

[Ссылка](#) / Daemon CLI (dockerd)

# dockerd

## демон

Оставить отзыв

```
Usage: dockerd [OPTIONS]

A self-sufficient runtime for containers.

Options:
  --add-runtime runtime          Register an additional OCI compatible runtime (default "run
  --allow-nondistributable-artifacts list Allow push of nondistributable artifacts to regi
  --api-cors-header string       Set CORS headers in the Engine API
  --authorization-plugin list    Authorization plugins to load
  --bip string                   Specify network bridge IP
  -b, --bridge string            Attach containers to a network bridge
  --cdi-spec-dir list            CDI specification directories to use
  --cgroup-parent string         Set parent cgroup for all containers
  --config-file string           Daemon configuration file (default "/etc/docker/daemon.
  --containerd string            containerd grpc address
  --containerd-namespace string Containerd namespace to use (default "moby")
  --containerd-plugins-namespace string Containerd namespace to use for plugins (default
  --cpu-rt-period int            Limit the CPU real-time period in microseconds for
  --cpu-rt-runtime int           Limit the CPU real-time runtime in microseconds for
  --cri-containerd               start containerd with cri
  --data-root string             Root directory of persistent Docker state (default
  -D, --debug                    Enable debug mode
  --default-address-pool pool-options Default address pools for node specific local ne
  --default-cgroupns-mode string Default mode for containers cgroup namespace ("h
  --default-gateway ip          Container default gateway IPv4 address
  --default-gateway-v6 ip       Container default gateway IPv6 address
  --default-ipc-mode string      Default mode for containers ipc ("shareable" | "p
  --default-network-opt mapmap  Default network options (default map[])
  --default-runtime string       Default OCI runtime for containers (default "run
```

--default-shm-size bytes	Default shm size for containers (default 64MiB)
--default-ulimit ulimit	Default ulimits for containers (default [])
--dns list	DNS server to use
--dns-opt list	DNS options to use
--dns-search list	DNS search domains to use
--exec-opt list	Runtime execution options
--exec-root string	Root directory for execution state files (default /var/lib/docker)
--experimental	Enable experimental features
--fixed-cidr string	IPv4 subnet for fixed IPs
--fixed-cidr-v6 string	IPv6 subnet for fixed IPs
-G, --group string	Group for the unix socket (default "docker")
--help	Print usage
-H, --host list	Daemon socket(s) to connect to
--host-gateway-ip ip	IP address that the special 'host-gateway' string defaults to (default 172.17.0.1)
--http-proxy string	HTTP proxy URL to use for outgoing traffic
--https-proxy string	HTTPS proxy URL to use for outgoing traffic
--icc	Enable inter-container communication (default true)
--init	Run an init in the container to forward signals
--init-path string	Path to the docker-init binary
--insecure-registry list	Enable insecure registry communication
--ip ip	Default IP when binding container ports (default 0.0.0.0)
--ip-forward	Enable net.ipv4.ip_forward (default true)
--ip-masq	Enable IP masquerading (default true)
--ip6tables	Enable addition of ip6tables rules (experimental)
--iptables	Enable addition of iptables rules (default true)
--ipv6	Enable IPv6 networking
--label list	Set key=value labels to the daemon
--live-restore	Enable live restore of docker when containers are restarted
--log-driver string	Default driver for container logs (default "json-file")
-l, --log-level string	Set the logging level ("debug" "info" "warn" "error")
--log-opt map	Default log driver options for containers (default {})
--max-concurrent-downloads int	Set the max concurrent downloads (default 3)
--max-concurrent-uploads int	Set the max concurrent uploads (default 5)
--max-download-attempts int	Set the max download attempts for each pull (default 5)
--metrics-addr string	Set default address and port to serve the metrics
--mtu int	Set the containers network MTU (default 1500)
--network-control-plane-mtu int	Network Control plane MTU (default 1500)
--no-new-privileges	Set no-new-privileges by default for new containers
--no-proxy string	Comma-separated list of hosts or IP addresses for which the proxy is not used
--node-generic-resource list	Advertise user-defined resource
--oom-score-adjust int	Set the oom_score_adj for the daemon

<code>-p, --pidfile string</code>	Path to use for daemon PID file (default <code>"/var/run/docker.pid"</code> )
<code>--raw-logs</code>	Full timestamps without ANSI coloring
<code>--registry-mirror list</code>	Preferred registry mirror
<code>--rootless</code>	Enable rootless mode; typically used with RootlessKit
<code>--seccomp-profile string</code>	Path to seccomp profile. Use <code>"unconfined"</code> to disable seccomp
<code>--selinux-enabled</code>	Enable selinux support
<code>--shutdown-timeout int</code>	Set the default shutdown timeout (default 15)
<code>-s, --storage-driver string</code>	Storage driver to use
<code>--storage-opt list</code>	Storage driver options
<code>--swarm-default-advertise-addr string</code>	Set default address or interface for swarm advertise
<code>--tls</code>	Use TLS; implied by <code>--tlsverify</code>
<code>--tlscacert string</code>	Trust certs signed only by this CA (default <code>"~/.docker/ca.pem"</code> )
<code>--tlscert string</code>	Path to TLS certificate file (default <code>"~/.docker/cert.pem"</code> )
<code>--tlskey string</code>	Path to TLS key file (default <code>"~/.docker/key.pem"</code> )
<code>--tlsverify</code>	Use TLS and verify the remote
<code>--userland-proxy</code>	Use userland proxy for loopback traffic (default <code>auto</code> )
<code>--userland-proxy-path string</code>	Path to the userland proxy binary
<code>--userns-remap string</code>	User/Group setting for user namespaces
<code>--validate</code>	Validate daemon configuration and exit
<code>-v, --version</code>	Print version information and quit

Оставить отзыв

Параметры с помощью `[]` могут указываться несколько раз.

## Описание

`dockerd` это постоянный процесс, который управляет контейнерами. Docker использует разные двоичные файлы для демона и клиента. Для запуска демона вы вводите `dockerd`.

Чтобы запустить демон с отладочным выводом, используйте `dockerd --debug` или добавьте `"debug": true` в `daemon.json` файл.

### Включение экспериментальных функций

Включите экспериментальные функции, начав с `dockerd` флага `--experimental` или добавив `"experimental": true` в `daemon.json` файл.

## Конфигурация окружения

Следующий список переменных окружения поддерживается `dockerd` демоном. Некоторые из этих переменных окружения поддерживаются как демоном Docker, так и `docker` CLI. Обратитесь к [Переменным среды](#) в разделе CLI, чтобы узнать о переменных среды, поддерживаемых `docker` CLI.

