Dockerize PostgreSQL

Tiempo estimado de lectura: 5 minutos

> **Nota :**
>
> - **Si no le gusta sudo** , consulte *Dar acceso no root* (https://docs.docker.com/engine/installation/binaries/#giving-non-root-access)

Instalación de PostgreSQL en Docker

Asumiendo que no hay una imagen de Docker que se adapte a sus necesidades en el Hub Docker (http://hub.docker.com) , puede crear uno usted mismo.

Comience creando un nuevo `Dockerfile` :

> **Nota** : Esta configuración de PostgreSQL es para propósitos de sólo desarrollo. Consulte la documentación de PostgreSQL para afinar estas configuraciones para que sean adecuadamente seguras.

```
#
# example Dockerfile for https://docs.docker.com/examples/postgresql_service/
#

FROM ubuntu

# Add the PostgreSQL PGP key to verify their Debian packages.
# It should be the same key as https://www.postgresql.org/media/keys/ACCC4CF8.asc
RUN apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80 --recv-keys B97B0AFCAA1A47F044F244A07FCC7D46ACCC4CF8

# Add PostgreSQL's repository. It contains the most recent stable release
#     of PostgreSQL, ``9.3``.
RUN echo "deb http://apt.postgresql.org/pub/repos/apt/ precise-pgdg main" > /etc/apt/sources.list.d/pgdg.list

# Install ``python-software-properties``, ``software-properties-common`` and PostgreSQL 9.3
#  There are some warnings (in red) that show up during the build. You can hide
#  them by prefixing each apt-get statement with DEBIAN_FRONTEND=noninteractive
RUN apt-get update && apt-get install -y python-software-properties software-properties-common postgresql-9.3 postgresql-client

# Note: The official Debian and Ubuntu images automatically ``apt-get clean``
# after each ``apt-get``

# Run the rest of the commands as the ``postgres`` user created by the ``postgres-9.3`` package when it was ``apt-get installed``
USER postgres

# Create a PostgreSQL role named ``docker`` with ``docker`` as the password and
# then create a database `docker` owned by the ``docker`` role.
# Note: here we use ``&&\`` to run commands one after the other - the ``\``
#       allows the RUN command to span multiple lines.
RUN    /etc/init.d/postgresql start &&\
    psql --command "CREATE USER docker WITH SUPERUSER PASSWORD 'docker';" &&\
    createdb -O docker docker

# Adjust PostgreSQL configuration so that remote connections to the
# database are possible.
RUN echo "host all  all    0.0.0.0/0  md5" >> /etc/postgresql/9.3/main/pg_hba.conf

# And add ``listen_addresses`` to ``/etc/postgresql/9.3/main/postgresql.conf``
RUN echo "listen_addresses='*'" >> /etc/postgresql/9.3/main/postgresql.conf

# Expose the PostgreSQL port
EXPOSE 5432

# Add VOLUMEs to allow backup of config, logs and databases
VOLUME  ["/etc/postgresql", "/var/log/postgresql", "/var/lib/postgresql"]

# Set the default command to run when starting the container
CMD ["/usr/lib/postgresql/9.3/bin/postgres", "-D", "/var/lib/postgresql/9.3/main", "-c", "config_file=/etc/postgresql/9.3/main/
```

Construir una imagen desde el Dockerfile asignarle un nombre.

```
$ docker build -t eg_postgresql .
```

Ejecute el contenedor del servidor PostgreSQL (en primer plano):

```
$ docker run --rm -P --name pg_test eg_postgresql
```

Hay 2 formas de conectarse al servidor PostgreSQL. Podemos usar *Link Containers* (https://docs.docker.com/engine/userguide/networking/default_network/dockerlinks/) , o podemos acceder a él desde nuestro host (o la red).

> **Nota** : `--rm` Elimina el contenedor y su imagen cuando el contenedor sale correctamente.

Uso del enlace de contenedores

Los contenedores pueden ser enlazados a los puertos de otro contenedor directamente usando `-link remote_name:local_alias` en el cliente `docker run` . Esto establecerá un número de variables de entorno que luego se pueden utilizar para conectar:

```
$ docker run --rm -t -i --link pg_test:pg eg_postgresql bash

postgres@7ef98b1b7243:/$ psql -h $PG_PORT_5432_TCP_ADDR -p $PG_PORT_5432_TCP_PORT -d docker -U docker --password
```

Conexión desde el sistema host

Asumiendo que el cliente postgresql está instalado, también puede utilizar el puerto asignado al host para probar. Es necesario utilizar `docker ps` para averiguar qué puerto de host local el contenedor está asignado a primero:

```
$ docker ps

CONTAINER ID        IMAGE                  COMMAND                CREATED             STATUS              PORTS
5e24362f27f6        eg_postgresql:latest   /usr/lib/postgresql/   About an hour ago   Up About an hour    0.0.0.0:49153->5432/t

$ psql -h localhost -p 49153 -d docker -U docker --password
```

Prueba de la base de datos

Una vez que se haya autenticado y tenga un `docker =#` mensaje, puede crear una tabla y rellenarlo.

```
psql (9.3.1)
Type "help" for help.

$ docker=# CREATE TABLE cities (
docker(#     name            varchar(80),
docker(#     location        point
docker(# );
CREATE TABLE
$ docker=# INSERT INTO cities VALUES ('San Francisco', '(-194.0, 53.0)');
INSERT 0 1
$ docker=# select * from cities;
     name      | location
---------------+-----------
 San Francisco | (-194,53)
(1 row)
```

Uso de los volúmenes de contenedores

Puede utilizar los volúmenes definidos para inspeccionar los archivos de registro de PostgreSQL y para hacer una copia de seguridad de la configuración y los datos:

```
$ docker run --rm --volumes-from pg_test -t -i busybox sh

/ # ls
bin      etc      lib      linuxrc  mnt      proc     run      sys      usr
dev      home     lib64    media    opt      root     sbin     tmp      var
/ # ls /etc/postgresql/9.3/main/
environment      pg_hba.conf      postgresql.conf
pg_ctl.conf      pg_ident.conf    start.conf
/tmp # ls /var/log
ldconfig    postgresql
```

Docker (https://docs.docker.com/glossary/?term=docker) , ejemplo (https://docs.docker.com/glossary/?term=example) , instalación de paquetes (https://docs.docker.com/glossary/?term=package%20installation) , postgresql (https://docs.docker.com/glossary/?term=postgresql)