Обзор топ-5 полезных утилит для Docker

[Блог компании FirstVDS](https://habr.com/ru/company/first/blog/)[Open source\*](https://habr.com/ru/hub/open_source/)[Системное администрирование\*](https://habr.com/ru/hub/sys_admin/)



Проект Docker, запущенный в 2013 году, стал одним из самых популярных инструментов в области контейнеризации. Спустя почти 10 лет Docker активно развивается, однако, не только сама компания Docker Inc привносит улучшения в свой продукт – обычные пользователи тоже вносят свой вклад, создавая различные инструменты, которые совершенствуют взаимодействие с системой Docker.

В статье мы рассмотрим топ-5 полезных утилит, которые упростят работу с Docker.

1. Portainer

Официальный сайт: [portainer.io](https://www.portainer.io/)

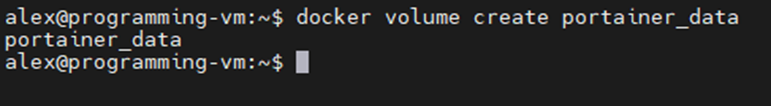
Документация: [docs.portainer.io](https://docs.portainer.io/)

Лицензия: коммерческая и свободная (open source)

С момента запуска в Docker официально никогда не было графического веб-интерфейса. Для решения этой проблемы появился инструмент под названием Portainer — проект с открытым исходным кодом, представляющий собой образ графического web-интерфейса для управления Docker. Помимо предоставления полной информации об установленном Docker, Portainer позволяет управлять всеми сущностями Docker, включая управление контейнерами, образами и сетями.

Произвести [установку](https://docs.portainer.io/start/install/server) можно в отдельно установленный хост (standalone) Docker, а также в кластеры Docker Swarm и Kubernetes. Рассмотрим пример установки в варианте standalone в операционной системе Ubuntu 22.04. Первым шагом необходимо создать том (volume), где Portainer будет хранить свои данные:

docker volume create portainer\_data

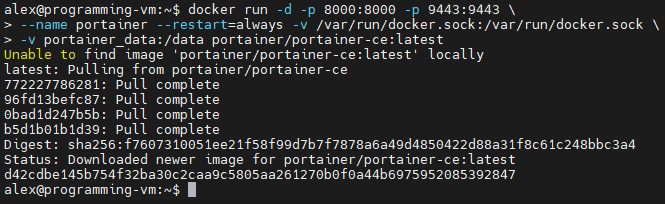


Далее необходимо запустить Portainer:

docker run -d -p 8000:8000 -p 9443:9443 \

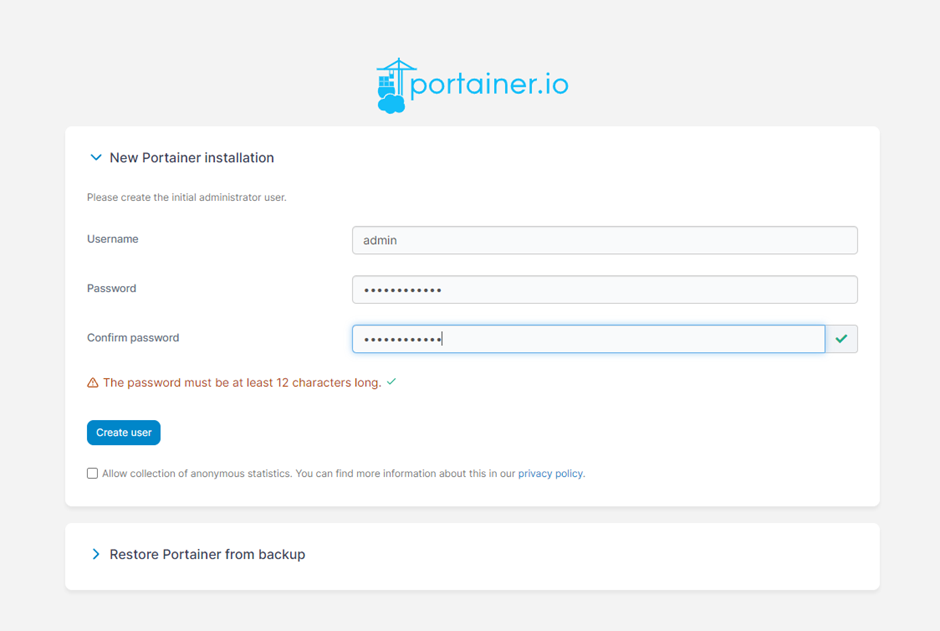
--name portainer --restart=always -v /var/run/docker.sock:/var/run/docker.sock \