Переменная	Описание
<code>DOCKER_CERT_PATH</code>	Расположение ваших ключей аутентификации. Эта переменная используется как <code>docker CLI</code> , так и <code>dockerd</code> демоном.
<code>DOCKER_DRIVER</code>	Используемый драйвер хранилища.
<code>DOCKER_RAMDISK</code>	Если этот параметр установлен, он отключается <code>pivot_root</code> .
<code>DOCKER_TLS_VERIFY</code>	При установке Docker использует TLS и проверяет удаленный сервер. Эта переменная используется как <code>docker CLI</code> , так и <code>dockerd</code> демоном.
<code>DOCKER_TMPDIR</code>	Расположение временных файлов, созданных демоном.
<code>HTTP_PROXY</code>	URL прокси-сервера для HTTP-запросов, если он не переопределен NoProxy. Подробнее см. в <a href="#">спецификации Go</a> .
<code>HTTPS_PROXY</code>	URL прокси-сервера для запросов HTTPS, если он не переопределен NoProxy. Подробнее смотрите в <a href="#">спецификации Go</a> .
<code>MOBY_DISABLE_PIGZ</code>	Отключает использование <code>unpigz</code> для параллельного распаковывания слоев при извлечении изображений, даже если оно установлено.
<code>NO_PROXY</code>	Значения через запятую, указывающие хосты, которые должны быть исключены из прокси. Подробнее см. в <a href="#">спецификации Go</a> .

Оставить отзыв

# Примеры

## Настройка прокси-сервера

### Примечание

Обратитесь к [руководству по Docker Desktop](#), если вы используете [Docker Desktop](#).

Если вы используете HTTP-прокси-сервер, например, в корпоративных настройках, вам, возможно, придется настроить демон Docker для использования прокси-сервера для



таких операций, как извлечение и отправка изображений. Демон можно настроить тремя способами:

1. Использование переменных окружения (`HTTP_PROXY`, `HTTPS_PROXY` и `NO_PROXY`).
2. Используя `http-proxy`, `https-proxy` и `no-proxy` поля в [файле конфигурации daemon](#) (движок Docker версии 23.0 или более поздней).
3. С помощью `--http-proxy`, `--https-proxy` и `--no-proxy` параметров командной строки. (Движок Docker версии 23.0 или более поздней).

Параметры командной строки и файла конфигурации имеют приоритет над переменными среды. Обратитесь к [управление и настройка Docker с помощью systemd](#), чтобы установить эти переменные среды на хосте с помощью `systemd`.

Оставить отзыв

## Опция сокета демона

Демон Docker может прослушивать запросы к [API Docker Engine](#) через сокеты трех различных типов: `unix`, `tcp` и `fd`.

По умолчанию по адресу `unix` создается `/var/run/docker.sock` доменный сокет (или сокет IPC), для которого требуется либо `root` разрешение, либо `docker` членство в группе.

Если вам нужно получить удаленный доступ к демону Docker, вам необходимо включить сокет `tcp`. При использовании сокета TCP демон Docker по умолчанию предоставляет прямой доступ к демону Docker без шифрования и аутентификации. Вам следует обезопасить демон либо с помощью [встроенного HTTPS-зашифрованного сокета](#), либо установив перед ним безопасный веб-прокси. Вы можете прослушивать порт `2375` на всех сетевых интерфейсах с `-H tcp://0.0.0.0:2375` или на определенном сетевом интерфейсе, используя его IP-адрес: `-H tcp://192.168.59.103:2375`. Обычно используется порт `2375` для незашифрованной и порт `2376` для зашифрованной связи с демоном.

### Примечание

Если вы используете зашифрованный сокет HTTPS, имейте в виду, что поддерживается только TLS версии 1.0 и выше. Протоколы SSLv3 и ниже не поддерживаются по соображениям безопасности.

В системах на базе systemd вы можете взаимодействовать с демоном через [активацию сокета systemd](#) <sup>↗</sup>, с помощью `dockerd -H fd://`. Использование `fd://` работает для большинства настроек, но вы также можете указать отдельные сокеты:

`dockerd -H fd:///3`. Если указанные файлы, активированные сокетами, не найдены, демон завершает работу. Примеры использования активации сокета systemd с помощью Docker и systemd можно найти в [дереве исходных текстов Docker](#) <sup>↗</sup>.

Вы можете настроить демон Docker для одновременного прослушивания нескольких сокетов, используя несколько `-H` опций:

Приведенный ниже пример запускает демон, прослушивающий сокет Unix по умолчанию и 2 определенных IP-адреса на этом хосте:

```
$ sudo dockerd -H unix:///var/run/docker.sock -H tcp://192.168.59.106 -H tcp://10.10.10.2
```

Оставить отзыв

Клиент Docker использует `DOCKER_HOST` переменную окружения, чтобы установить флаг `-H` для клиента. Используйте одну из следующих команд:

```
$ docker -H tcp://0.0.0.0:2375 ps
```

```
$ export DOCKER_HOST="tcp://0.0.0.0:2375"
```

```
$ docker ps
```

Установка для `DOCKER_TLS_VERIFY` переменной окружения любого значения, отличного от пустой строки эквивалентна установке `--tlsverify` флага. Следующие действия эквивалентны:

```
$ docker --tlsverify ps
# or
$ export DOCKER_TLS_VERIFY=1
$ docker ps
```

Клиент Docker учитывает `HTTP_PROXY`, `HTTPS_PROXY` и `NO_PROXY` переменные среды (или их версии в нижнем регистре). `HTTPS_PROXY` имеет приоритет над `HTTP_PROXY`.

Клиент Docker поддерживает подключение к удаленному демону по SSH:

```
$ docker -H ssh://me@example.com:22/var/run/docker.sock ps
$ docker -H ssh://me@example.com:22 ps
$ docker -H ssh://me@example.com ps
$ docker -H ssh://example.com ps
```

Чтобы использовать SSH-соединение, вам необходимо настроить `ssh` таким образом, чтобы оно могло достигать удаленного хоста с аутентификацией по открытому ключу. Аутентификация по паролю не поддерживается. Если ваш ключ защищен парольной фразой, вам необходимо настроить `ssh-agent`.

## Привязать Docker к другому хосту / порту или сокету Unix

### Предупреждение

Изменение по умолчанию `docker` привязки демона к TCP-порту или группе пользователей Unix `docker` создает риски для безопасности, поскольку это может позволить пользователям, не имеющим прав `root`, получить `root`-доступ к хосту. Убедитесь, что вы контролируете доступ к `docker`. Если вы привязываетесь к TCP-порту, любой, у кого есть доступ к этому порту, имеет полный доступ к Docker; поэтому в открытой сети это нежелательно.

С помощью `-H` можно заставить демон Docker прослушивать определенный IP и порт. По умолчанию он прослушивает `unix:///var/run/docker.sock`, чтобы разрешить только локальные подключения пользователя `root`. Вы могли бы установить для него значение `0.0.0.0:2375` или определенный IP-адрес хоста, чтобы предоставить доступ всем, но это не рекомендуется, потому что кто-то может получить `root`-доступ к хосту, на котором запущен демон.

