# Настройка окружения под Web

## Node.js

## **Node Version Manager**

Для управления установленной версией Node.js рекомендуется использовать Node Version Manager.

Установка или обновление nvm (версия 0.39.3):

```
curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
```

```
wget -q0- https://raw.githubusercontent.com/nvm-sh/nvm/v0.39.3/install.sh | bash
```

Открыть новое окно терминала для загрузки установленных переменных окружения и выполнить

```
nvm --version
```

## Установка Node.js

1. Определение актуальной версии на сайте проекта Node.js или через nvm командой:

```
nvm ls-remote
```

2. Установка требуемой версии node.js:

```
nvm install 18.16.1
```

3. Проверка установки в новом окне терминала:

```
node --version
```

## **Docker**

## Учебные материалы

Экосистема Docker (DigitalOcean):

- знакомство с общими компонентами;
- обзор контейнеризации;
- сетевое взаимодействие;
- распределение задач и оркестровка;
- обнаружение сервисов и распределённые хранилища конфигураций.

Изучаем Docker (Хабр):

14:15

upuale. 2024/02/13 itechnology:web\_develop:environment https://jurik-phys.net/itechnology:web\_develop:environment

- часть 1: основы;
- часть 2: термины и концепции;
- часть 3: файлы Dockerfile;
- часть 4: уменьшение размеров образов и ускорение их сборки;
- часть 5: команды;
- часть 6: работа с данными.

#### Официальное руководство:

• Основные консольные команды

## Основные компоненты и инструменты

- Docker Engine ядро, базовый компонент, отвечающий за создание и запуск контейнеров. Обычно под Docker'ом подразумевают именно Docker Engine. Существует две версии Docker Engine: проприетарная (Docker Engine Enterprise) и открытая (Docker Engine Community).
- Docker Desktop предоставляет изолированное окружение для запуска Docker Engine, а также графический интерфейс для создания, запуска и управления контейнерами. Особенностью является то, что запуск контейнеров происходит внутри виртуальной машины как в Windows, MacOS, так и в Linux. В последнем случае не рекомендуется устанавливать Docker Desktop без веских причин.
- Docker CLI tool набор инструментов командной строки, которые используются для взаимодействия с Docker Engine с целью запуска контейнеров, создания новых образов и т.д.
- Docker Compose инструмент работы с многоконтейнерными приложениями. Выполняет команды, описываемые в файле docker-compose.yml для, например, сборки нескольких контейнеров.
- Docker Registry облачное хранилище образов контейнеров, позволяет создавать контейнеры на основе представленных в реестре образов. Примером может служить публичный реестр образов Docker Hub, используемый при работе с Docker по умолчанию.

#### Безопасность

Docker контейнер, являющийся с точки зрения хост-системы процессом, всегда запускается от пользователя root, при этом внутри контейнера по умолчанию также используется root пользователь. Этим обусловлено повышенное внимание к требованиям безопасности при использовании docker'a.

Способы повышения привилегий для запуска контейнера:

- Добавление пользователя в группу docker. Данный способ категорически не рекомендуется, т.к. фактически пользователь наделяется root-правами. Пользователь из группы docker может запустить контейнер (в нём он будет root), примонтировать (опцией -v) часть файловой системы хоста внутрь контейнера и модифицировать её каким угодно способом. Видео с демонстрацией уязвимости по ссылке.
- Использование sudo рекомендованный способ. В Debian'e sudo необходимо установить и настроить. Установка:

#### apt install sudo

Для настройки достаточно добавить пользователя в группу sudo:

```
usermod —a —G sudo user_name
# или
adduser user_name sudo
```

Изменения вступят в силу после повторной авторизации пользователя в системе. Если пользователь user\_name имеет доступ sudo, то на выходе команды будет root:

#### sudo whoami

О дополнительных мерах усиления безопасности при использовании docker контейнеров см. ссылку.

#### Установка в Debian

Для свободной установки доступно несколько вариантов:

- docker.io название пакета в репозитории дистрибутивов в основанных на Debian;
- docker-ce официальная общественная (community edition) версия Docker'a.

Версия из репозитория лучше интегрирована в систему, более предсказуемо обновляется, но может не иметь новых возможностей официальной версии, рекомендуется использовать официальную версию, чтобы лишний раз не наступать на подводные камни возможной несовместимости двух версий.

