????????? ? ????????????? Portainer

https://timeweb.cloud/tutorials/docker/ustanovka-i-ispolzovanie-portainer

Portainer — это инструмент управления контейнерами, который эффективно взаимодействует как с Docker, так и с Kubernetes.

Он доступен в двух версиях:

- бесплатная и открытая Community Edition;
- платная Business Edition с дополнительными функциями для корпоративных клиентов.

В этой статье мы сосредоточимся на установке Portainer на Ubuntu 22.04 и использовании версии Community Edition. Хотя процесс установки и использования будет показан на примере Ubuntu, большинство шагов аналогичны для других операционных систем, что делает информацию применимой для различных сценариев использования.

Portainer отлично подходит как для начинающих, так и для профессионалов в области контейнеризации. Его интуитивно понятный графический интерфейс значительно упрощает управление, делая технологии контейнеров доступными даже для новичков в этой области. Опытные пользователи также найдут в нем богатый выбор опций для тонкой настройки и персонализации.

В статье мы продемонстрируем установку Portainer на локальной машине, но если вы планируете использовать его в команде, то приложение также можно установить на сервере, обеспечивая централизованное управление и доступность для всех членов команды. Основной фокус статьи будет на установке, обзоре основных функций и настроек, подключении внешнего сервера в качестве окружения, а также на примере развертывания WordPress на внешнем сервере с использованием Portainer.

????????? Portainer ? Docker

Шаг 1: Установка Docker и Docker Compose

Перед установкой убедитесь, что на вашей системе установлен Docker. Если Docker уже установлен, этот шаг можно пропустить. В противном случае выполните следующие команды для установки:

```
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh ./get-docker.sh
```

После установки проверьте версии утилит, выполнив команды:

```
docker -v
docker compose version
```

Это подтвердит успешную установку и покажет версии установленных программ.

Шаг 2: Создание рабочего каталога

Создайте каталог для приложения в `/opt` и перейдите в него:

```
cd /opt
sudo mkdir twportainer
cd ./twportainer
```

Шаг 3: Создание файла конфигурации

Теперь создайте файл `docker-compose.yml` в каталоге `twportainer`. В этом файле будет описана конфигурация для запуска. Используйте редактор `nano` или любой другой текстовый редактор для создания файла:

```
sudo nano docker-compose.yml
```

Вставьте в файл следующее содержимое:

```
version: "3.3"
services:
  twportainer:
    image: portainer/portainer-ce:latest
    container_name: twportainer
    environment:
      - TZ=Europe/Moscow
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - /opt/twportainer/portainer_data:/data
    ports:
      - "8000:8000"
      - "9443:9443"
    restart: always
```

Описание параметров:

- `version: "3.3"`: Указывает версию Docker-compose, которую вы используете. Версия 3.3 подходит для большинства современных приложений.
- `services`: В этой секции описываются запускаемые сервисы.
- `twportainer`: Имя сервиса. Используется в качестве идентификатора.
- `image: portainer/portainer-ce:latest`: Определяет образ, который будет использоваться. Здесь используется последняя версия Community Edition.
- `container_name: twportainer`: Назначает имя контейнера, чтобы было легче его идентифицировать.
- `environment`: Позволяет задать переменные окружения. Например, `- TZ=Europe/Moscow` устанавливает временную зону контейнера.
- `volumes`:
  - `/var/run/docker.sock:/var/run/docker.sock` позволяет Portainer взаимодействовать с Docker на вашем хосте;
  - `/opt/twportainer/portainer_data:/data` создает постоянное хранилище данных.
- `ports`:
  - `"8000:8000"` и `"9443:9443"` открывают соответствующие порты для доступа к Portainer. 9443 используется для HTTPS подключения.
- `restart: always`: Гарантирует, что контейнер будет автоматически перезапускаться при необходимости, например, после перезагрузки сервера.

Шаг 4: Запуск

После создания файла конфигурации запустите Portainer командой:

```
docker compose up -d
```

Шаг 5: Доступ к интерфейсу

Portainer теперь запущен и доступен по адресу `https://<ip_или_localhost>:9443`. Откройте этот адрес в браузере для доступа к веб-интерфейсу.

`Image1`

Шаг 6: Создание учётной записи администратора

При первом входе в систему вас попросят зарегистрировать аккаунт администратора. Учтите, что для пароля требуется минимум 12 символов. Завершив процесс регистрации, вы получите доступ к настройкам и функционалу управления контейнерами в интерфейсе.

????? ???????? ??????? ???? ? ????????

Для доступа к настройкам перейдите на вкладку «Settings». Здесь будут рассмотрены ключевые настройки, которые наиболее важны для базовой конфигурации. Для более глубокого понимания всех доступных настроек рекомендуем ознакомиться с официальной документацией.

`Image3`

1. **Application settings**. В этом разделе вы можете настроить такие параметры, как частота создания снапшотов состояния и отправку анонимной статистики использования приложения.
2. **App Templates**. Здесь можно указать URL JSON-файла с шаблонами для быстрого развёртывания контейнеров. Также доступна возможность использовать предустановленные шаблоны, что облегчает процесс запуска новых приложений.