Аналогично, клиент Docker может использовать `-H` для подключения к пользовательскому порту. Клиент Docker по умолчанию использует подключение к `unix:///var/run/docker.sock` в Linux и `tcp://127.0.0.1:2376` в Windows.

`-H` принимает назначение хоста и порта в следующем формате:

```
tcp://[host]:[port][path] or unix://path
```

Пример:

- `tcp://` -> TCP-подключение к `127.0.0.1` любому порту `2376`, когда включено шифрование TLS, или к порту `2375`, когда обмен данными осуществляется обычным текстом.
- `tcp://host:2375` -> TCP-соединение на хосте: 2375
- `tcp://host:2375/path` -> TCP-соединение на хосте: 2375 и добавляйте путь ко всем запросам
- `unix://path/to/socket` -> Сокет Unix, расположенный по адресу `path/to/socket`

`-H` когда значение пусто, по умолчанию используется то же значение, что и при передаче по `-H`.

`-H` также принимает краткую форму для привязок TCP: `host:` или `host:port` или `:port`

Оставить отзыв

Запустите Docker в режиме демона:

```
$ sudo <path to>/dockerd -H 0.0.0.0:5555 &
```

Загрузить `ubuntu` изображение:

```
$ docker -H :5555 pull ubuntu
```

Вы можете использовать несколько `-H`, например, если вы хотите слушать с обеих TCP и UNIX-сокета


```
$ sudo dockerd -H tcp://127.0.0.1:2375 -H unix:///var/run/docker.sock &  
# Download an ubuntu image, use default Unix socket  
$ docker pull ubuntu  
# OR use the TCP port  
$ docker -H tcp://127.0.0.1:2375 pull ubuntu
```

## Daemon storage-драйвер

В Linux демон Docker поддерживает несколько различных драйверов хранилища уровня изображений: `overlay2`, `fuse-overlayfs`, `btrfs`, и `zfs`.



`overlay2` является предпочтительным драйвером хранилища для всех поддерживаемых в настоящее время дистрибутивов Linux и выбирается по умолчанию. Если у пользователей нет веских причин предпочесть другой драйвер хранилища, `overlay2` следует использовать.

Вы можете узнать больше о драйверах хранилища и о том, как выбрать один из них, в [Выберите драйвер хранилища](#) .

В Windows демон Docker поддерживает только `windowsfilter` драйвер хранилища.

## Параметры для каждого драйвера хранилища

Конкретный драйвер хранилища может быть настроен с параметрами, указанными с помощью `--storage-opt` флагов. Параметры для `zfs` начать с `zfs` и параметры для `btrfs` начать с `btrfs`.

Оставить отзыв

### Параметры ZFS

`zfs.fsname`

Указывает файловую систему ZFS, которую демон должен использовать для создания своих наборов данных. По умолчанию используется файловая система ZFS в `/var/lib/docker`.

Пример

```
$ sudo dockerd -s zfs --storage-opt zfs.fsname=zroot/docker
```

### Параметры Btrfs

`btrfs.min_space`

Указывает минимальный размер, который будет использоваться при создании вложенного тома, используемого для контейнеров. Если пользователь использует дисковую квоту для btrfs при создании или запуске контейнера с параметром `--storage-opt size`, Docker должен убедиться, что размер не может быть меньше, чем `btrfs.min_space`.

Пример

```
$ sudo dockerd -s btrfs --storage-opt btrfs.min_space=10G
```

## Параметры наложения 2

### наложение2.размер

Устанавливает максимальный размер контейнера по умолчанию. Он поддерживается только в том случае, если используется резервная файловая система `xfs` и смонтирована с `quota` опцией `mount`. В этих условиях пользователь может передать любой размер, меньший размера резервной файловой системы.

#### Пример

```
$ sudo dockerd -s overlay2 --storage-opt overlay2.size=1G
```

Оставить отзыв

## Параметры Windowsfilter

### размер

Определяет размер, который будет использоваться при создании изолированной среды, используемой для контейнеров. По умолчанию используется 20G.

#### Пример

```
C:\> dockerd --storage-opt size=40G
```

## Параметры среды выполнения

Демон Docker использует [OCI](#) <sup>↗</sup>-совместимую среду выполнения (вызываемую через `containerd` демон) в качестве интерфейса к ядру Linux (`namespaces`, `cgroups` и `SELinux`).

## Настройка времени выполнения контейнера

По умолчанию демон Docker использует `glibc` в качестве среды выполнения контейнера. Вы можете настроить демон для добавления дополнительных сред выполнения.



оболочки containerd, установленные на `PATH`, можно использовать напрямую, без необходимости редактировать конфигурацию демона. Например, если вы устанавливаете оболочку контейнеров Kata (`containerd-shim-kata-v2`) на `PATH`, вы можете выбрать эту среду выполнения с помощью `docker run` без необходимости редактировать конфигурацию демона:

```
$ docker run --runtime io.containerd.kata.v2
```

Среды выполнения контейнеров, в которых не реализованы оболочки containerd или containerd shims, установленные вне `PATH`, должны быть зарегистрированы в демоне либо через файл конфигурации, либо с помощью `--add-runtime` флага командной строки.

Примеры использования других сред выполнения контейнеров см. в разделе

[Альтернативные среды выполнения контейнеров](#) 

## Настройте время выполнения с помощью daemon.json

Чтобы зарегистрировать и настроить среды выполнения контейнера с помощью файла конфигурации демона, добавьте среды выполнения в качестве записей в разделе

`runtimes`:

```
{
  "runtimes": {
    "<runtime>": {}
  }
}
```

Ключ записи (`<runtime>` в предыдущем примере) представляет имя среды выполнения. Это имя, на которое вы ссылаетесь при запуске контейнера, используя

```
docker run --runtime <runtime>.
```

Запись среды выполнения содержит объект, определяющий конфигурацию для вашей среды выполнения. Свойства объекта зависят от того, какую среду выполнения вы хотите зарегистрировать:

Если среда выполнения реализует свою собственную оболочку containerd, объект должен содержать `runtimeType` поле и необязательное `options` поле.

```
{
  "runtimes": {
    "<runtime>": {
      "runtimeType": "<name-or-path>",
      "options": {}
    }
  }
}
```

Смотрите [Настройте прокладки](#).

- Если среда выполнения предназначена для замены `glibc`, объект содержит `path` поле и необязательное `runtimeArgs` поле.

```
{
  "runtimes": {
    "<runtime>": {
      "path": "/path/to/bin",
      "runtimeArgs": ["...args"]
    }
  }
}
```

Оставить отзыв

Смотрите [Настройте раскрывающиеся замены glibc](#).