#### Шаг №1. Удаление версии из репозитория

```
apt purge docker docker.io docker-compose
```

#### Шаг №2. Установка пакетов для работы с репозиториями

Скорее всего данные пакеты в системе уже будут установлены

```
apt install ca-certificates curl gnupg lsb-release
```

### Шаг №3. Добавление официального GPG ключа Docker репозитория в систему

```
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --no-default-keyring
--keyring gnupg-ring:/etc/apt/trusted.gpg.d/docker-pub.gpg --import
chmod 644 /etc/apt/trusted.gpg.d/docker-pub.gpg
```

#### Шаг №4. Добавление репозитория в систему

```
echo "deb [arch=amd64 signed-by=/etc/apt/trusted.gpg.d/docker-pub.gpg]
```

Last update:

upuate. 2024/02/13 itechnology:web\_develop:environment https://jurik-phys.net/itechnology:web\_develop:environment

14:15

```
https://download.docker.com/linux/debian $(lsb\_release -cs) stable" | tee
/etc/apt/sources.list.d/docker.list > /dev/null
```

#### Шаг №5. Обновление списка доступных пакетов

apt update

#### Шаг №6. Установка Docker'а

apt install docker-ce

При этом будут установлены все необходимые компоненты: containerd.io, docker-compose-plugin и др.

#### Шаг №7. Проверка статуса Docker сервиса

systemctl status docker

#### Шаг №8. Запуск первого контейнера

```
sudo docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
719385e32844: Pull complete
Digest: sha256:dcba6daec718f547568c562956fa47e1b03673dd010fe6ee58ca806767031d1c
Status: Downloaded newer image for hello-world:latest
Hello from Docker!
This message shows that your installation appears to be working correctly.
To generate this message, Docker took the following steps:
```

- 1. The Docker client contacted the Docker daemon.
- 2. The Docker daemon pulled the "hello-world" image from the Docker Hub. (amd64)
- 3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
- 4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something **more** ambitious, you can run an Ubuntu container with: \$ docker run -it ubuntu **bash** 

Share images, automate workflows, and **more** with a **free** Docker ID: https://hub.docker.com/

For more examples and ideas, visit:

```
https://docs.docker.com/get-started/
```

## Настройка

#### Настройка Firewall

Если межсетевой экран в хост системе использует правила фильтрации по умолчанию, то Docker при установке настроит возможность проброса портов из контейнера в хост систему, а также обеспечит контейнеры выходом в интернет.

Использование собственных правил фильтрации и/или утилит, реализующих удобную генерацию правил фильтрации, скорее всего, потребует самостоятельной настройки данных возможностей.

**Проброс портов.** При установке Docker'а в хост системе появляется сетевой интерфейс docker0, для которого необходимо разрешить устанавливать входящие соединения:

```
iptables -I INPUT -i docker0 -j ACCEPT iptables -I OUTPUT -o docker0 -j ACCEPT
```

Для FireHol'a правила будут выглядеть следующим образом:

```
interface docker0 DockerNET
    client all accept
    server all accept
```

**Интернет для контейнера.** Для работы интернета внутри контейнера необходимо настроить NAT для интерфейса docker0:

```
iptables -I FORWARD -i docker0 -o eth0    -j ACCEPT
iptables -I FORWARD -i eth0    -o docker0 -j ACCEPT
iptables -P FORWARD DROP
iptables -t nat -I POSTROUTING -s 172.17.0.1/16 -o eth0 -j MASQUERADE
```

Здесь eth0 - внешний сетевой интерфейс с выходом в интернет, 172.17.0.1/16 - подсеть Docker'а по умолчанию. Соответствующая настройка NAT через FireHol:

```
router Wan-to-Docker inface eth0 outface docker0
route all accept
router Docker-to-Wan inface docker0 outface eth0
route all accept
```

Дополнительно о совместной работе FireHol'a и Docker'a см. обсуждение на GitHub'e; об особенностях работы Docker'a с iptables в статье по ссылке.

## Проверка проброса портов

```
sudo docker run -p 8000:80 ubuntu/apache2
```

Переход в хост системе по адресу http://localhost:8000 должен показать «Apache2 Default Page»

Last

upuate. 2024/02/13 itechnology:web\_develop:environment https://jurik-phys.net/itechnology:web\_develop:environment

14:15

#### Проверка интернета в контейнере

Список запущенных контейнеров

docker **ps** CONTAINER ID IMAGE CREATED **STATUS** COMMAND PORTS NAMES Up 51 minutes 6eed6a35de0c ubuntu/apache2 "apache2-foreground" 24 hours ago 0.0.0.0:8000->80/tcp, :::8000->80/tcp cranky\_bartik

Подключение к запущенному контейнеру либо через eго ID «6eed6a35de0c», либо через NAME «cranky\_bartik»

sudo docker exec -it cranky\_bartik /bin/bash

В оболочке контейнера попытаться обновить список доступных пакетов:

root@6eed6a35de0c:/# apt update

При наличии интернета в контейнере начнётся стандартное обновление списка доступных пактов.

## **Реестр DockerHub**

Docker Hub — это публичный реестр настроенных образов, которые можно использовать для быстрой контейнеризации своих приложений с помощью Docker

**Регистрация.** Для взаимодействия с публичным реестром DockerHub требуется учётная запись, создать которую можно по ссылке.

