my-domain.host/new-ubuntu-image  latest    c9dba412ae64   40 minutes ago 72.7MB
ubuntu              latest    7e0aa2d69a15   6 weeks ago    72.7MB
your-user@Client-Server:~$
```

Здесь вы видите, что образ `new-ubuntu-image` снова присутствует на сервере-клиенте. Значит, к нему можно подключиться:

```
$ docker run -t -i my-domain.host/new-ubuntu-image
```

Подключившись, посмотрите содержимое корня системы:

```
root@65c69421278c:/# ls -l
```



```

root@65c69421278c:/# ls -l
total 52
lrwxrwxrwx   1 root root    7 Apr 16 05:11 bin -> usr/bin
drwxr-xr-x   2 root root 4096 Apr 15 2020 boot
drwxr-xr-x   5 root root  360 Jun 11 19:40 dev
drwxr-xr-x   1 root root 4096 Jun 11 19:40 etc
drwxr-xr-x   2 root root 4096 Apr 15 2020 home
lrwxrwxrwx   1 root root    7 Apr 16 05:11 lib -> usr/lib
lrwxrwxrwx   1 root root    9 Apr 16 05:11 lib32 -> usr/lib32
lrwxrwxrwx   1 root root    9 Apr 16 05:11 lib64 -> usr/lib64
lrwxrwxrwx   1 root root   10 Apr 16 05:11 libx32 -> usr/libx32
drwxr-xr-x   2 root root 4096 Apr 16 05:11 media
drwxr-xr-x   2 root root 4096 Apr 16 05:11 mnt
-rw-r--r--   1 root root    0 Jun 11 19:29 my.IMAGE
drwxr-xr-x   2 root root 4096 Apr 16 05:11 opt
dr-xr-xr-x 128 root root    0 Jun 11 19:40 proc
drwx-----   1 root root 4096 Jun 11 19:29 root
drwxr-xr-x   1 root root 4096 Apr 23 22:21 run
lrwxrwxrwx   1 root root    8 Apr 16 05:11/sbin -> usr/sbin
drwxr-xr-x   2 root root 4096 Apr 16 05:11 srv
dr-xr-xr-x  13 root root    0 Jun 11 19:40 sys
drwxrwxrwt   2 root root 4096 Apr 16 05:33 tmp
drwxr-xr-x   1 root root 4096 Apr 16 05:11 usr
drwxr-xr-x   1 root root 4096 Apr 16 05:32 var
root@65c69421278c:/#

```

Увидев файл `my.IMAGE`, можно сделать вывод, что это – экземпляр именно того образа, в который мы вносили коррективы перед загрузкой в наше хранилище.

Закрыть оболочку можно командой `exit`, либо нажав `Ctrl+D`.

## Заключение

В данном руководстве мы увидели, как развернуть своё приватное Docker хранилище на сервере, работающем под управлением Ubuntu 20.04. Мы защитили подключение к развёрнутому хранилищу при помощи системы проверки подлинности учётных записей. А также, мы попробовали создать контейнер с образом Docker, передать его в хранилище и снова загрузить на сервер, использовавшийся нами как клиент нашего Docker хранилища.



(<https://twitter.com/intent/tweet?url=https://ruvds.com/ru/helpcenter/docker-registry-ubuntu/>)



(mailto:?Subject=Интересная статья&body=https://ruvds.com/ru/helpcenter/docker-registry-ubuntu/)

Статья полезна?

👍 13 Да

👎 3 Нет







Похожие статьи

- 📄 Установка Nagios в Ubuntu 20.04  
(<https://ruvds.com/ru/helpcenter/ustanovka-nagios->
- 📄 Как установить и настроить Samba на Ubuntu 20.04



[v-ubuntu-20-04/](#)[\(https://ruvds.com/ru/helpcenter/kak-ustanovit-i-nastroit-samba-na-ubuntu/\)](https://ruvds.com/ru/helpcenter/kak-ustanovit-i-nastroit-samba-na-ubuntu/)

-  Создание и настройка почтового домена, импорт почты из других сервисов с помощью ISPmanager 6 Lite (<https://ruvds.com/ru/helpcenter/sozдание-i-nastroyka-pochtovogo-domena/>)
-  Настройка альтернативных версий PHP в ISPmanager 6 Lite (<https://ruvds.com/ru/helpcenter/nastroyka-alternativnyh-versiy-php-v-ispmanager/>)
-  Массовый дефейс серверов "1С-Битрикс: Управление сайтом" и мероприятия по противодействию ему (<https://ruvds.com/ru/helpcenter/massovy-defeis-serverov/>)
-  Установка Docker на Windows Server 2019 (<https://ruvds.com/ru/helpcenter/ustanovka-docker-na-windows-server-2019/>)

## 1 комментарий о "Как развернуть свое Docker хранилище в Ubuntu 20.04"

**Андрей** says:

У меня не получилось по данной методичке установить docker compose. Помогла команда: `sudo apt-get install docker-compose-plugin` (<https://docs.docker.com/compose/install/linux/#install-using-the-repository>)

2023-05-24 at 09:24

## Оставить комментарий

Комментарий

Имя \*

Email \*

Вебсайт

☐ Сохранить моё имя, email и адрес сайта в этом браузере для последующих моих комментариев.




Я не робот

reCAPTCHA