После изменения конфигурации среды выполнения в файле конфигурации необходимо перезагрузить демон, чтобы изменения вступили в силу:

```
$ sudo systemctl reload dockerd
```

## Настройка прокладок containerd

Если среда выполнения, которую вы хотите зарегистрировать, реализует оболочку `containerd`, или если вы хотите зарегистрировать среду выполнения, которая использует оболочку `glibc`, используйте следующий формат для записи среды выполнения:

```
{
  "runtimes": {
    "<runtime>": {
      "runtimeType": "<name-or-path>",
```

```
    "options": {}
  }
}
```

`runtimeType` относится либо к:

- Полное имя прокладки контейнера.

Полное имя оболочки совпадает с `runtime_type`, используемым для регистрации среды выполнения в конфигурации CRI containerd. Например,

`io.containerd.runsc.v1`.

- Путь к двоичному файлу-оболочке containerd.

Этот параметр полезен, если вы установили двоичный файл оболочки containerd за пределами `PATH`.

`options` является необязательным. Он позволяет вам указать конфигурацию среды выполнения, которую вы хотите использовать для прокладки. Параметры конфигурации, которые вы можете указать в `options` зависят от среды выполнения, которую вы регистрируете. Для большинства прокладок поддерживаются следующие параметры конфигурации: `TypeUrl` и `ConfigPath`. Например:

```
{
  "runtimes": {
    "gvisor": {
      "runtimeType": "io.containerd.runsc.v1",
      "options": {
        "TypeUrl": "io.containerd.runsc.v1.options",
        "ConfigPath": "/etc/containerd/runsc.toml",
      }
    }
  }
}
```

Вы можете настроить несколько сред выполнения, используя один и тот же тип выполнения. Например:

```
{
  "runtimes": {
    "gvisor-foo": {
```

```
    "runtimeType": "io.containerd.runsc.v1",
    "options": {
      "TypeUrl": "io.containerd.runsc.v1.options",
      "ConfigPath": "/etc/containerd/runsc-foo.toml"
    }
  },
  "gvisor-bar": {
    "runtimeType": "io.containerd.runsc.v1",
    "options": {
      "TypeUrl": "io.containerd.runsc.v1.options",
      "ConfigPath": "/etc/containerd/runsc-bar.toml"
    }
  }
}
```

Оставить отзыв

Поле `options` принимает специальный набор параметров конфигурации при использовании с `"runtimeType": "io.containerd.runc.v2"`. Для получения дополнительной информации о параметрах runc обратитесь к разделу конфигурации runc в [Руководство по настройке плагина CRI](#).

## Настройка раскрывающихся замен runc

Если среда выполнения, которую вы хотите зарегистрировать, может выступать в качестве замены runc, вы можете зарегистрировать среду выполнения либо с помощью файла конфигурации daemon, либо с помощью `--add-runtime` флага для `dockerd` cli.

При использовании файла конфигурации запись используется в следующем формате:

```
{
  "runtimes": {
    "<runtime>": {
      "path": "/path/to/binary",
      "runtimeArgs": ["...args"]
    }
  }
}
```

Где `path` is either the absolute path to the runtime executable, or the name of the executable installed on `PATH`:

```
{
  "runtimes": {
    "runc": {
      "path": "runc"
    }
  }
}
```

And `runtimeArgs` lets you optionally pass additional arguments to the runtime. Entries with this format use the containerd runc shim to invoke a custom runtime binary.

When you use the `--add-runtime` CLI flag, use the following format:

```
$ sudo dockerd --add-runtime <runtime>=<path>
```

Определение аргументов среды выполнения через командную строку не поддерживается.

Пример конфигурации для замены раскрывающегося списка runc см. в разделе

[Альтернативные среды выполнения контейнера > youki](#) 

## Настройте среду выполнения контейнера по умолчанию

Вы можете указать либо имя полностью настроенной оболочки среды выполнения containerd, либо имя зарегистрированной среды выполнения. Вы можете указать среду выполнения по умолчанию либо с помощью файла конфигурации daemon, либо с помощью

`--default-runtime` флага для `dockerd` cli.

При использовании файла конфигурации запись используется в следующем формате:

```
{
  "default-runtime": "io.containerd.runsc.v1"
}
```

При использовании `--default-runtime` флага CLI используйте следующий формат:

```
$ dockerd --default-runtime io.containerd.runsc.v1
```

## Run containerd standalone

By default, the Docker daemon automatically starts `containerd`. If you want to control `containerd` startup, manually start `containerd` and pass the path to the `containerd` socket using the `--containerd` flag. For example:

```
$ sudo dockerd --containerd /run/containerd/containerd.sock
```

## Configure cgroup driver

You can configure how the runtime should manage container cgroups, using the `--exec-opt native.cgroupdriver` CLI flag.

Вы можете указать только `cgroupfs` или `systemd`. Если вы укажете `systemd`, а оно недоступно, система выдаст ошибку. Если вы опустите этот `native.cgroupdriver` параметр, `cgroupfs` используется на хостах cgroup v1, `systemd` используется на хостах cgroup v2 с доступным `systemd`.

В этом примере значение `cgroupdriver` равно `systemd`:

```
$ sudo dockerd --exec-opt native.cgroupdriver=systemd
```

Установка этого параметра применяется ко всем контейнерам, которые запускает демон.

## Настройка технологии изоляции контейнеров (Windows)

Для контейнеров Windows вы можете указать технологию изоляции контейнеров по умолчанию для использования, используя флаг `--exec-opt isolation`.

В следующем примере `hyperv` используется технология изоляции по умолчанию:

```
> dockerd --exec-opt isolation=hyperv
```

Если значение изоляции не указано при запуске демона в клиенте Windows, значение по умолчанию равно `hyperv`, а в Windows server по умолчанию равно `process`.

## Параметры DNS демона

Set the DNS server for all Docker containers, use:



```
$ sudo dockerd --dns 8.8.8.8
```

To set the DNS search domain for all Docker containers, use:

```
$ sudo dockerd --dns-search example.com
```

## Разрешить отправку нераспределяемых артефактов

Некоторые образы (например, базовые образы Windows) содержат артефакты, распространение которых ограничено лицензией. Когда эти изображения помещаются в реестр, артефакты с ограниченным доступом не включаются.

Чтобы переопределить это поведение для определенных реестров, используйте

`--allow-nondistributable-artifacts` параметр в одной из следующих форм:

- `--allow-nondistributable-artifacts myregistry:5000` сообщает демону Docker отправить нераспределяемые артефакты в myregistry: 5000.
- `--allow-nondistributable-artifacts 10.1.0.0/16` указывает демону Docker отправлять нераспределяемые артефакты во все реестры, разрешенный IP-адрес которых находится в подсети, описываемой синтаксисом CIDR.

Эту опцию можно использовать несколько раз.

Этот параметр полезен при загрузке изображений, содержащих не подлежащие распространению артефакты, в реестр в сети с воздушным зазором, чтобы хосты в этой сети могли извлекать изображения без подключения к другому серверу.