`Image12`

3. **SSL certificate**. В этом разделе предоставляется возможность загрузить собственные SSL- сертификаты для безопасного соединения. Хотя это не требуется для локальной установки, при развёртывании Portainer на удалённом сервере подключение собственного SSL-сертификата повышает безопасность.

`Image6`

4. **Backup up Portainer**. Этот раздел позволяет создать резервную копию настроек и конфигурации приложения. Такой подход полезен для обеспечения безопасности

данных и упрощения процесса миграции на другие системы.

Image7

5. **Authentication**. Здесь можно настроить продолжительность пользовательской сессии и выбрать метод аутентификации. В Community Edition доступны следующие методы: Internal (используется по умолчанию), LDAP и OAuth. Однако, стоит отметить, что настройка OAuth в Community Edition имеет ограничения, и такие популярные сервисы, как Microsoft OAuth, Google OAuth, Github OAuth, не поддерживаются, что требует ручной настройки. При использовании внутренней аутентификации (Internal) вы можете изменить требования к паролю, например, уменьшить минимальное количество символов.

Для изменения собственного пароля перейдите в правый верхний угол экрана, нажмите на имя вашего аккаунта и выберите «My account». Это позволит вам обновить свой пароль и другие личные настройки.

После изучения основных настроек, перейдем к другим важным разделам, доступным в левом меню интерфейса.

Users. Этот раздел предназначен для управления пользователями. Он особенно полезен в тех случаях, когда система используется командой для разграничения доступа к ресурсам. Здесь можно создавать отдельных пользователей и управлять ими. В дополнение, существует вкладка «Teams», позволяющая формировать команды из различных пользователей для более гранулярного контроля доступа. Однако, следует отметить, что более продвинутые настройки ролей доступны только в Business Edition.

Image2

Registries. В разделе «Registries» пользователи могут настроить доступ к репозиториям образов. Интерфейс управления облегчает интеграцию с популярными хранилищами, такими как DockerHub, AWS ECR, Quay.io, ProGet, Azure и GitLab, позволяя эффективно управлять образами контейнеров непосредственно через графический пользовательский интерфейс.

Image10

Environments. Ключевой раздел Portainer для подключения к внешним серверам или окружениям и управления ими. Здесь можно управлять различными средами, включая Docker, Docker Swarm, Kubernetes и ACI. В Business Edition также доступен Nomad. Этот раздел позволяет Portainer Server управлять множеством окружений, упрощая масштабирование и управление инфраструктурой.

?????????? ?????? ?????????

Чтобы продемонстрировать процесс добавления нового окружения, мы подключим сервер на Ubuntu 22.04 с предварительно установленным Docker. Это может быть как новый сервер,

так и сервер, на котором уже работают контейнеры.

1. Начните с нажатия кнопки «Add environment» на странице Environments.
2. Выберите «Docker Standalone» и воспользуйтесь мастером настройки, нажав «Start Wizard».

`Image13`

3. В процессе настройки выберите пункт «Agent» и выполните следующую команду на сервере, который вы планируете подключить в качестве окружения:

```
docker run -d \
-p 9001:9001 \
--name portainer_agent \
--restart=always \
-v /var/run/docker.sock:/var/run/docker.sock \
-v /var/lib/docker/volumes:/var/lib/docker/volumes \
portainer/agent:2.19.4
```

Эта команда запустит агент Portainer, позволяя Portainer Server подключаться к серверу и управлять контейнерами.

4. После успешной установки и запуска агента на сервере, вернитесь в веб-интерфейс и завершите процесс подключения, указав имя окружения и его адрес в формате `ip_сервера:9001`.

`Image9`

5. Нажмите «Connect» для завершения подключения. После успешного добавления окружения в интерфейсе отобразится всплывающее уведомление «Environment created».

????? ??????? ???? ??? ?????????

При переходе на страницу «Home» мы увидим два окружения: «local» (устройство, где запущено приложение) и ранее добавленный сервер. После выбора добавленного ранее сервера, меню слева обновится, добавляя функции управления для окружения.

`Image4`

????? ??????? ??????????? ???????????

Images. В разделе «Images» отображаются все доступные образы в системе. Здесь можно удалять образы по отдельности или массово, а также скачивать новые образы через опцию «Pull image».

`Image16`

Networks. На странице «Networks» представлены все доступные сети. С помощью интуитивно понятного мастера настройки, доступного через «Add Network», пользователи могут создавать новые сети, расширяя возможности связи между контейнерами.

`Image11`

Volumes. В разделе «Volumes» находится информация о всех томах. Этот раздел позволяет не только просматривать существующие тома, но и удалять их или создавать новые с помощью мастера настройки «Add volume».

