Дата последней сборки документа:
16 мая 2020 г. в 23:55

Практика использования и наиболее полезные конструкции Docker

Подвойский А.О.

Содержание

1. Общие сведения о системе Docker

1.1. Установка

Установить Docker можно с помощью менеджера пакетов conda

```
conda install -c conda-forge docker-py
```

На текущий момент без серьезных проблем Docker работает только на 64-битовом Linux.

Для нормальной работы на MacOS X или Windows потребуется дополнительно установить какую-либо виртуальную машину в полной комплектации или пакет Docker Toolbox:

- для MacOS X: https://docs.docker.com/toolbox/toolbox_install_mac/,
- для Windows[1]: https://docs.docker.com/toolbox/toolbox_install_windows/; после установки Docker Toolbox останется только запустить Docker Quick Start Terminal.

[1] Поддерживается даже Windows 7

1.2. Контейнеры

Контейнеры представляют собой средства инкапсуляции приложения вместе со всеми его зависимостями.

Поскольку `Docker` сам по себе не обеспечивает реализацию любого типа виртуализации, контейнеры всегда должны соответствовать ядру хоста – контейнер на `Windows Server` может работать только на хосте под управлением операционной системы `Windows Server`, а 64-битный `Linux`-контейнер работает только на хосте с установленной 64-битной версией операционной системы `Linux` [1].

1.3. Создание образов из Dockerfile

`Dockerfile` – это обычный текстовый файл, содержащий набор операций, которые могут быть использованы для создания Docker-образа.

Пример. Для начала создадим новый каталог и собственно Dockerfile

```
$ mkdir cowsay
$ cd cowsay
$ touch Dockerfile
```

Затем в созданный `Dockerfile` добавим следующее

Dockerfile

```
FROM debian:wheezy
MAINTAINER John Smith <john@smith.com>
RUN apt-get update && apt-get install -y cowsay fortune
COPY entrypoint.sh /
ENTRYPOINT ["/entrypoint.sh"]
```

Инструкция `FROM` определяет базовый образ ОС (это в данном случае `debian` с уточненной версией «wheezy»). Инструкция `FROM` является строго обязательной для всех файлов Dockerfile как самая первая незакомментированная инструкция.

Инструкция `MAINTAINER` просто определяет информацию, позволяющую связаться с автором образа.

Инструкция `COPY` копирует файл из файловой системы хоста в файловую систему образа, где первый аргумент определяет файл хост, а второй – целевой путь.

Инструкция `RUN` определяет команды, выполняемые в командной оболочке внутри данного образа.

Комментарии к скрипту `entrypoint.sh`. Файл `entrypoint.sh` должен лежать в той же директории, что и файл `Dockerfile` и иметь содержание на подобие следующего

entrypoint.sh

```
if [ $# -eq 0 ]; then
    /usr/games/fortune | /usr/games/cowsay
else
    /usr/games/cowsay "$@"
fi
```

Здесь конструкция `[...]` – это форма[2] команды `test` для проверки различных условий. Последовательность символов `$#` – встроенная переменная, обозначающая количество аргументов

[2]Есть еще вариант `test выражение`, но форма `[ выражение ]` более популярна

в командной строке. Последовательность символов `$@` – это все аргументы командной строки, а `"$@"` – все аргументы командной строки, заключенные по отдельности в кавычки [2, стр. 44].

После сохранения необходимо сделать этот файл исполняемым при помощи команды `chmod +x entrypoint.sh`.

Теперь можно создать образ на основе файла `Dockerfile`

```
docker build -t test/cowsay-dockerfile .
```

Здесь `test` – имя репозитория, а `cowsay-dockerfile` – имя образа.

После этого можно запускать контейнер, который строится на основе образа `test/cowsay-dockerfile`

```
docker run test/cowsay-dockerfile /usr/games/cowsay 'Moo'
```

2. Общие сведения о компьютерных сетях

2.1. Термины и определения

localhost (так называемый, «локальный хост», по смыслу «этот компьютер») – стандартное, официально зарезервированное доменное имя для *частых* (или что то же самое *локальных*) IP-адресов[3] *петлевого интерфейса*[4] (диапазон `127.0.0.1` – `127.255.255.255`). Использование IP-адреса `127.0.0.1` позволяет устанавливать соединение и передавать информацию для программ-серверов, работающих на том же компьютере, что и программа-клиент. Примером может быть запущенный на компьютере веб-сервер приложений, обращение к которому выполняется с этого же компьютера для веб-разработки на данном компьютере без необходимости выкладывать веб-программу в сеть Интернет, пока ее разработка не закончена. Традиционно IP-адресу `127.0.0.1` однозначно сопоставляется имя хоста `localhost`.