### Предупреждение

Нераспространяемые артефакты обычно имеют ограничения на то, как и где они могут быть распространены. Используйте эту функцию только для отправки артефактов в частные реестры и убедитесь, что вы соблюдаете все условия, которые распространяются на не подлежащие распространению артефакты.

Оставить отзыв



Secure registries

In this section, "registry" refers to a private registry, and `myregistry:5000` is a placeholder example of a private registry.

Docker considers a private registry either secure or insecure. A secure registry uses TLS and a copy of its CA certificate is placed on the Docker host at `/etc/docker/certs.d/myregistry:5000/ca.crt`. An insecure registry is either not using TLS (i.e., listening on plain text HTTP), or is using TLS with a CA certificate not known by the Docker daemon. The latter can happen when the certificate wasn't found under `/etc/docker/certs.d/myregistry:5000/`, or if the certificate verification failed (i.e., wrong CA).

By default, Docker assumes all registries to be secure, except for local registries. Communicating with an insecure registry isn't possible if Docker assumes that registry is secure. In order to communicate with an insecure registry, the Docker daemon requires `--insecure-registry` in one of the following two forms:

- `--insecure-registry myregistry:5000` tells the Docker daemon that `myregistry:5000` should be considered insecure.
- `--insecure-registry 10.1.0.0/16` сообщает демону Docker, что все реестры, домен которых, разрешенный на IP-адрес, является частью подсети, описываемой синтаксисом CIDR, следует считать небезопасными.

Этот флаг можно использовать несколько раз, чтобы несколько реестров могли быть помечены как небезопасные.

Если небезопасный реестр не помечен как небезопасный, `docker pull`, `docker push` и `docker search` появляются сообщения об ошибках, предлагающие пользователю либо защитить, либо передать `--insecure-registry` флаг демону Docker, как описано выше.

Локальные реестры, IP-адрес которых попадает в диапазон `127.0.0.0 / 8`, автоматически помечаются как небезопасные, начиная с версии Docker 1.3.2. Не рекомендуется полагаться на это, поскольку в будущем это может измениться.

Включение `--insecure-registry`, т.е. разрешение незашифрованной и / или ненадежной связи, может быть полезно при запуске локального реестра. Однако, поскольку его использование создает уязвимости в системе безопасности, его следует включать только в целях тестирования. Для повышения безопасности пользователям следует

добавлять свои центры сертификации в список доверенных центров сертификации своей системы вместо включения `--insecure-registry`.


## Устаревшие реестры

Операции с реестрами, поддерживающими только устаревший протокол v1, больше не поддерживаются. В частности, демон не пытается выполнять нажатие, извлечение или вход в реестры v1. Исключением из этого является `search` который все еще может выполняться в реестрах версии v1.

## Запуск демона Docker за HTTPS\_PROXY

When running inside a LAN that uses an `HTTPS` proxy, the proxy's certificates replace Docker Hub's certificates. These certificates must be added to your Docker host's configuration:

1. Install the `ca-certificates` package for your distribution
2. Ask your network admin for the proxy's CA certificate and append them to `/etc/pki/tls/certs/ca-bundle.crt`
3. Then start your Docker daemon with `HTTPS_PROXY=http://username:password@proxy:port/ dockerd`. The `username:` and `password@` are optional - and are only needed if your proxy is set up to require authentication.

Это только добавляет прокси-сервер и аутентификацию к запросам демона Docker. Чтобы использовать прокси-сервер при создании образов и запуске контейнеров, см. раздел [Настройка Docker на использование прокси-сервера](#) 

## Настройки ulimit по умолчанию

Флаг `--default-ulimit` позволяет установить параметры по умолчанию `ulimit` для использования во всех контейнерах. Он использует те же параметры, что и `--ulimit` для `docker run`. Если эти значения по умолчанию не установлены, `ulimit` настройки наследуются от демона Docker. Любые `--ulimit` параметры, переданные для `docker run` переопределяют значения демона по умолчанию.

Будьте осторожны при настройке `nproc` с `ulimit` флагом, поскольку `nproc` он работает в Linux для установки максимального количества процессов, доступных

пользователю, а не контейнеру. Подробнее см. в [docker run](#) [справке](#).

## Авторизация доступа

Авторизация доступа Docker может быть расширена с помощью плагинов авторизации, которые ваша организация может приобрести или создать самостоятельно. Вы можете установить один или несколько плагинов авторизации при запуске Docker, `daemon` используя `--authorization-plugin=PLUGIN_ID` опцию.

```
$ sudo dockerd --authorization-plugin=plugin1 --authorization-plugin=plugin2,...
```

`PLUGIN_ID` Значением является либо имя плагина, либо путь к его спецификации файл. Реализация плагина определяет, можете ли вы указать имя или путь. Обратитесь к администратору Docker, чтобы получить информацию о Плагины доступны для вас.

Оставить отзыв

После установки плагина запросы, сделанные к `daemon` через командную строку или API движка Docker, разрешаются или запрещаются плагином. Если у вас установлено несколько плагинов, каждый плагин по порядку должен разрешать выполнение запроса на него.

Для получения информации о том, как создать плагин авторизации, обратитесь к разделу [плагин авторизации](#).

## Параметры пространства имен пользователя Daemon

[Поддержка пользовательского пространства имен](#) в ядре Linux обеспечивает дополнительную безопасность, позволяя процессу и, следовательно, контейнеру иметь уникальный диапазон идентификаторов пользователей и групп, которые выходят за рамки традиционного диапазона пользователей и групп, используемого хост-системой. Одно из наиболее важных улучшений безопасности заключается в том, что по умолчанию процессы контейнера, запускаемые от имени `root` пользователя, имеют ожидаемые административные привилегии (с некоторыми ограничениями) внутри контейнера, но фактически сопоставляются с непривилегированными `uid` на хосте.

Подробнее об использовании этой функции, а также об ограничениях см. в разделе [Изолировать контейнеры с пользовательским пространством имен](#).

## Configure host gateway IP

Демон Docker поддерживает специальное `host-gateway` значение для `--add-host` флага для команд `docker run` и `docker build`. Это значение соответствует IP-адресу шлюза хоста и позволяет контейнерам подключаться к службам, запущенным на хосте.

По умолчанию `host-gateway` преобразуется в IP-адрес моста по умолчанию. Вы можете настроить это для разрешения на другой IP-адрес, используя `--host-gateway-ip` флаг интерфейса командной строки `dockerd` или `host-gateway-ip` ключ в файле конфигурации `daemon`.

```
$ cat > /etc/docker/daemon.json
{ "host-gateway-ip": "192.0.2.0" }
$ sudo systemctl restart docker
$ docker run -it --add-host host.docker.internal:host-gateway \
  busybox ping host.docker.internal
PING host.docker.internal (192.0.2.0): 56 data bytes
```


Оставить отзыв

## Включить устройства CDI

### Примечание

Это экспериментальная функция и как таковая не представляет собой стабильный API.