Containers. Раздел «Containers» предоставляет обширные возможности управления контейнерами. В этом разделе видны все существующие контейнеры, их можно удалять, приостанавливать, активировать или перезапускать. Доступ к дополнительным функциям, включая просмотр информации о контейнере, статистики и доступа к консоли, осуществляется через меню «Quick Actions», расположенное в соответствующем столбце.

`Image14`

Создание нового контейнера осуществляется через мастер настройки «Add container». В качестве примера можно создать контейнер с Nginx, указав имя, образ «nginx» и настроив сетевые порты (нажмем на «publish a new network port» и укажем в `host` 9090 порт, а в `container` 80).

`Image5`

Далее нажмем на кнопку «Deploy the container» и дождемся завершения развертывания контейнера. По завершению произойдет редирект на страницу «Container list». После развертывания контейнера переход по адресу `http://ip_сервера:9090` покажет работающий Nginx.

??????????? ???????????: App Templates ? Stacks

Раздел «**App Templates**» представляет собой коллекцию предварительно настроенных шаблонов для развертывания распространенных приложений и сервисов. Эти шаблоны предназначены для упрощения процесса создания новых контейнеров, минимизируя необходимость в ручной конфигурации. Пользователи могут выбирать из разнообразия доступных шаблонов, которые варьируются от базовых веб-серверов до сложных многоуровневых приложений.

При использовании шаблона достаточно указать некоторые основные параметры, такие как имя контейнера, сетевые настройки и, в некоторых случаях, специфические настройки, например, пароли или переменные среды. Это делает раздел «App Templates» особенно удобным для быстрого тестирования новых идей и утилит, а также для обучения и экспериментирования с новыми технологиями.

`Image8`

Stacks — это эффективный способ управления группами контейнеров. Определяются они через файлы `docker-compose`, группируя их для выполнения общих задач. Это упрощает управление сложными приложениями и обеспечивает автоматизацию и консистентность в развертывании.

????????????? ?????????????? Stacks ??? ?????????????? WordPress

Особенностью Stacks является гибкость в выборе метода определения конфигурации: можно использовать встроенный редактор для прямого написания или редактирования файлов Docker Compose, загрузить уже готовый `docker-compose.yml`-файл или даже подключить git-репозиторий для автоматического обновления и развертывания контейнеров. Теперь давайте применим эту технологию на практике. На примере развертывания WordPress покажем, как использовать Stacks для создания и управления многоконтейнерными приложениями. Этот пример поможет понять, как можно упростить и автоматизировать процессы развертывания ваших приложений с помощью Stacks.

- Во вкладке «Stacks» нажмите на «Add stack», чтобы открыть конфигуратор.
- Используя «Web editor», задайте имя и опишите конфигурацию приложения в YAML-формате. Для WordPress и базы данных MariaDB конфигурация может выглядеть следующим образом:

```
services:
  db:
    image: mariadb:10.6.4-focal
    command: '--default-authentication-plugin=mysql_native_password'
    volumes:
      - db_data:/var/lib/mysql
    restart: always
    expose:
      - 3306
      - 33060
    environment:
```

```
  - MYSQL_ROOT_PASSWORD=twtest
  - MYSQL_DATABASE=tw_wp
  - MYSQL_USER=tw
  - MYSQL_PASSWORD=password
  wordpress:
image: wordpress:latest
ports:
  - 80:80
restart: always
environment:
  - WORDPRESS_DB_HOST=db
  - WORDPRESS_DB_USER=tw
  - WORDPRESS_DB_PASSWORD=password
  - WORDPRESS_DB_NAME=tw_wp
volumes:
  db_data:
```

- Переменные окружения могут быть вынесены в отдельный раздел. В «Environment variables» выберите «Advanced mode» и укажите переменные:

```
MYSQL_ROOT_PASSWORD=twtest
MYSQL_DATABASE=tw_wp
MYSQL_USER=tw
MYSQL_PASSWORD=password
WORDPRESS_DB_HOST=db
WORDPRESS_DB_USER=tw
WORDPRESS_DB_PASSWORD=password
WORDPRESS_DB_NAME=tw_wp
```

- Удалите раздел `environment` из основного YAML-файла, чтобы избежать дублирования:

```
services:
  db:
image: mariadb:10.6.4-focal
command: '--default-authentication-plugin=mysql_native_password'
volumes:
```

```
  - db_data:/var/lib/mysql
restart: always
expose:
  - 3306
  - 33060
  wordpress:
image: wordpress:latest
ports:
  - 80:80
restart: always
volumes:
  db_data:
```

`Image15`

- Нажмите «Deploy the stack». После некоторого времени ожидания, при успешном развертывании, произойдет перенаправление на страницу «Stacks list», где будет отображаться наш экземпляр Wordpress.
- Проверьте работу WordPress, перейдя по адресу `http://ip_сервера:80`. Вы должны увидеть стартовую страницу WordPress, что подтверждает успешное развертывание.

Revision #3
Created 9 November 2025 10:06:41 by itimokhin
Updated 9 November 2025 10:17:18 by itimokhin