-v portainer\_data:/data portainer/portainer-ce:latest

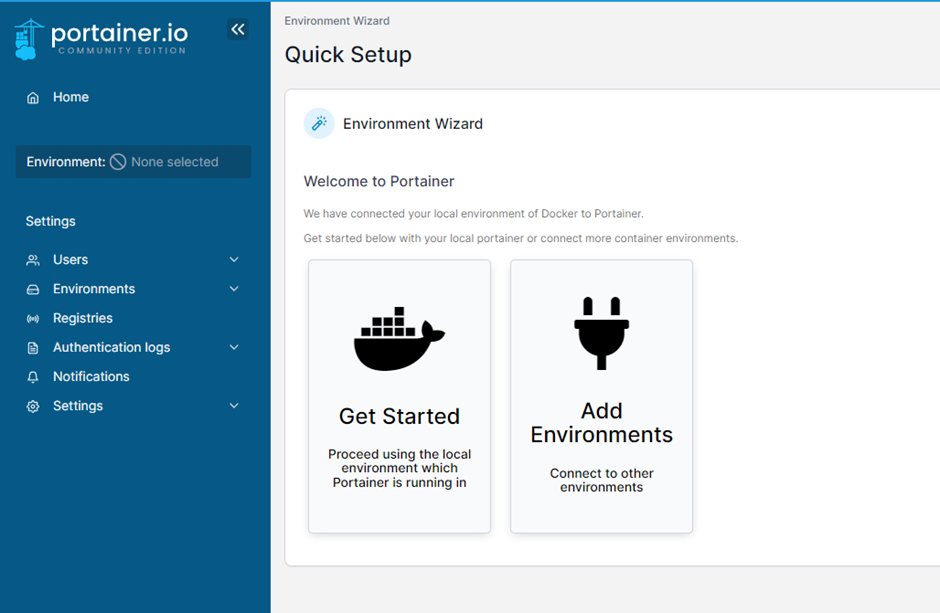


Порт для входа в веб-интерфейс по умолчанию – 9443.

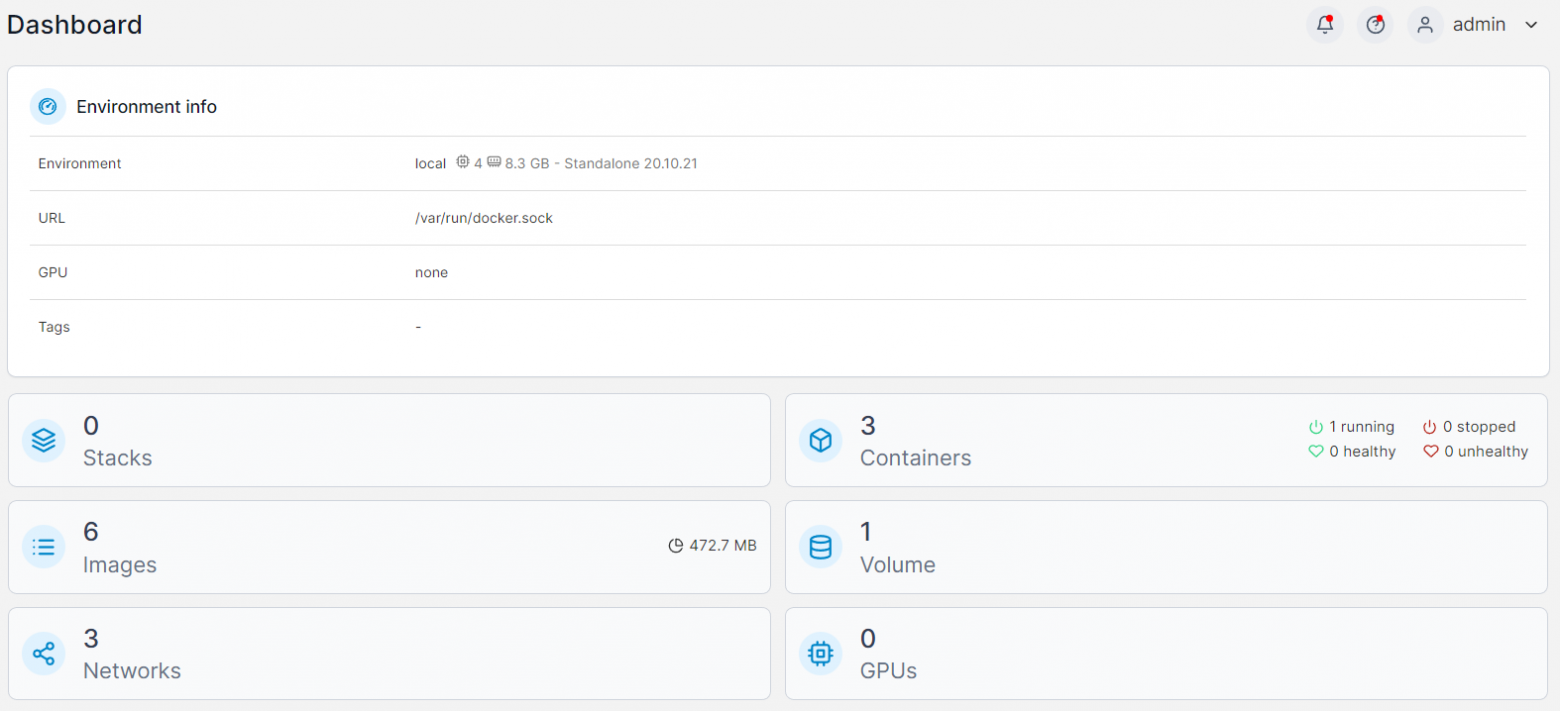
При первом входе необходимо создать пользователя и задать ему пароль:



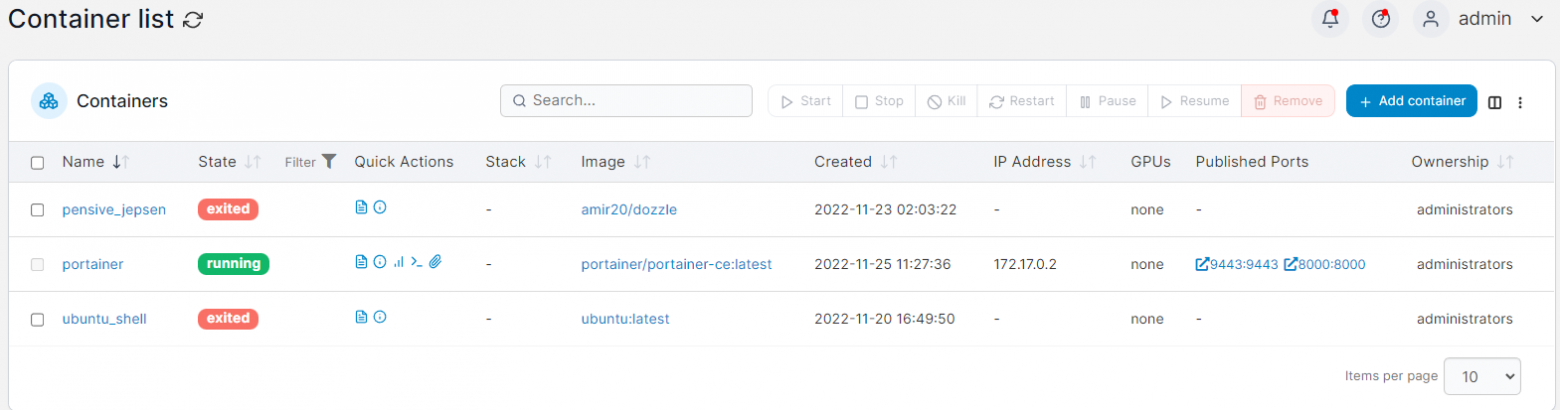
Далее необходимо подключиться к локальному окружению, где установлен Docker, или подключиться к удаленному хосту с установленным там Docker:



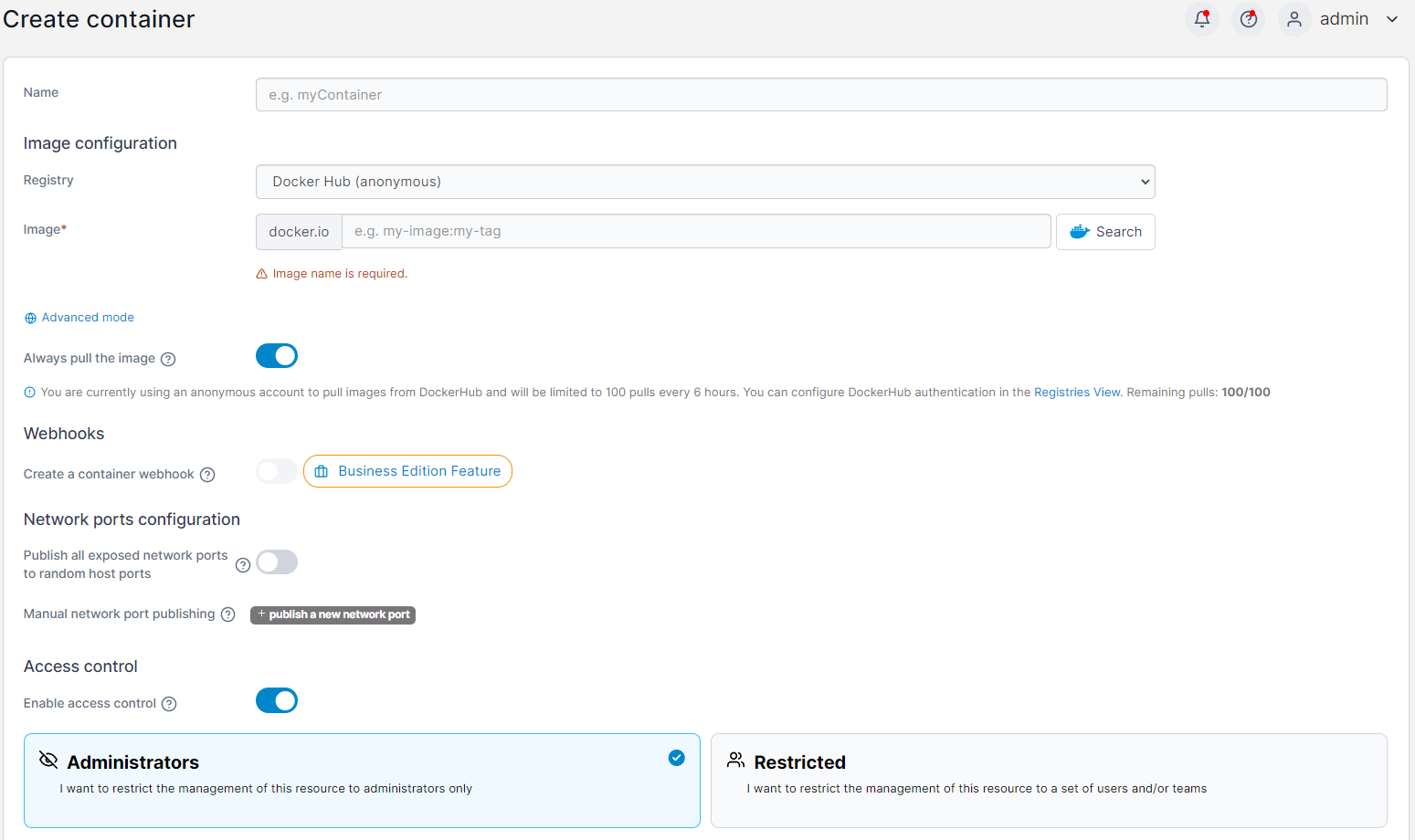
После подключения к необходимому хосту откроется основное меню программы, где будет отображена информация о технических характеристиках сервера, а также — разделы с количеством сервисов, контейнеров, образов, томов и сетей:



В каждый из этих разделов можно перейти и выполнить определенные действия. В разделе **Контейнеры**можно запустить/остановить/завершить/перезапустить/поставить на паузу/удалить контейнер:

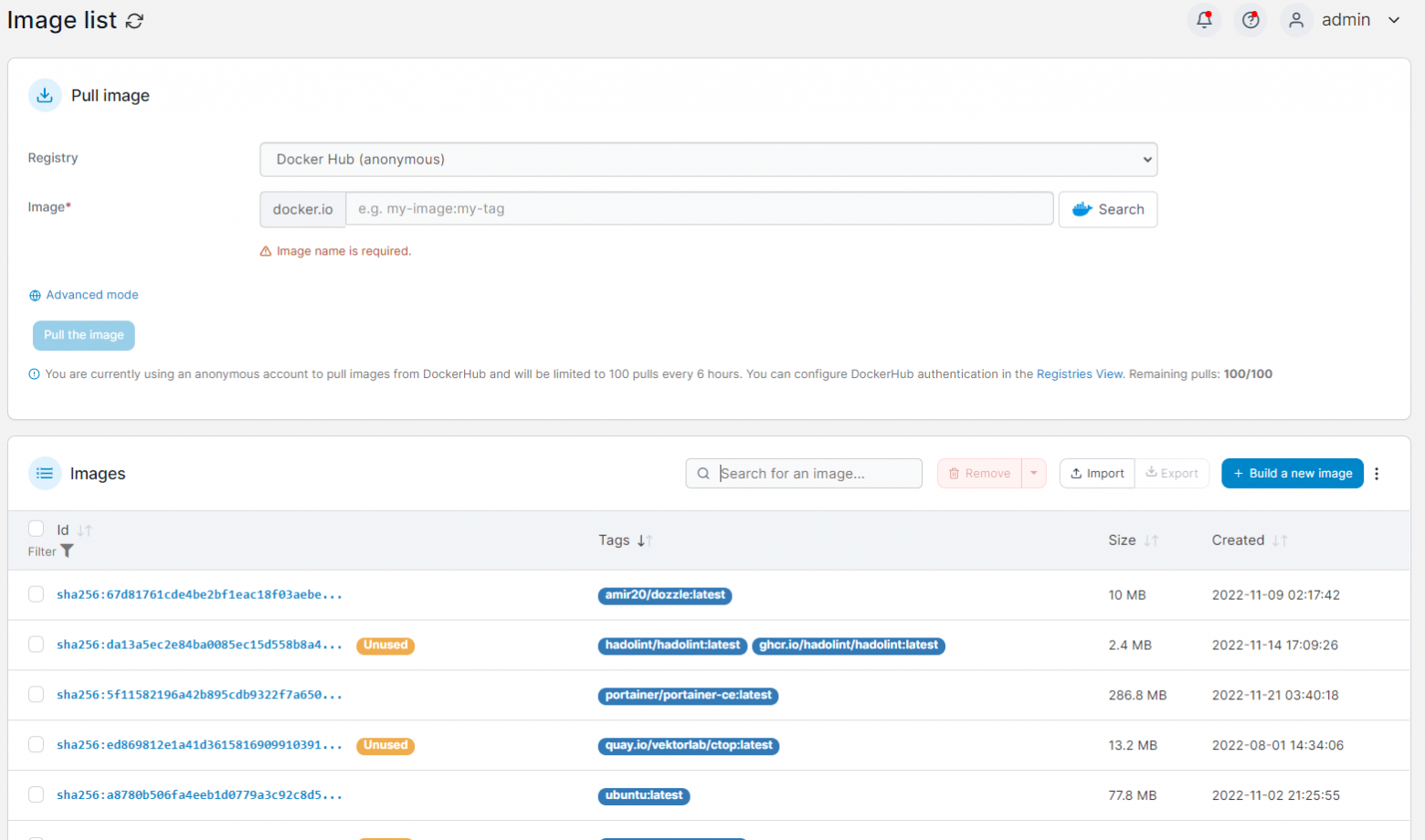


Также в этом разделе можно создать контейнер:

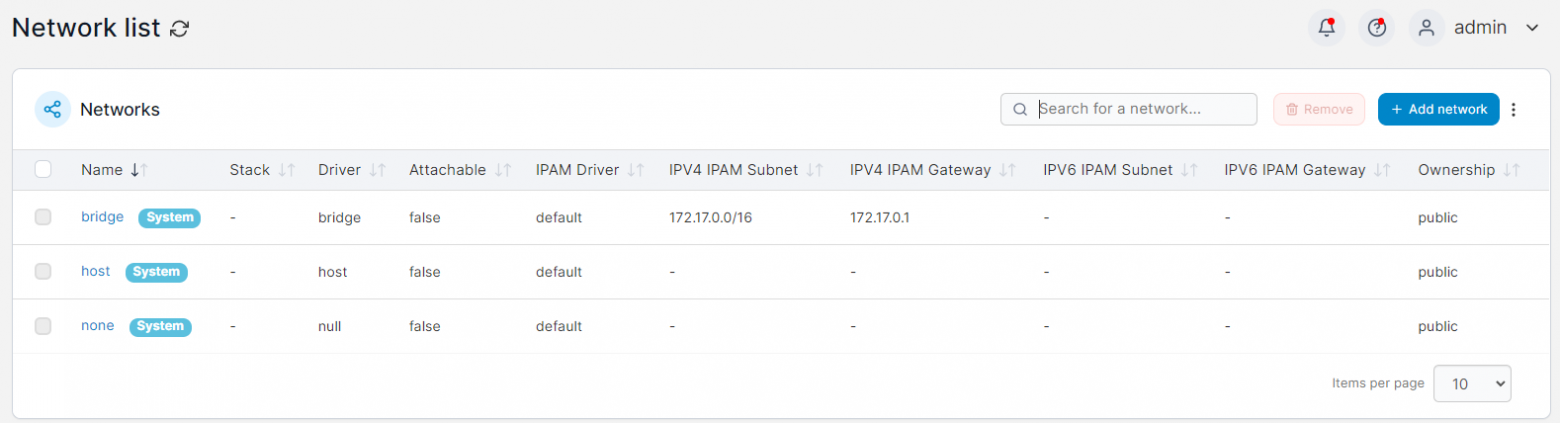


Принцип работы точно такой же, как и у команды **docker run** – задается имя контейнера, образ, при необходимости в контейнере можно открыть порты, а также настроить контроль доступа, только вместо командной строки и параметров здесь используется графический интерфейс.

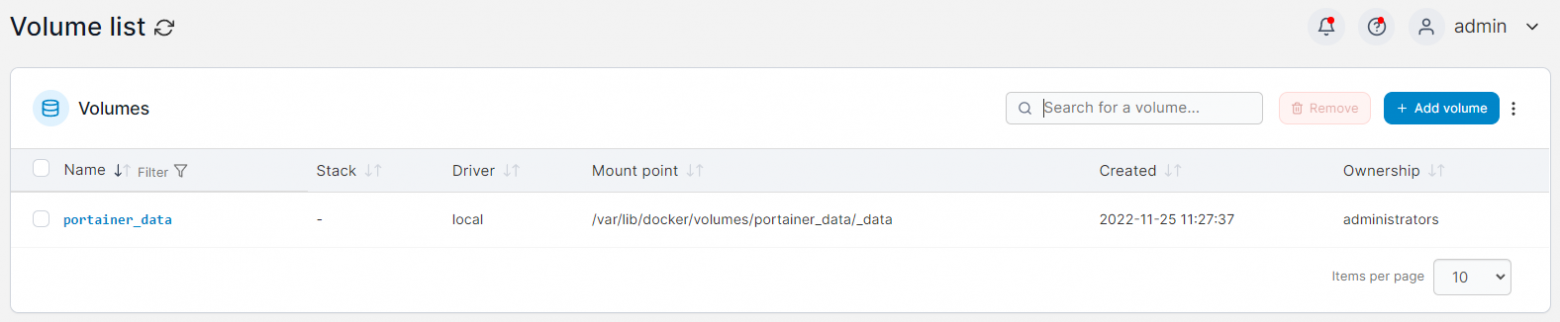
В разделе **Образы**можно скачивать и удалять образы:



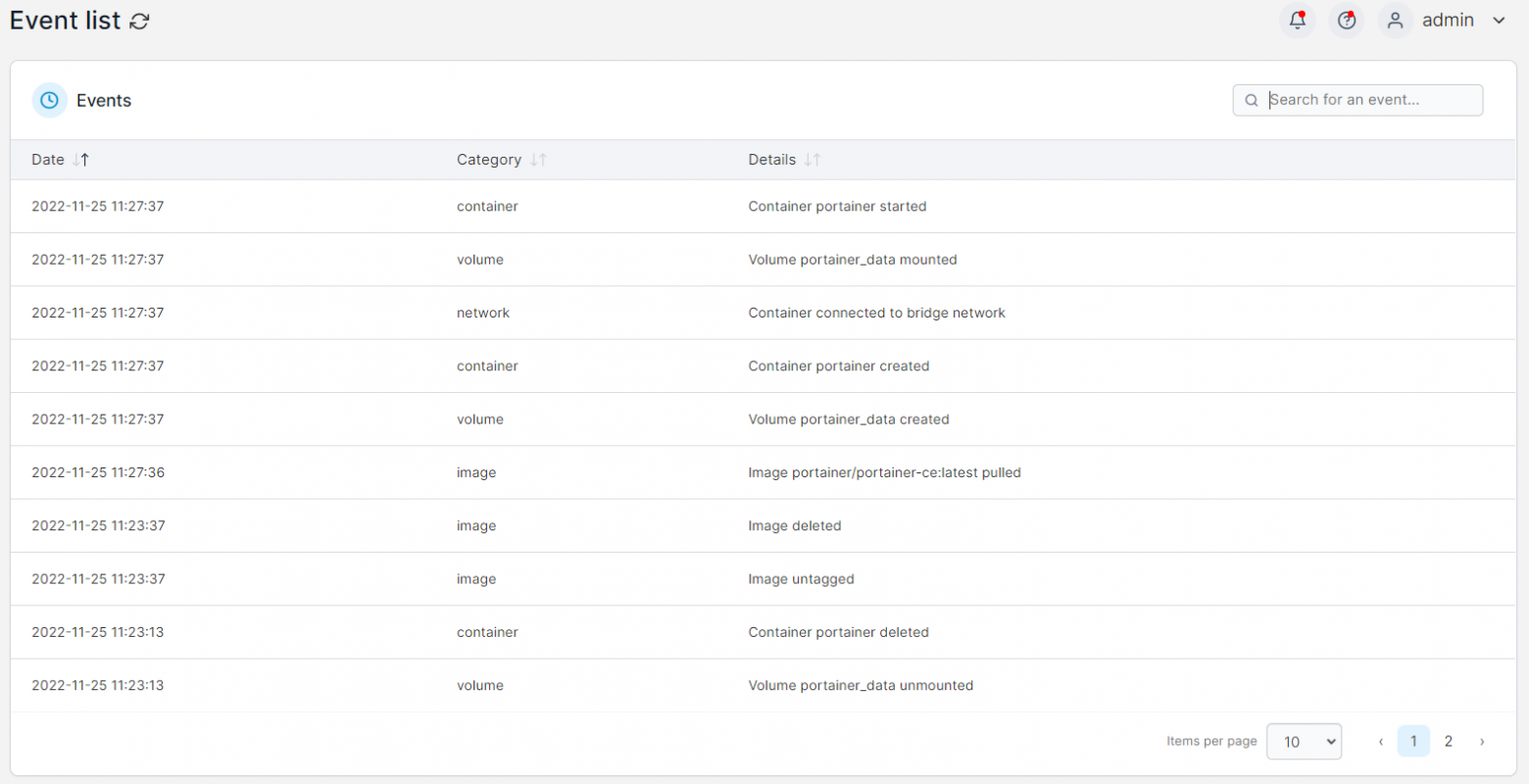
В разделе **Сети**можно создавать и удалять сети:



В разделе **Тома**можно создавать и удалять тома:



Также в Portainer можно просматривать события:



2. Hadolint

Официальный сайт:[GitHub](https://github.com/hadolint/hadolint)

Документация:[GitHub](https://github.com/hadolint/hadolint/blob/master/README.md)

Лицензия: свободная (open source)

Hadolint представляет собой утилиту (линтер), предназначенную для оценки Dockerfile с точки зрения корректности синтаксиса и безопасности инструкций. Также Hadolint проводит проверку на соблюдение лучших практик по написанию инструкций в Dockerfile.

В качестве примера возьмем Dockerfile со следующим содержанием:

FROM debian

RUN export node\_version="0.10" \

&& apt-get update && apt-get -y install nodejs="$node\_verion"

COPY package.json usr/src/app

RUN cd /usr/src/app \

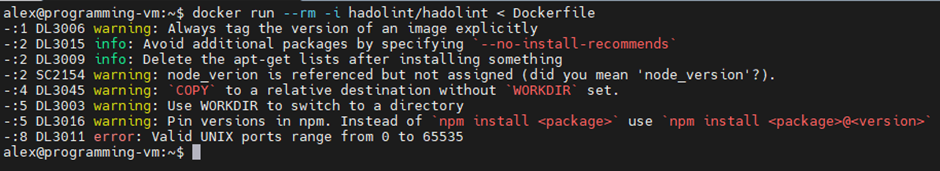
&& npm install node-static

EXPOSE 80000

CMD ["npm", "start"]

Hadolint можно запустить прямо в контейнере Docker для этого достаточно выполнить команду ниже, передав имя Dockerfile в качестве параметра:

docker run --rm -i hadolint/hadolint < Dockerfile



После того, как анализ будет завершен, в терминале будут выведены все недочеты, которые были найдены в Dockerfile. Можно заметить, что утилита указала на то, что в нашем Dockerfile используется тег latest, в то время как необходимо указывать конкретную версию в базовом образе. Также утилита сообщила нам, что для использования портов необходимо использовать диапазон с 0 до 65535, а в нашем Dockerfile указан порт 80000. В качестве уровней предупреждения используются следующие:

**info** — рекомендации не являются обязательными к исправлению, но помогут оптимизировать образ;

**warning**— необходимо обратить внимание на недочеты, как правило, предоставляются советы из[официальной документации](https://docs.docker.com/develop/develop-images/dockerfile_best-practices/) по оптимизации Dockerfile;

**error**— найдены критические ошибки, которые влияют на сборку образа и могут привести к проблем после запуска контейнера.

Для идентификации ошибок Hadolint использует правила. Каждое правило имеет свой уникальный буквенно-цифровой код. Найти все правила и их описания, содержащие шаги по устранению ошибок, можно по [ссылке](https://github.com/hadolint/hadolint#rules).

Если вы не хотите запускать Hadolint в контейнере, его можно[установить на хост](https://github.com/hadolint/hadolint/blob/master/README.md#Install) или использовать[онлайн](https://hadolint.github.io/hadolint).

3. Dive

Официальный сайт:[GitHub](https://github.com/wagoodman/dive)

Документация:[GitHub](https://github.com/wagoodman/dive/blob/master/README.md)

Лицензия: свободная (open source)

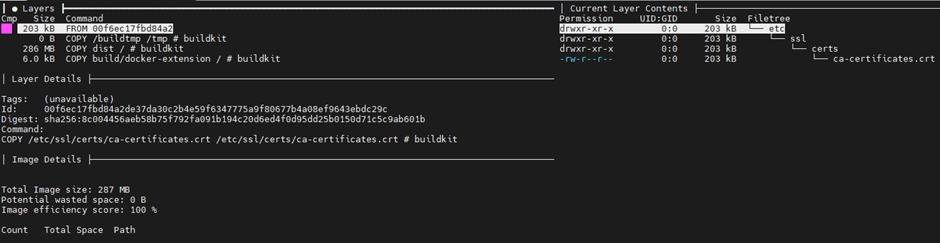
Dive — утилита, которая визуально отображает подробную информацию о Docker образах и их слоях.