По умолчанию эта функция не включена. Для этой функции установите `features.cdi` значение `true` в `daemon.json` файле конфигурации.

Container Device Interface (CDI) - это [стандартизированный](#)  механизм для выполнения контейнеров для создания контейнеров, способных взаимодействовать со сторонними устройствами.

Демон Docker поддерживает запуск контейнеров с устройствами CDI, если запрошенные спецификации устройств доступны в файловой системе демона.

Директорами спецификаций по умолчанию являются:

 `/etc/cdi/` для получения статических спецификаций CDI

- `/var/run/cdi` для сгенерированных спецификаций CDI

В качестве альтернативы, вы можете задать пользовательские расположения для спецификаций CDI, используя `cdi-spec-dirs` опцию в `daemon.json` файле конфигурации или `--cdi-spec-dir` флаг для `dockerd` командной строки.

```
{
  "features": {
    "cdi": true
  },
  "cdi-spec-dirs": ["/etc/cdi/", "/var/run/cdi"]
}
```

Оставить отзыв

Когда CDI включен для демона, вы можете просмотреть настроенную спецификацию CDI каталоги с помощью команды `docker info`.

## Разные опции

Маскировка IP-адресов использует преобразование адресов, чтобы позволить контейнерам без общедоступного IP-адреса взаимодействовать с другими компьютерами в Интернете. Это может создавать помехи для некоторых сетевых топологий и может быть отключено с помощью `--ip-masq=false`.

Docker supports soft links for the Docker data directory (`/var/lib/docker`) and for `/var/lib/docker/tmp`. The `DOCKER_TMPDIR` and the data directory can be set like this:

```
$ export DOCKER_TMPDIR=/mnt/disk2/tmp
$ sudo -E dockerd --data-root /var/lib/docker -H unix://
```

## Default cgroup parent

Опция `--cgroup-parent` позволяет установить родительскую группу по умолчанию для контейнеров. Если этот параметр не установлен, по умолчанию он равен `/docker` для драйвера cgroupfs и `system.slice` для драйвера systemd cgroup.

Если cgroup имеет начальную косую черту (`/`), cgroup создается под корневой cgroup, в противном случае cgroup создается под cgroup демона.

Предполагая, что демон запущен в сgroup `daemoncgroup`, `--cgroup-parent=/foobar` создает сgroup в `/sys/fs/cgroup/memory/foobar`, тогда как использование `--cgroup-parent=foobar` создает сgroup в `/sys/fs/cgroup/memory/daemoncgroup/foobar`

Драйвер systemd сgroup имеет другие правила для `--cgroup-parent`. systemd представляет иерархию по фрагментам, а имя фрагмента кодирует местоположение в дереве. Итак, `--cgroup-parent` для системных групп должно быть имя фрагмента. Имя может состоять из серии имен, разделенных тире, которые описывают путь к фрагменту из корневого фрагмента. Например, `--cgroup-parent=user-a-b.slice` означает, что сgroup памяти для контейнера создается в `/sys/fs/cgroup/memory/user.slice/user-a.slice/user-a-b.slice/docker-<id>.scope`.

Этот параметр также может быть установлен для каждого контейнера, используя `--cgroup-parent` параметр в `docker create` и `docker run`, и имеет приоритет над `--cgroup-parent` параметром в демоне.

Оставить отзыв

## Метрики демонов

Опция `--metrics-addr` использует TCP-адрес для обслуживания metrics API. Эта функция все еще экспериментальная, поэтому демон должен быть запущен в экспериментальном режиме, чтобы эта функция заработала.

Для обслуживания API метрик на `localhost:9323` вы должны указать `--metrics-addr 127.0.0.1:9323`, что позволяет вам отправлять запросы в API на `127.0.0.1:9323/metrics` для получения метрик в формате [prometheus](#).

Порт `9323` - это [порт по умолчанию, связанный с метриками Docker](#) во избежание конфликтов с другими экспортерами и сервисами Prometheus.

Если вы используете сервер Prometheus, вы можете добавить этот адрес в свои конфигурации scrape, чтобы Prometheus собирал показатели в Docker. Для получения дополнительной информации см. в разделе [Сбор показателей Docker с помощью Prometheus](#).

## Общие ресурсы узла

Опция `--node-generic-resources` использует список пар ключ-значение (`key=value`), который позволяет вам рекламировать определенные пользователем ресурсы в кластере.

Текущий ожидаемый вариант использования - рекламировать графические процессоры NVIDIA, чтобы запрашивающие службы `NVIDIA-GPU=[0-16]` могли попадать на узел, на котором достаточно графических процессоров для выполнения задачи.

Пример использования:

```
{
  "node-generic-resources": [
    "NVIDIA-GPU=UUID1",
    "NVIDIA-GPU=UUID2"
  ]
}
```

Оставить отзыв

## Файл конфигурации демона

Опция `--config-file` позволяет вам задать любой параметр конфигурации для демона в формате JSON. В этом файле используются те же имена флагов, что и в ключах, за исключением флагов, разрешающих несколько записей, где используется множественное число от имени флага, например, `labels` для `label` флага.

Параметры, установленные в файле конфигурации, не должны конфликтовать с параметрами, установленными с помощью флагов. Демон Docker не запускается, если параметр дублируется между файлом и флагами, независимо от их значения. Это сделано намеренно и позволяет избежать автоматического игнорирования изменений, вносимых при перезагрузке конфигурации. Например, демон не запускается, если вы задаете метки демонов в файле конфигурации, а также устанавливаете метки демонов с помощью флага `--label`. Параметры, отсутствующие в файле, игнорируются при запуске демона.

Опция `--validate` позволяет проверять файл конфигурации без запуска демона Docker. Для недействительных файлов конфигурации возвращается ненулевой код выхода.

```
$ dockerd --validate --config-file=/tmp/valid-config.json
configuration OK
```

```
$ echo $?
0
```

```
$ dockerd --validate --config-file /tmp/invalid-config.json
unable to configure the Docker daemon with file /tmp/invalid-config.json: the following direct
```



```
$ echo $?  
1
```

## В Linux

Расположение файла конфигурации по умолчанию в Linux - `/etc/docker/daemon.json`.  
Используйте флаг `--config-file`, чтобы указать расположение, отличное от расположения по умолчанию.