Авторизация. Для взаимодействия с DockerHub из консоли необходимо авторизоваться с помощью команды:

sudo docker login

Также авторизоваться можно на сайте DockerHub'a.

Выход из реестра, соответственно:

sudo docker logout

**Поиск образов.** Для поиска публичных образов в DockerHub'e служит команда:

sudo docker search ubuntu

Здесь «ubuntu» - произвольное имя искомого docker-образа. Результаты поиска можно фильтровать, например, по числу звёзд и принадлежности к разработчикам ПО:

docker search --filter is-official=true --filter stars=99 ubuntu

Дополнительно при поиске через сайт DockerHub можно узнать общее число скачиваний образа.

## Работа с docker-образом

**Docker-образ** представляет собой неизменяемую основу (файловая система, метаданные и настройки) на базе которой, создается и запускается Docker-контейнер, являющийся запущенным экземпляром docker-образа.

**Скачать из реестра.** После поиска необходимого образа, например ubuntu/apache2, его можно загрузить:

```
sudo docker pull ubuntu/apache2
```

Получить список образов. Доступные локально образы:

```
sudo docker images
```

Удалить образ. Удалить локальный образ «hello-world»:

```
sudo docker image rm hello-world
# or
sudo docker rmi hello-world
```

Удаление доступно, когда к образу не прикреплён какой-либо контейнер.

#### Создание docker-образа:

• на базе файла Dockerfile

```
sudo docker build -t you_image_name Dockerfile .
```

• из контейнера

```
sudo docker commit container you_image_name
```

#### Передача переменных окружения в контейнер:

```
sudo docker run --name "openapi-editor" -d -p 1010:8080 -e
URL_SWAGGER2_GENERATOR=null -e URL_0AS3_GENERATOR=null swaggerapi/swagger-editor
```

Здесь запускается контейнер с Swagger Editor'ом, в котором отключены функции генерации кода.

## Работа с docker-контейнером

**Получить список контейнеров.** Получить список всех локально существующих контейнеров. Параметр «-а» выведет полный список контейнеров, включая со статусом «Exited», параметр «-s» покажет соответвтвующий размер.:

```
sudo docker ps -as
```

14:15

upuale. 2024/02/13 itechnology:web\_develop:environment https://jurik-phys.net/itechnology:web\_develop:environment

#### Создать и запустить контейнер:

- создается новый контейнер на базе docker-образа «ubuntu/apache2» с именем «container-apach2»;
- запускается в фоне с выводом в терминал container id.
- доступ к 80-му порту веб-сервера apache2 из контейнера предоставляется через 1010 порт системы на которой запущена docker-инфраструктура

```
sudo docker run --name "container-apache2" -d -p 1010:80 ubuntu/apache2
```

#### Создать и запустить контейнер в интерактивном режиме

- создается новый контейнер на базе docker-образа «ubuntu/apache2» с именем «container-apach2»;
- в терминале появляется командная строка оболочки контейнера

```
sudo docker run -it --name "container-apache2" ubuntu/apache2 /bin/bash
```

## Запустить существующий контейнер:

```
sudo docker start container-apach2
```

### Установить автозапуск контейнера:

Использовать опцию —restart policies, где параметр policies может принимать следующие значения:

- **no** не перезапускать контейнер автоматически. (по умолчанию)
- on-failure перезапускать контейнер, если он завершает работу из-за ошибки, которая проявляется в виде ненулевого кода выхода.
- always всегда перезапускать контейнер, если он остановлен. Если он остановлен вручную, он перезапускается только при перезапуске демона Docker или перезапуске самого контейнера вручную.
- unless-stopped аналогично always, за исключением того, что когда контейнер остановлен (вручную или иным образом), он не перезапускается даже после перезапуска демона Docker.
- Первый запуск контейнера:

```
sudo docker run -d --restart unless-stopped <image>
```

• Созданные ранее контейнеры:

```
sudo docker update --restart unless-stopped <container>
```

#### Выполнить команду внутри контейнера:

• запуск команды «ls -l /» в контейнере с именем «container-apache2» без интерактивного режима:

```
sudo docker exec container-apache2 ls -l /
```

• запуск интерактивного терминала внутри контейнера с именем «container-apache2»

sudo docker exec -it container-apache2 /bin/bash

#### Перезапустить контейнер:

```
sudo docker restart %container name%
```

**Остановить контейнер.** Процесс в контейнере получает сигнал SIGTERM и через некоторое время SIGKILL. Остановка контейнера эквивалентна штатному выключению системы при обработке процессом сигнала SIGTERM и отключению питания, если процесс получает сигнал SIGKILL.

sudo docker stop container-apache2

## Удалить контейнер:

• удаление происходит без подтверждения, контейнер должен быть остановлен:

```
sudo docker rm "container-apache2"
```

• удалить все контейнеры со статусом «Exited»

```
sudo docker rm $(sudo docker ps --filter status=exited -q)
```

### Переименовать контейнер

```
sudo docker rename tst-apache2 container-apache2
```

#### Просмотр логов

```
sudo docker logs --follow %container_name%
```

## **Toma Docker**

**Docker том** — это каталог файловой системы хост-машины, который монтируется к файловой системе контейнера для обеспечения сохранения информации после, например, удаления контейнера.