Dive можно запустить в контейнере, где последним аргументом указывается название образа или контейнера, который необходимо проанализировать:

docker run --rm -it \

-v /var/run/docker.sock:/var/run/docker.sock \

wagoodman/dive:latest portainer/portainer-ce



На панели справа отображаются слои образа, а слева — подробная информация о каждом слое. Dive позволяет проанализировать размер образа и выявить те слои, которые занимают больше всех места на диске.

Также можно просматривать файлы, которые были изменены, добавлены или удалены. Данные изменения указаны в древовидной структуре файлов справа. Эта функция помогает наглядно отобразить изменения, которые вносились на каждом этапе сборки. Это может оказаться полезным, если нет доступа к файлу Dockerfile, из которого происходит сборка образа.

4. Ctop

Официальный сайт:[GitHub](https://github.com/bcicen/ctop)

Документация:[GitHub](https://github.com/bcicen/ctop/blob/master/README.md)

Лицензия: свободная (open source)

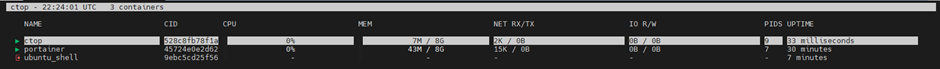
Ctop — утилита для мониторинга метрик в контейнерах, которая напоминает утилиту top в Unix системах. Для запуска утилиты достаточно выполнить команду:

docker run --rm -ti \

--name=ctop \

--volume /var/run/docker.sock:/var/run/docker.sock:ro \

quay.io/vektorlab/ctop:latest

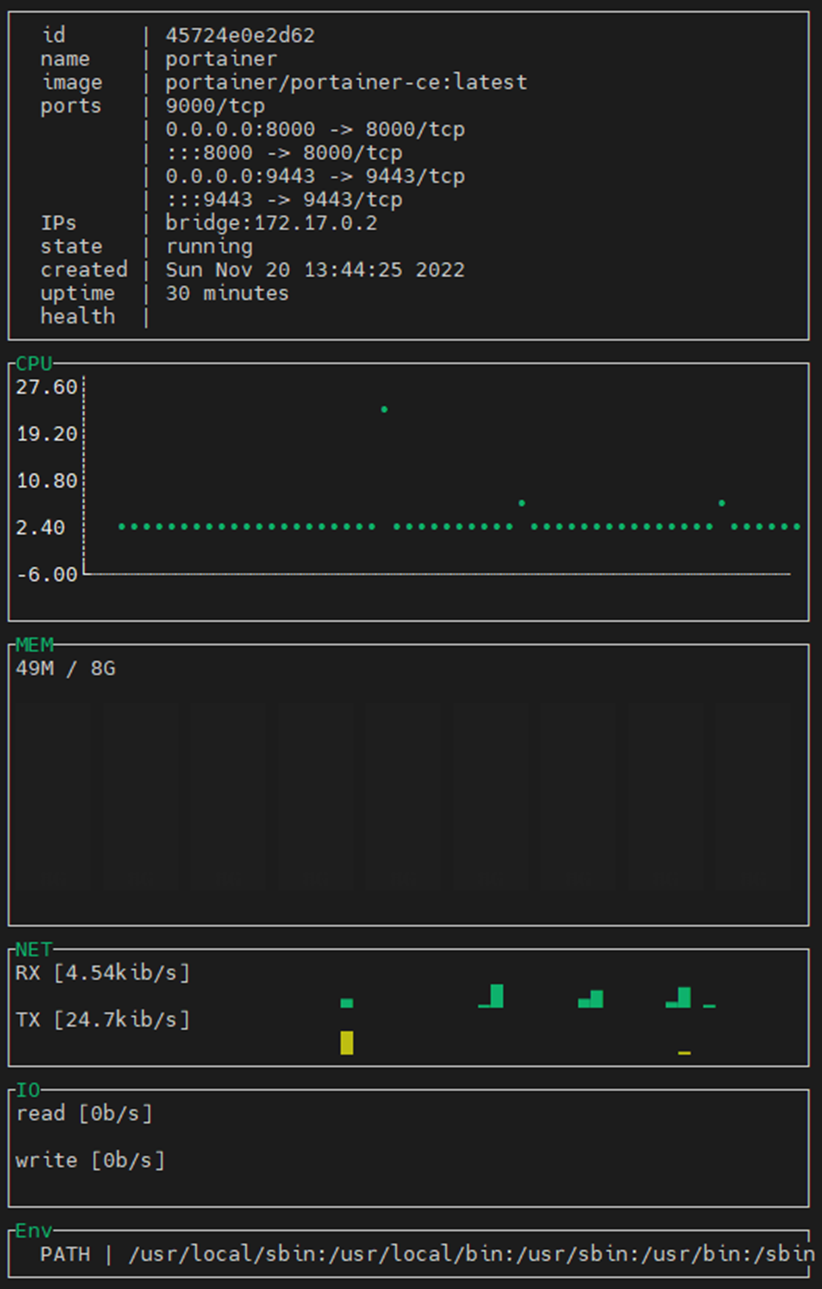


У ctop отсутствуют какие-либо ключи и параметры. После запуска утилита отобразит список всех контейнеров – как запущенных, так и завершенных.