Ниже приведен полный пример разрешенных параметров конфигурации в Linux:

```
{  
  "allow-nondistributable-artifacts": [],  
  "api-cors-header": "",  
  "authorization-plugins": [],  
  "bip": "",  
  "bridge": "",  
  "builder": {  
    "gc": {  
      "enabled": true,  
      "defaultKeepStorage": "10GB",  
      "policy": [  
        { "keepStorage": "10GB", "filter": ["unused-for=2200h"] },  
        { "keepStorage": "50GB", "filter": ["unused-for=3300h"] },  
        { "keepStorage": "100GB", "all": true }  
      ]  
    }  
  },  
  "cgroup-parent": "",  
  "containerd": "/run/containerd/containerd.sock",  
  "containerd-namespace": "docker",  
  "containerd-plugin-namespace": "docker-plugins",  
  "data-root": "",  
  "debug": true,  
  "default-address-pools": [  
    {  
      "base": "172.30.0.0/16",  
      "size": 24  
    }  
  ],  
  {
```

```
    "base": "172.31.0.0/16",
    "size": 24
  }
],
"default-cgroupns-mode": "private",
"default-gateway": "",
"default-gateway-v6": "",
"default-network-opts": {},
"default-runtime": "runc",
"default-shm-size": "64M",
"default-ulimits": {
  "nofile": {
    "Hard": 64000,
    "Name": "nofile",
    "Soft": 64000
  }
},
"dns": [],
"dns-opts": [],
"dns-search": [],
"exec-opts": [],
"exec-root": "",
"experimental": false,
"features": {},
"fixed-cidr": "",
"fixed-cidr-v6": "",
"group": "",
"host-gateway-ip": "",
"hosts": [],
"proxies": {
  "http-proxy": "http://proxy.example.com:80",
  "https-proxy": "https://proxy.example.com:443",
  "no-proxy": "*.test.example.com,example.org",
},
"icc": false,
"init": false,
"init-path": "/usr/libexec/docker-init",
"insecure-registries": [],
"ip": "0.0.0.0",
"ip-forward": false,
"ip-masq": false,
"iptables": false,
```

```
"ip6tables": false,
"ipv6": false,
"labels": [],
"live-restore": true,
"log-driver": "json-file",
"log-level": "",
"log-opts": {
  "cache-disabled": "false",
  "cache-max-file": "5",
  "cache-max-size": "20m",
  "cache-compress": "true",
  "env": "os,customer",
  "labels": "somelabel",
  "max-file": "5",
  "max-size": "10m"
},
"max-concurrent-downloads": 3,
"max-concurrent-uploads": 5,
"max-download-attempts": 5,
"mtu": 0,
"no-new-privileges": false,
"node-generic-resources": [
  "NVIDIA-GPU=UUID1",
  "NVIDIA-GPU=UUID2"
],
"oom-score-adjust": 0,
"pidfile": "",
"raw-logs": false,
"registry-mirrors": [],
"runtimes": {
  "cc-runtime": {
    "path": "/usr/bin/cc-runtime"
  },
  "custom": {
    "path": "/usr/local/bin/my-runc-replacement",
    "runtimeArgs": [
      "--debug"
    ]
  }
},
"seccomp-profile": "",
"selinux-enabled": false,
```

```
"shutdown-timeout": 15,
"storage-driver": "",
"storage-opts": [],
"swarm-default-advertise-addr": "",
"tls": true,
"tlscacert": "",
"tlscert": "",
"tlskey": "",
"tlsverify": true,
"userland-proxy": false,
"userland-proxy-path": "/usr/libexec/docker-proxy",
"usersns-remap": ""
}
```

### Примечание

Вы не можете установить параметры в `daemon.json`, которые уже были установлены при запуске демона в качестве флага. В системах, использующих `systemd` для запуска демона Docker, `-H` уже установлен, поэтому вы не можете использовать `hosts` ключ в `daemon.json` для добавления адресов прослушивания. Смотрите [Пользовательские параметры демона Docker](#) в качестве примера настройки демона с использованием файлов `systemd drop-in`.

## В Windows

Расположение файла конфигурации по умолчанию в Windows -

`%programdata%\docker\config\daemon.json`. Используйте флаг `--config-file`, чтобы указать расположение, отличное от расположения по умолчанию.

Ниже приведен полный пример разрешенных параметров конфигурации в Windows:

```
{
  "allow-nondistributable-artifacts": [],
  "authorization-plugins": [],
  "bridge": "",
  "containerd": "\\\\.\\pipe\\containerd-containerd",
  "containerd-namespace": "docker",
  "containerd-plugin-namespace": "docker-plugins",
  "data-root": "",

```

```
"debug": true,
"default-network-opts": {},
"default-runtime": "",
"default-ulimits": {},
"dns": [],
"dns-opts": [],
"dns-search": [],
"exec-opts": [],
"experimental": false,
"features": {},
"fixed-cidr": "",
"group": "",
"host-gateway-ip": "",
"hosts": [],
"insecure-registries": [],
"labels": [],
"log-driver": "",
"log-level": "",
"max-concurrent-downloads": 3,
"max-concurrent-uploads": 5,
"max-download-attempts": 5,
"mtu": 0,
"pidfile": "",
"raw-logs": false,
"registry-mirrors": [],
"shutdown-timeout": 15,
"storage-driver": "",
"storage-opts": [],
"swarm-default-advertise-addr": "",
"tlscacert": "",
"tlscert": "",
"tlskey": "",
"tlsverify": true
}
```

Параметр `default-runtime` по умолчанию отключен, и в этом случае `dockerd` автоматически определяет среду выполнения. Это определение основано на том, установлен ли флаг `containerd`.

Полустимые значения:

- `com.docker.hcsshim.v1` - Это встроенная среда выполнения, которую Docker использует с момента первого добавления Windows supported и использует API HCS версии v1 в Windows.
- `io.containerd.runhcs.v1` - Это использует оболочку containerd `runhcs` для запуска контейнера и использует API HCS версии v2 в Windows.


## Параметры функций

Необязательное поле `features` в `daemon.json` позволяет включать или отключать определенные функции daemon.

```
{
  "features": {
    "some-feature": true,
    "some-disabled-feature-enabled-by-default": false
  }
}
```

Оставить отзыв

Список функциональных возможностей включает:

- `containerd-snapshotter`: при значении `true` демон использует контейнерные моментальные снимки вместо классических драйверов хранилища для хранения данных изображений и контейнеров. Для получения дополнительной информации см. в разделе [Контейнерное хранилище](#) .

## Поведение при перезагрузке конфигурации

Некоторые параметры можно перенастроить при запуске демона, не требуя перезапуска процесса. Демон использует `SIGHUP` сигнал в Linux для перезагрузки и глобальное событие в Windows с ключом `Global\docker-daemon-config-$PID`. Вы можете изменить параметры в файле конфигурации, но демон по-прежнему проверяет конфликтующие настройки с помощью указанных флагов CLI. Демону не удастся перенастроить себя при наличии конфликтов, но выполнение не останавливается.

