УДК 004.65

Афанасьев Г.И.
кандидат технических наук
доцент кафедры «Системы обработки информации и управления»
Московский государственный технический университет им. Н.Э. Баумана
Россия, г. Москва
Абулкасимов М.М.
старший преподаватель кафедры «Системы обработки информации и управления»
Московский государственный технический университет им. Н.Э. Баумана
Россия, г. Москва
Белоногов И.Б.
старший преподаватель кафедры «Системы обработки информации и управления»
Московский государственный технический университет им. Н.Э. Баумана
Россия, г. Москва

МЕТОДИКА СОЗДАНИЯ DOCKER-ОБРАЗА POSTGRESQL В СРЕДЕ UBUNTU LINUX

Аннотация: *В статье приводится пошаговая практическая методика по созданию Docker-образа PostgreSQL в среде Ubuntu Linux. Приведены многочисленные реальные спецификации производимых операций. Отмечается нюансы создания Docker-образа, а так же пути разрешения возможных возникающих при этом проблем*

Ключевые слова: *Docker, PostgreSQL, Ubuntu Linux*

Abstract: *The article provides a step-by-step practical procedure for creating a PostgreSQL Docker image in Ubuntu Linux environment. Numerous*

real specifications of the operations are given. It is noted nuances of creating a Docker-image, as well as ways to resolve possible problems that arise in this case

Key words: *Docker, PostgreSQL, Ubuntu Linux*

Введение

Популярность Docker растёт быстрыми темпами и всё большее количество разработчиков используют эту технологию при разработке программного обеспечения. Docker - это инструмент, предоставляющий удобный интерфейс для работы с LXC (Linux Containers). С помощью Docker можно запускать процессы в изолированном окружении.

Официальные образы для PostgreSQL уже существует и доступны для скачивания из Docker hub. Данная статья нацелена на то, чтобы показать пошаговое создание собственного образа и некоторые особенности Docker.

В качестве операционной системы для исследования использовалась Ubuntu Linux с установленным Docker. Для того, чтобы запустить Docker на отличной от Linux ОС, можно использовать boot2docker.

Dockerfile

Для создания образа необходимо создать текстовый файл **Dockerfile** и использовать доступные команды и синтаксис, описывающие процесс сборки образа. В начале файла необходимо указать базовый образ, который будет использоваться и контактные данные создателя:

```
FROM ubuntu:14.04
MAINTAINER hello<hello@gmail.com>
```

В нашем случае в качестве базового образа используется **Ubuntu 14.04**. После этого необходимо добавить репозиторий пакетов **PostgreSQL** и открытый ключ **GnuPG**:

```
RUN apt-key adv --keyserver keyserver.ubuntu.com --recv-keys 89780AFCAA1A47F044F244A07FCC7D46ACCC4CF8
```

```
RUN echo “deb http://apt.postgresql.org/pub/repos/apt/ precise-pgdg main”> /etc/apt/sources.list.d/pgdg.list
```

Затем необходимо обновить пакеты, имеющиеся в Ubuntu и установить PostgreSQL:

```
RUN apt-get update && apt-get -y -q install python-software-properties software-properties-common \ && apt-get -y -q install postgresql-9.3 postgresql-client-9.3 postgresql-contrib-9.3
```

Устанавливаем версию 9.3 PostgreSQL, инструкция будет очень похожа на установку любой другой версии базы данных. Необходимо, чтобы команды **apt-get update** и **apt-get install** находились в одной строке RUN, иначе они будут рассмотрены Docker как два различных слоя. В этом случае при появлении обновленного пакета, его установка во время повторной сборки образа не сможет быть выполнена.

Далее необходимо переключиться на пользователя **postgres**, чтобы создать нового пользователя и создать новую базу данных:

```
USER postgres
RUN /etc/init.d/postgresql start \
    && psql --command “CREATE USER pguser WITH SUPERUSER PASSWORD 'pguser';” \
    && createdb -0 pguser pgdb
```

Следующим пунктом необходимо разрешить подключение к базе данных из сети. Для этого необходимо переключиться на пользователя **root** и выполнить функции добавления строк в конфигурацию:

```
USER root
RUN echo “host all all 0.0.0.0/0 md5” » /etc/postgresql/9.3/main/pg_hba.conf
RUN echo “listen_addresses=’*’” » /etc/postgresql/9.3/main/postgresql.conf
```

Для того, чтобы иметь возможность подключиться к контейнеру, необходимо объявить порт который будет слушать PostgreSQL:

```
EXPOSE 5432
```

Объявляем, какие данные и общие папки будут использоваться в дальнейшем:

```
RUN mkdir -p /var/run/postgresql && chown -R postgres /var/run/postgresql
VOLUME [“/etc/postgresql”, “ /var/log/postgresql”, “/var/lib/postgresql”]
```

