

Docker, демонстрация

Юрий Литвинов

yurii.litvinov@gmail.com

16.10.2018

1. Требования

Для начала надо поставить Docker, либо как написано здесь: <https://docs.docker.com/engine/installation/linux/docker-ce/ubuntu/>, либо, если у вас Windows, просто скачав и поставив Docker Community Edition. Правда, под Windows Docker включит Hyper V, после чего перестанут работать все остальные виртуальные машины типа VirtualBox, поэтому если VirtualBox нужен, лучше ставить Docker Toolbox, который то же самое, но использует VirtualBox для виртуализации. Hyper V, впрочем, можно потом отключить. При установке компьютер захочет перезагрузиться (возможно, пару раз).

2. Docker

Docker — в общем-то, промышленный стандарт в области Continuous Delivery. Идея Docker заключается в том, чтобы вообще не заниматься установкой приложения и всех его зависимостей на целевой системе, а просто таскать всё с собой, включая все необходимые библиотеки, настройки и т.д. При этом приложение изолировано от всей остальной системы, так что конфликты версий библиотек или файлов конфигурации невозможны в принципе. Docker очень похож на виртуальные машины, но, в отличие от них, не эмулирует аппаратное обеспечение, а предоставляет абстракцию уровня ядра операционной системы. Так что несколько докер-контейнеров могут использовать одно ядро, нет нужды грузить “гостевую” ОС (потому что нет гостевой ОС, есть одна ОС и “гостевые” окружения), так что докер-контейнеры работают быстрее и весят меньше, чем виртуальные машины (зато не могут эмулировать несуществующее железо, как умеют многие виртуальные машины).

Кстати, *образ* (image) — это само приложение плюс все библиотеки, переменные окружения и конфиги, которые нужны ему для работы, а *контейнер* (container) — это результат запуска образа на выполнение (то есть экземпляр образа в рантайме).

Чтобы попробовать, поставьте докер и наберите в командной строке `docker run hello-world`. Docker не найдёт образ hello-world на локальной машине, полезет на Docker Hub и скачает образ оттуда, после чего запустит контейнер и выведет сообщение. Всё это практически мгновенно. На линуксе оно захочет `sudo`, но если добавить своего юзера в группу docker, то можно и без `sudo`.

Повторим tutorial с официального сайта (<https://docs.docker.com/get-started>), чтобы посмотреть на докер своими глазами. Там создаётся приложение на питоне, которое будет работать в контейнере. Причём, магия докера такова, что питон для этого ставить не надо.

Создадим пустую папку, создадим в ней файл Dockerfile вот такого содержания:

```
# Use an official Python runtime as a parent image
FROM python:2.7-slim

# Set the working directory to /app
WORKDIR /app

# Copy the current directory contents into the container at /app
ADD . /app

# Install any needed packages specified in requirements.txt
RUN pip install --trusted-host pypi.python.org -r requirements.txt

# Make port 80 available to the world outside this container
EXPOSE 80

# Define environment variable
ENV NAME World

# Run app.py when the container launches
CMD ["python", "app.py"]
```

Создаём там же requirements.txt такого содержания:

```
Flask
Redis
```

Ну и, собственно, приложение, app.py:

```
from flask import Flask
from redis import Redis, RedisError
import os
import socket

# Connect to Redis
redis = Redis(host="redis", db=0, socket_connect_timeout=2, socket_timeout=2)

app = Flask(__name__)

@app.route("/")
def hello():
    try:
        visits = redis.incr("counter")
    except RedisError:
```

```
visits = "<i>cannot connect to Redis, counter disabled</i>"
```

```
html = "<h3>Hello {name}!</h3>" \
      "<b>Hostname:</b> {hostname}<br/>" \
      "<b>Visits:</b> {visits}"
return html.format(name=os.getenv("NAME", "world"),
                   hostname=socket.gethostname(),
                   visits=visits)
```

```
if __name__ == "__main__":
    app.run(host='0.0.0.0', port=80)
```

Теперь соберём образ:

```
docker build -t friendlyhello .
```

Проверим, что оно собралось:

```
docker images
```

Должен вывести список всех докер-образов на машине, в том числе и свежесобранный.

Ну и, наконец, запустим контейнер:

```
docker run -p 4000:80 friendlyhello
```

Обратите внимание, что контейнер думает, что слушает порт 80, но мы замапили его на порт 4000 хостовой системы (-p — “publish”, опубликовать порт такой-то как такой-то). В докерфайле же прописано EXPOSE 80, что говорит контейнеру, что его (внутренний) порт 80 должен быть доступен извне.

Чтобы остановить контейнер, надо нажать Ctrl-C, как оно просит, но если бы всё было так просто, я бы про это не рассказывал. Под виндой надо остановить контейнер явно, командами

```
docker container ls
...
docker container stop ...
```

Где ... — либо CONTAINER ID, либо имя (автоматически сгенерированное) контейнера. На линуксе же, чтобы контейнер работал как демон, надо его запустить с ключём -d (detached).

Теперь давайте выложим это барахло в публичный доступ с помощью команд, очень похожих на git push (и вообще, работа с образами в докере сильно напоминает гит, там есть даже тэги). Делаем мы это для того, чтобы использовать докер-образы где-нибудь ещё, например, на Travis. Для начала надо завести аккаунт на <https://cloud.docker.com/>, запомнить своё имя пользователя. Логинимся из консоли командой docker login, вводим свой логин и пароль.

Дальше надо создать тэг для своего образа (не обязательно, на самом деле, но очень рекомендуется), например, так:

```
docker tag friendlyhello yuriilitvinov/get-started:part2
```

В тэге сначала идёт имя пользователя, потом имя репозитория, потом тэг текущего образа в репозитории. Можно посмотреть на то, что получилось, командой `docker image ls`. Теперь запустим вновь созданный тэг в репозиторий:

```
docker push yuriilitvinov/get-started:part2
```

Теперь можно пойти на <https://cloud.docker.com/> и посмотреть, что там появился новый репозиторий с нашим образом. Теперь на **любой** машине достаточно набрать

```
docker run -p 4000:80 yuriilitvinov/get-started:part2
```

Правда, контейнеры с виндовыми приложениями так просто на линуксе не заработают, зато скорее всего линуксовые контейнеры будут нормально работать на винде.