Список поддерживаемых в настоящее время параметров, которые можно перенастроить, выглядит следующим образом:



Опция	Описание
<code>debug</code>	Переключает режим отладки демона.
<code>labels</code>	Заменяет метки демона новым набором меток.
<code>live-restore</code>	Включает <a href="#">оперативное восстановление</a>  .
<code>max-concurrent-downloads</code>	Настраивает максимальное количество одновременных загрузок для каждого извлечения.
<code>max-concurrent-uploads</code>	Настраивает максимальное количество одновременных загрузок для каждого push-запроса.
<code>max-download-attempts</code>	Настраивает максимальное количество попыток загрузки для каждого извлечения.
<code>default-runtime</code>	Настраивает среду выполнения, которая будет использоваться, если при создании контейнера не указано иное.
<code>runtimes</code>	Настраивает список доступных сред выполнения OCI, которые могут использоваться для запуска контейнеров.
<code>authorization-plugin</code>	Определяет используемые плагины авторизации.
<code>allow-nondistributable-artifacts</code>	Определяет список реестров, в которые демон будет отправлять нераспределяемые артефакты.
<code>insecure-registries</code>	Определяет список реестров, которые демон должен считать небезопасными.
<code>registry-mirrors</code>	Определяет список зеркал реестра.
<code>shutdown-timeout</code>	Настраивает существующий тайм-аут конфигурации демона на новый тайм-аут для завершения работы всех контейнеров.
<code>features</code>	Включает или отключает определенные функции.

Оставить отзыв

## Запуск нескольких демонов

### Примечание

Запуск нескольких демонов на одном хосте считается экспериментальным. Вы можете столкнуться с нерешенными проблемами, и в некоторых случаях все может

работать не так, как ожидалось.

В этом разделе описывается, как запустить несколько демонов Docker на одном хосте. Чтобы запустить несколько демонов, необходимо настроить каждого демона таким образом, чтобы он не конфликтовал с другими демонами на том же хосте. Вы можете установить эти параметры либо предоставив их в виде флагов, либо используя [файл конфигурации daemon](#).

Следующие параметры демона должны быть настроены для каждого демона:

<code>-b, --bridge=</code>	Attach containers to a network bridge
<code>--exec-root=/var/run/docker</code>	Root of the Docker execdriver
<code>--data-root=/var/lib/docker</code>	Root of persisted Docker data
<code>-p, --pidfile=/var/run/docker.pid</code>	Path to use for daemon PID file
<code>-H, --host=[]</code>	Daemon socket(s) to connect to
<code>--iptables=true</code>	Enable addition of iptables rules
<code>--config-file=/etc/docker/daemon.json</code>	Daemon configuration file
<code>--tlscacert=~/.docker/ca.pem</code>	Trust certs signed only by this CA
<code>--tlscert=~/.docker/cert.pem</code>	Path to TLS certificate file
<code>--tlskey=~/.docker/key.pem</code>	Path to TLS key file

Оставить отзыв

Когда ваши демоны используют разные значения для этих флагов, вы можете запускать их на одном хосте без каких-либо проблем. Важно, чтобы вы понимали значение этих параметров и правильно их использовали.

- Для `-b, --bridge=` флага установлено значение `docker0` как мостовая сеть по умолчанию. Он создается автоматически при установке Docker. Если вы не используете мост по умолчанию, вы должны создать и настроить мост вручную или установить для него значение "нет": `--bridge=none`
- `--exec-root` это путь, по которому сохраняется состояние контейнера. Значение по умолчанию равно `/var/run/docker`. Укажите здесь путь для вашего запущенного демона.
- `--data-root` это путь, по которому хранятся сохраненные данные, такие как изображения, тома и состояние кластера. Значение по умолчанию - `/var/lib/docker`. Чтобы избежать конфликта с другими демонами, установите этот параметр отдельно для каждого демона.



- `-p, --pidfile=/var/run/docker.pid` это путь, по которому хранится идентификатор процесса демона. Укажите здесь путь к вашему PID-файлу.
- `--host=[]` указывает, где демон Docker прослушивает клиентские подключения. Если не указано, по умолчанию используется значение `/var/run/docker.sock`.
- `--iptables=false` запрещает демону Docker добавлять правила iptables. Если несколько демонов управляют правилами iptables, они могут перезаписать правила, установленные другим демоном. Имейте в виду, что для отключения этой опции требуется вручную добавить правила iptables для предоставления доступа к портам контейнера. Если вы запрещаете Docker добавлять правила iptables, Docker также не добавляет правила маскировки IP, даже если вы установили `--ip-masq` значение `true`. Без правил маскировки IP контейнеры Docker не могут подключаться к внешним хостам или Интернету при использовании сети, отличной от моста по умолчанию.
- `--config-file=/etc/docker/daemon.json` это путь, по которому хранится файл конфигурации. Вы можете использовать его вместо флагов демона. Укажите путь для каждого демона.
- `--tls*` Docker daemon поддерживает `--tlsverify` режим, который обеспечивает принудительное шифрование и аутентификацию удаленных подключений. `--tls*` Опции позволяют использовать определенные сертификаты для отдельных демонов.

Оставить отзыв

Пример сценария для отдельного экземпляра “bootstrap” демона Docker без сети:

```
$ sudo dockerd \  
  -H unix:///var/run/docker-bootstrap.sock \  
  -p /var/run/docker-bootstrap.pid \  
  --iptables=false \  
  --ip-masq=false \  
  --bridge=none \  
  --data-root=/var/lib/docker-bootstrap \  
  --exec-root=/var/run/docker-bootstrap
```

## Параметры сети по умолчанию

`default-network-opts` Ключ в `daemon.json` файле конфигурации и эквивалентный ему `--default-network-opt` флаг CLI позволяют указать значения по умолчанию для параметров сетевого драйвера для новых сетей.

В следующем примере показано, как настроить параметры для `bridge` драйвера с помощью `daemon.json` файла.

```
{
  "default-network-opts": {
    "bridge": {
      "com.docker.network.bridge.host_binding_ipv4": "127.0.0.1",
      "com.docker.network.bridge.mtu": "1234"
    }
  }
}
```

В этом примере используется `bridge` сетевой драйвер. Обратитесь к [страница сетевого драйвера моста](#) для получения обзора доступных параметров драйвера.

После изменения конфигурации и перезапуска демона новые сети, которые вы создаете, используют эти настройки параметров по умолчанию.

```
$ docker network create mynet
$ docker network inspect mynet --format "{{json .Options}}"
{"com.docker.network.bridge.host_binding_ipv4":"127.0.0.1","com.docker.network.bridge.mtu":"12
```

Обратите внимание, что изменение этой конфигурации демона не влияет на уже существующие сети .

Использование `--default-network-opt` флага CLI полезно для тестирования и отладки, но вы должны предпочесть использовать `daemon.json` файл для постоянной настройки демона. Флаг CLI ожидает значение в следующем формате: `driver=opt=value`, например:

```
$ sudo dockerd \
  --default-network-opt bridge=com.docker.network.bridge.host_binding_ipv4=127.0.0.1 \
  --default-network-opt bridge=com.docker.network.bridge.mtu=1234
```

Оставить отзыв



Авторские права © 2013-2024 Docker Inc. Все права защищены.



[Условия обслуживания](#) [Статус](#) [Юридическая информация](#)

Оставить отзыв