Последней командой в **dockerfile** является выполнение команды запуска. Для запуска PostgreSQL необходимо переключиться на пользователя **postgres** и выполнить команду запуска:

```
USER postgres
CMD [“/user/lib/postgresql/9.3/bin/postgres”, “-D”,
“/var/lib/postges/9.3/main”,
“-c”, “config_file=/etc/postgresql/9.3/main/postgresql.conf”]
```

Сборка Docker-образа

Когда Dockerfile готов необходимо собрать образ перед запуском его в контейнер. При настройке тэга имени, необходимо использовать имя собственного docker.io hub-аккаунта (заменить hello):

```
docker build -rm=true -t hello/postgresql:9.3
```

После выполнения сборки, образ загрузиться на docker.io hub и станет доступным для использования.

Запуск PostgreSQL Docker контейнера

Для запуска контейнера после сборки образа необходимо использовать следующую команду:

```
docker run -i -t -p 5432:5432 andreagrandi/postgresql:9.3
```

Проверка работы PostgreSQL

Проверить запуск контейнера можно с помощью любого клиента. Самым простым является подключение через командную строку:

```
psql -h localhost -p 5432 -U pgusr -W pgdb
```

При получении запроса на ввод пароля, необходимо ввести следующую команду: **pguser**. Использование **localhost** возможно только при запуске Docker на локальной машине под Linux. Пользователям OSX необходимо узнать правильный IP с помощью: **boot2docker ip**.

Сохранение данных

Особенностью Docker является то, что после остановки контейнера происходит потеря некоторых данных, которые ранее записывались в БД. Это происходит потому, что по умолчанию существование Docker контейнеров ограничено во времени. Данную проблему можно решить с помощью контейнера данных. Рекомендуется не делать это вручную, а для организации работы использовать специальный инструмент – **fig**.

Fig является инструментом, организующим работу контейнеров, его функции переписаны на языке Go и интегрированы в Docker. Инструкции по установке можно найти на веб-сайте **Fig.sh** (кратко: пользователи Ubuntu должны использовать **pip install fig,** пользователи OSX должны использовать **brew install fig**).

Необходимо сохранить этот файл как fig.yml в той же папке где храниться Dockerfile и выполнить в терминале команду **fig up**:

```
dbdata:
        image: andreagrandi/postgresql:9.3
        volumes:
                    -/var/lib/postgresql
db:
        image:  image: andreagrandi/postgresql:9.3
        volumes_from:
        -dbdata
        ports:
        -“5432:5432”
```

```
hello:postgresql-docker hello [master] $ fig up
Recreating  postgresqldocker_dbdata_1...
Recreating  postgresqldocker_db_1...
Attaching to postgresqldocker_db_1

db_1 | 2017-11-27 19:01:07 UTC (6-1) LOG: database system was interrupted; last known up at 2017-11-27 17:46:10 UTC

db_1 | 2017-11-27 19:01:07 UTC (6-2) LOG: database system was not properly shut down; automatic recovery in progress

db_1 | 2017-11-27 19:01:07 UTC (6-3) LOG: redo starts at 0/1782F68

db_1 | 2017-11-27 19:01:07 UTC (6-4) LOG: record with zero length at 0/1782FA8
```

db_1 | 2017-11-27 19:01:07 UTC (6-5) LOG: redo done at 0/1782F68

db_1 | 2017-11-27 19:01:07 UTC (6-6) LOG: last completed transaction was at log time 2017-11-27 17:46:10.61746+00

db_1 | 2017-11-27 19:01:07 UTC (1-1) LOG: database system is ready to accept connections

db_1 | 2017-11-27 19:01:07 UTC (10-1) LOG: autovacuum launcher started

Теперь, после записи некоторых данных в базу данных, последующей остановки (CTRL + C) работающих контейнеров и затем повторного их запуска, данные все еще находятся на прежнем месте.

Заключение

Данная статья является примером последовательности действий для создания Docker-образа, использование PostgreSQL выбрано только в качестве примера. Показано как создать простой образ с базой данных и данными которые будут сохраняться после перезапуска контейнера, собрать этот образ и зарегистрировать его на Docker hub.

Некоторую сложность создания Docker-образов вызывают ситуации, при которых приходится запускать несколько сервисов (например, веб-приложение Django с использованием PostgreSQL, RabbitMQ, MongoDB и т.д …), соединять их воедино и организовывать полноценное решение. Для этого необходимо использовать другие функции Docker [3].

Использованные источники

1. Документация по Docker, [Электронный ресурс]. Режим доступа: https://docs.docker.com (дата обращения: 21.01.2017).
2. Документация PostgreSQL, [Электронный ресурс]. Режим доступа: https://www.postgresql.org/docs/ (дата обращения: 21.01.2017).
3. Jeff Nickoloff. Docker in Action. USA, Manning 2016, 304 p.