Расположены тома в хостовой файловой системе в каталоге:

```
/var/lib/docker/volumes/%name%
```

#### Создание тома

• из командной строки:

```
sudo docker volume create --name %volume_name%
```

• из Dockerfile'a

```
VOLUME /var/lib/mysql
```

При этом создаётся новый том с именем из 64-х символов, данные в который копируются из каталога хост-машины /var/lib/mysql.

 $\frac{\mathsf{upuale.}}{2024/02/13} \ itechnology: web\_develop: environment \ https://jurik-phys.net/itechnology: web\_develop: environment$ 

14:15

#### Удаление тома

• подтверждения операции не будет:

```
sudo docker volume rm %vol_name%
```

#### Просмотр информации:

• СПИСОК ТОМОВ:

```
sudo docker volume ls
```

• подробности о томе:

```
sudo docker volume inspect %volume name%
```

#### Монтировние тома

• Для монтирования служит параметр -mount:

```
sudo docker run --mount src=%vol_name%,dest=%/path/in/container% %image_name%
# or
sudo docker run -v %vol name%:%/path/in/container% %image name%
```

Часто используемые параметры -mount или -v:

- type тип монтирования (bind, volume или tmpfs);
- src источник монтирования (имя тома или пусть файловой системы);
- dst путь, к которому файл или папка монтируется в контейнере;
- readonly монтирует том, который предназначен только для чтения.

## Очистка данных

#### Удаление неиспользуемых сетей

```
sudo docker network prune
```

#### Удаление всех неиспользуемых объектов:

- sudo docker system prune
- по умолчанию тома не удаляются:

```
sudo docker system prune --volumes
```

## GUI для управления контейнерами

- Docker Desktop с крупным набором нюансов при работе в Linux (ссылка);
- Portainer:
- Dockstation.

## **Dockerfile**

## Создание образа

Сценарий создания собственного Docker образа описывается текстовым файлом с именем «Dockerfile».

• Создание рабочего каталога, содержащего Dockerfile:

```
mkdir -p ~/docker/project_name
```

• Переход в созданный каталог:

```
cd ~/docker/project_name
```

• Создание Dockerfile следующего содержания:

```
vim Dockerfile
```

```
# Название базового образа
FROM debian:12.5

# Скопировать содержимое текущего каталога
# в каталог /opt/project_name oбраза
COPY . /opt/project_name/

# Текущий каталог проекта
WORKDIR /opt/project_name/

# Порт используемый приложением внутри контейнера
EXPOSE 7500

# Запуск команды при запуске контейнера.
CMD /opt/project_name/project.elf
```

Более подробное описание инструкций доступно по ссылке.

• Сборка образа

```
sudo docker build -t project_nave:version
...
...
Successfully built 53315083d9f8
Successfully tagged project_name:version
```

14:15

 $\frac{\mathsf{upuale.}}{2024/02/13} \ itechnology: web\_develop: environment \ https://jurik-phys.net/itechnology: web\_develop: environment$ 

• Просмотр списка образов:

```
sudo docker images
```

• Запуск контейнера на базе созданного образа:

```
sudo docker run -d -p 7500:7575 project name:version
```

Здесь 7500 - порт приложения внутри контейнера, 7575 - порт приложения для внешнего подключения т.е., вне контейнера.

• Для отладки можно запустить командную оболочку контейнера:

```
sudo docker run -it project name:version /bin/bash
```

## Загрузка образа на Docker Hub

• Для загрузки образа на Docker Hub необходимо авторизоваться:

```
docker login --username user name
```

• Сформировать правильный тег образа в виде <user\_name>/<project>:<version>, который позволит осуществить загрузку в требуемый аккаунт

```
sudo docker tag project name:version user name/project name:version
```

- На сайте https://hub.docker.com/ в разделе Repositories должен появиться загруженный образ
- Теперь его можно везде загрузить и запустить:

```
sudo docker pull user_name/project_name:version
sudo docker run --name your-container-name -d -p xxxx:yyyy
user name/project name:version
```

хххх - внутренний порт контейнера, уууу - внешний порт.

#### From:

https://jurik-phys.net/ - Jurik-Phys.Net

Permanent link:

https://jurik-phys.net/itechnology:web develop:environment

Last update: 2024/02/13 14:15