порт целое неотрицательное число, записываемое в заголовках *протоколов транспортного уровня* модели OSI (TCP, UDP, SCTP, DCCP). Используется для определения процесса-получателя пакета в пределах одного хоста (локального компьютера).

3. Базовые концепции

3.1. Виртуальный сетевой интерфейс

Все TCP/IP-реализации поддерживают loopback-механизмы, которые реализуют виртуальный сетевой интерфейс исключительно программно и не связанны с каким-либо оборудованием, но при этом полностью интегрированны во внутреннюю сетевую инфраструктуру компьютерной системы. Пожалуй самый распространенным IP-адресом в механизмах loopback является `127.0.0.1`. В IPv4 в него также отображается любой адрес из диапазона `127.0.0.0` – `127.255.255.255`. IPv6 определяет единственные адрес для этой функции – `0:0:0:0:0:0:0:1/128` (так же записывается как `::1/128`). Стандартное, официально зарезервированное доменное имя для этих адресов – `localhost`.

[3] Уникальный сетевой адрес узла в компьютерной сети, построенной на базе стека протоколов TCP/IP

[4] Обычно используется термин *loopback*, который описывает методы или процедуры маршрутизации электронных сигналов, цифровых потоков данных, или других движущихся сущностей от их источника и обратно к тому же источнику без специальной обработки или модификации

Интерфейс loopback имеет несколько путей применения. Он может быть использован сетевым клиентским программным обеспечением, чтобы общаться с серверным приложением, расположенным на том компьютере. То есть если на компьютере, на котором запущен веб-сервер, указать в веб-браузере URL `http://127.0.0.1/` или `http:/localhost/`, то он попадает на вебсайт этого компьютера.

4. Наиболее полезные конструкции

4.1. Манипуляции с контейнерами

Запустить контейнер с именем `leorcont`, создав сеанс интерактивной работы (`-i`) на подключаемом терминальном устройстве (`-t`) tty, и вызывать командную оболочку `bash` из-под ОС `Ubuntu Linux`

```
docker run -it --name leorcont ubuntu bash
```

Запустить контейнер, а после остановки удалить сам контейнер и созданную на время его существования файловую систему

```
docker run --rm -it ubuntu bash
```

Перезапустить остановленный контейнер

```
docker start quizzical_wright
```

4.2. Информация о контейнере

Получить информацию о контейнере

```
docker inspect quizzical_wright
```

Вывести информацию о контейнере с использованием утилиты `grep`

```
docker inspect quizzical_wright | grep SandboxID
```

Вывести информацию о контейнере с использованием шаблона языка `Go` https://metanit.com/go/web/2.2.php

```
docker inspect --format {{.NetworkSettings.SandboxID}} quizzical_wright
```

Вывести список файлов в работающем контейнере. Для контейнеров `Docker` использует файловую систему `UnionFS`, которая позволяет монтировать несколько файловых систем в общую иерархию, которая выглядит как *единая файловая система*. Файловая система конкретного *образа* смонтирована как уровень *только для чтения*, а любые изменения в работающем контейнере происходят на уровне с разрешенной записью, монтируемого поверх основной файловой системы образа. Поэтому `Docker` при поиске изменений в работающей системе должен рассматривать только самый верхний уровень, на котором возможна запись [1]

```
docker diff quizzical_wright
```

Вывести список работающих контейнеров

```
docker ps
```

Вывести список всех контейнеров, включая остановленные (stopped)[5]. Такие контейнеры могут быть перезапущены с помощью `docker start`

```
docker ps -a
```

4.3. Удаление контейнеров и образов

Удалить контейнер

```
docker rm quizzical_wright
```

Удалить несколько остановленных контейнеров можно следующим способом. Значение флагов: `-a` (все контейнеры), `-q` (вывести только числовой идентификатор контейнера), `-f` (фильтр), `-v` (все тома, на которые не ссылаются какие-либо другие контейнеры)

```
docker rm -v $(docker ps -aq -f status=exited)
```

Список литературы

1. *Моуэт Э.* Использование Docker. – М.: ДМК Пресс, 2017. – 354 с.
2. *Роббинс А.* Bash. Карманный справочник системного администратора, 2-е изд.: Пер. с англ. – СПб.: ООО «Альфа-книга», 2017. – 152 с.

[5] Формально их называют контейнерами, из которых был совершен выход (exited containers)