Утилита отображает следующую информацию о контейнерах Docker:

* Имя;
* ID (номер);
* Нагрузку на процессор;
* Нагрузку на оперативную память;
* NET RX/TX — количество переданных/полученных данных;
* IO R/W — количество операций ввода-вывода в секунду;
* PID – уникальный идентификатор процесса;
* uptime.

Также можно посмотреть информацию о конкретном контейнере, для этого необходимо выбрать нужный контейнер и нажать на клавишу **Enter**.



Метрики динамические и получают информацию в режиме реального времени.

5. Dozzle

Официальный сайт:[GitHub](https://github.com/amir20/dozzle)

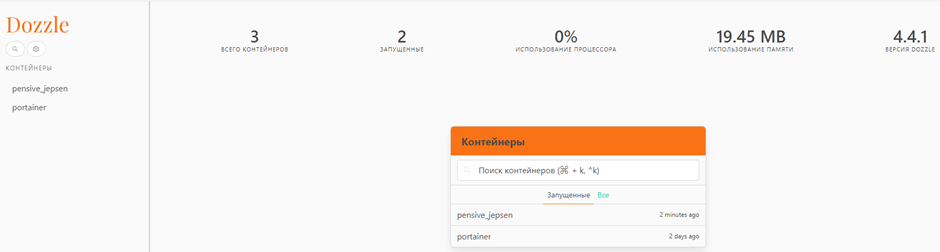
Документация:[GitHub](https://github.com/amir20/dozzle/blob/master/README.md)

Лицензия: свободная (open source)

Dozzle представляет собой веб-интерфейс для отображения логов контейнеров в режиме реального времени. Для запуска Dozzle достаточно одной команды:

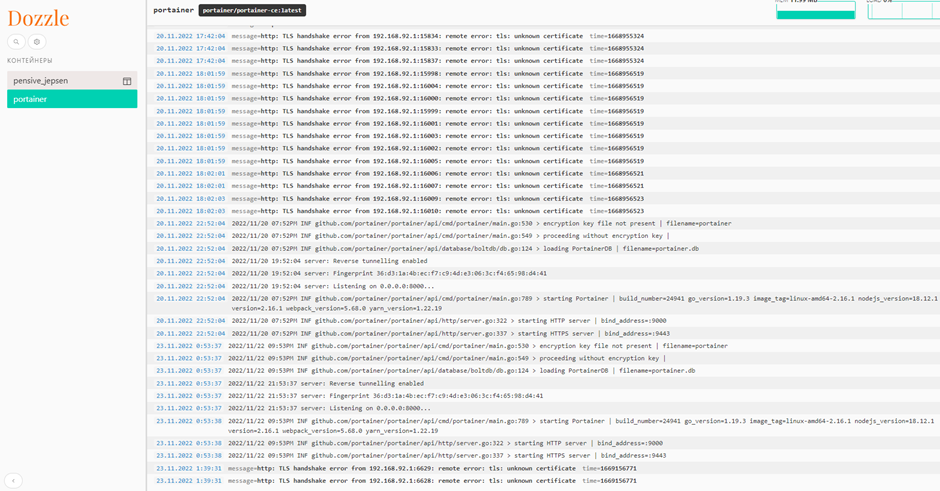
docker run --detach --volume=/var/run/docker.sock:/var/run/docker.sock -p 8080:8080 amir20/dozzle

Для доступа в веб-интерфейс необходимо перейти в браузере по IP-адресу сервера и порту 8080:



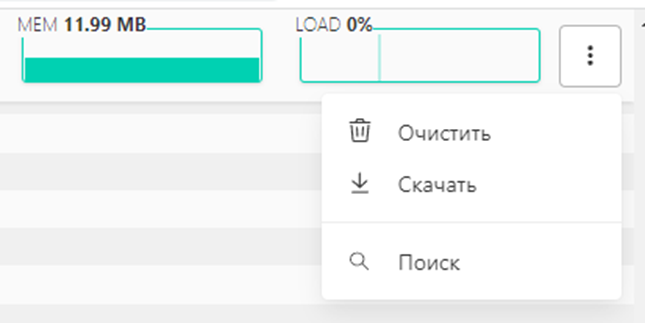
На главной странице отображаются запущенные контейнеры. Для того чтобы просмотреть список всех контейнеров (остановленных и завершенных), необходимо перейти в раздел **«Все»**. Сверху над списком контейнеров расположена информация об их общем количестве, отдельно отображено количество запущенных контейнеров, а также информация об использовании процессора и оперативной памяти.

Чтобы посмотреть лог файл какого-либо контейнера, необходимо выбрать его из списка. Далее откроется файл с логами:



Лог файл динамический, т. е. информация обновляется в режиме реального времени.

Также существует возможность скачать файл с логами, произвести его очистку и воспользоваться поиском:



Итог

Функциональность Docker можно существенно расширить при помощи сторонних утилит. Все описанные в статье утилиты являются бесплатными. У них открытый исходный код, что позволяет вносить в них свои правки, тем самым оптимизируя их под свои нужды. Также каждую утилиту можно запустить в контейнере Docker, не устанавливая ее на хост, что упрощает работу с ней.