[Конфиденциальность](#) - [Условия использования](#)

ОТПРАВИТЬ КОММЕНТАРИЙ

## Популярные статьи

 Как подключиться по RDP из-под MacOS (<https://ruvds.com/ru/helpcenter/rdp-macos/>)





- 📄 Как установить PostgreSQL и pgAdmin4 в Ubuntu 20.04 (<https://ruvds.com/ru/helpcenter/postgresql-pgadmin-ubuntu/>)
- 📄 Как настроить OpenVPN Server на Ubuntu 20.04 (<https://ruvds.com/ru/helpcenter/kak-nastroit-openvpn-server-na-ubuntu-20-04/>)
- 📄 Управление портами в Linux (Ubuntu/Debian/CentOS) (<https://ruvds.com/ru/helpcenter/upravlenie-portami-v-linux/>)
- 📄 Как настроить FTP на Ubuntu 20.04 LTS (<https://ruvds.com/ru/helpcenter/kak-nastroit-ftp-na-ubuntu-20-04-lts/>)

## Разделы Справочника

- 📁 Настройка VPS Сервера (<https://ruvds.com/ru/help/2-vps-settings/>)
- 📁 Начало работы (<https://ruvds.com/ru/help/1-for-beginners/>)
- 📁 Особенности виртуального сервера (<https://ruvds.com/ru/help/6-vps-features/>)
- 📁 Партнерам (<https://ruvds.com/ru/help/5-for-partners/>)
- 📁 **Развертывание ПО на VPS сервере (<https://ruvds.com/ru/help/4-programms-install/>)**
- 📁 Сетевые настройки сервера (<https://ruvds.com/ru/help/3-networking/>)

GAME OVERNIGHT (<https://gameovernight.ru/>)

CLOUDRUSSIA (<http://cloudrussia.ru/>)

STRATONET (<https://stratonet.net/>)

SPACE DC (<https://sputnik.rucloud.host/>)

### VPS/VDS СЕРВЕРЫ:

Тестовый период ([https://ruvds.com/vps\\_test/](https://ruvds.com/vps_test/))  
Дешевый VPS ([https://ruvds.com/cheap\\_vps/](https://ruvds.com/cheap_vps/))  
VPS Старт ([https://ruvds.com/vps\\_start/](https://ruvds.com/vps_start/))  
VPS Мощные ([https://ruvds.com/turbo\\_vps/](https://ruvds.com/turbo_vps/))  
VPS Windows (<https://ruvds.com/ru/windows/>)  
VPS Быстрые NVMe (<https://ruvds.com/drive/>)  
VPS с 1C ([https://ruvds.com/1c\\_vps/](https://ruvds.com/1c_vps/))  
Форекс VPS ([https://ruvds.com/vps\\_forex/](https://ruvds.com/vps_forex/))  
Игровые серверы ([https://ruvds.com/vps\\_games/](https://ruvds.com/vps_games/))  
VPS для бизнеса ([https://ruvds.com/vps\\_business/](https://ruvds.com/vps_business/))  
Пинг до дата-центров ([https://ruvds.com/vps\\_ping/](https://ruvds.com/vps_ping/))

### КЛИЕНТАМ:

О компании (<https://ruvds.com/ru-rub/about/>)  
Дата-Центры (<https://ruvds.com/ru-rub/data/>)  
Новости (<https://ruvds.com/ru/category/news-and-events/>)  
Аттестация по ФСТЭК (<https://ruvds.com/ru/fstek/>)  
Бонусная программа (<https://ruvds.com/ru/bonus/>)

### УСЛУГИ:

Антивирусная защита (<https://ruvds.com/ru-rub/kaspersky/>)  
Аренда лицензий (<https://ruvds.com/ru-rub/licence/>)  
Облачное хранилище ([https://ruvds.com/ru-rub/huge\\_disk/](https://ruvds.com/ru-rub/huge_disk/))  
VPS серверы с Plesk Obsidian ([https://ruvds.com/plesk\\_vps/](https://ruvds.com/plesk_vps/))  
Колокация (<https://ruvds.com/colocation/>)  
Looking Glass (<https://lg.ruvds.com/>)  
DNS (<https://dns.ruvds.com/>)

### ПАРТНЕРАМ:

Партнерская программа (<https://ruvds.com/ru-rub/partner/>)  
API ([https://ruvds.com/ru-rub/use\\_api/](https://ruvds.com/ru-rub/use_api/))



Публичная Оферта (<https://ruvds.com/ru-rub/contract>)

Политика обработки персональных данных ([https://ruvds.com/ru-rub/personal\\_data](https://ruvds.com/ru-rub/personal_data))

Сувениры от RuVDS (<http://ruvds.printdirect.ru>)

ПОМОЩЬ:

Справочник (<https://ruvds.com/ru/helpcenter>)

FAQ (<https://ruvds.com/ru/faq/common-questions/>)

Созданных серверов

543511




МЫ В СОЦИАЛЬНЫХ СЕТЯХ

  
[1abr.ru/company/ruvds/](https://t.me/ruvds_community) ([https://t.me/ruvds\\_community](https://t.me/ruvds_community))

  
([https://vk.com/ru\\_vds](https://vk.com/ru_vds))

  
([https://twitter.com/ru\\_vds](https://twitter.com/ru_vds))

  
([https://www.youtube.com/ru\\_vds](https://www.youtube.com/ru_vds))



support@RUVDS.com (<mailto:support@ruvds.com>)

8 (800) 775-97-42

+7 (495) 135-10-99



(<https://play.google.com/store/apps/details?id=com.rudvs.ruvdsclient>)

(<https://apps.apple.com/ru/app/ruvds-client/id1492272255>)



Copyright © 2023 RuVDS. Все права защищены.

