使用Docker安装自定义PostgreSQL(CentOS 7)

本文档使用环境：

VMware 15.5.0 build-14665864

CentOS-Linux-7-2009

Docker version 20.10.17, build 100c701

Docker Compose version v2.9.0

一、简略版

传输文件到本机（参考）

```
[root@master ~] scp docker_postgres_setup.tar.gz root@node1:/usr/local/src
```

如果没有安装docker：安装和部署postgresql（主要步骤，需要具有root权限）

```
# 1.进入src目录
[root@node1 ~] cd /usr/local/src
# 2.解压压缩包
[root@node1 src] tar -zxvf docker_postgres_setup.tar.gz
# 3.进入安装目录
[root@node1 src] cd docker_postgres_setup
# 4.执行安装脚本
[root@node1 docker_postgres_setup] ./setup.sh
# 5.切换到docker用户
[root@node1 docker_postgres_setup] su docker
默认密码是：docker
# 6.进入postgres安装子目录
[docker@node1 docker_postgres_setup]$ cd postgres_build
# 7.启动postgres容器
[docker@node1 postgres_build]$ docker-compose up -d
```

如果已经安装docker：安装和部署postgresql（主要步骤，需要具有root权限）（第四步不一样）

```
# 1.进入src目录
[root@node1 ~] cd /usr/local/src
# 2.解压压缩包
[root@node1 src] tar -zxvf docker_postgres_setup.tar.gz
# 3.进入安装目录
[root@node1 src] cd docker_postgres_setup
# 4.载入镜像文件
[root@node1 docker_postgres_setup] docker load -i postgres_conf-12.5.tar.gz
# 5.切换到docker用户
[root@node1 docker_postgres_setup] su docker
默认密码是：docker
# 6.进入postgres安装子目录
```

```
[docker@node1 docker_postgres_setup]$ cd postgres_build
# 7.启动postgres容器
[docker@node1 postgres_build]$ docker-compose up -d
```

连接到postgresql（从外部访问容器数据库）

数据库用户名：postgres 数据库用户密码：postgres 初始数据库：postgres

数据库连接地址：docker 本机 IP 地址（在本例中是 192.168.182.132，仅供参考）

数据库连接端口：5432

二、准备文件

下载并解压安装压缩文件，得到安装目录

```
# 进入src目录
[root@node1 ~] cd /usr/local/src
# 查看压缩包
[root@node1 src] ls
docker_postgres_setup.tar.gz
# 解压压缩包
[root@node1 src] tar -zxvf docker_postgres_setup.tar.gz
# 进入安装目录
[root@node1 src] cd docker_postgres_setup
# 查看安装目录
[root@node1 docker_postgres_setup] ls
dockerrpm
postgres_build
postgres_conf-12.5.tar.gz
setup.sh
```

三、使用脚本进行安装

执行 setup.sh 脚本即可安装

```
[root@node1 docker_postgres_setup] ./setup.sh
```

控制台的输出（仅供参考）

如果报错的话，需要使用 vim setup.sh 查看脚本里面的命令，然后手动执行并排查错误

```
---------------------------------
Installing docker...
---------------------------------
warning: dockerrpm/containerd.io-1.6.6-3.1.el7.x86_64.rpm: Header V4 RSA/SHA512
Signature, key ID 621e9f35: NOKEY
Preparing...                          ################################# [100%]
Updating / installing...
   1:libseccomp-2.3.1-4.el7           ################################# [  6%]
   2:docker-scan-plugin-0:0.17.0-3.el7################################# [ 12%]
   3:docker-ce-cli-1:20.10.17-3.el7   ################################# [ 18%]
   4:libcgroup-0.41-21.el7            ################################# [ 24%]
   5:slirp4netns-0.4.3-4.el7_8        ################################# [ 29%]
   6:setools-libs-3.3.8-4.el7         ################################# [ 35%]
```

```
   7:python-IPy-0.75-6.el7            ################################# [ 41%]
   8:libsemanage-python-2.5-14.el7    ################################# [ 47%]
   9:fuse3-libs-3.6.1-4.el7           ################################# [ 53%]
  10:fuse-overlayfs-0.7.2-6.el7_8     ################################# [ 59%]
  11:checkpolicy-2.5-8.el7            ################################# [ 65%]
  12:audit-libs-python-2.8.5-4.el7    ################################# [ 71%]
  13:policycoreutils-python-2.5-34.el7################################# [ 76%]
  14:container-selinux-2:2.119.2-1.911################################# [ 82%]
  15:containerd.io-1.6.6-3.1.el7      ################################# [ 88%]
  16:docker-ce-rootless-extras-0:20.10################################# [ 94%]
  17:docker-ce-3:20.10.17-3.el7       ################################# [100%]
groupadd: group 'docker' already exists
Created symlink from /etc/systemd/system/multi-user.target.wants/docker.service
to /usr/lib/systemd/system/docker.service.
---------------------------------
docker and docker-compose version
---------------------------------
Docker version 20.10.17, build 100c701
Docker Compose version v2.9.0
---------------------------------
Loading image...
---------------------------------
9eb82f04c782: Loading layer  72.49MB/72.49MB
# 中间略
6ca858a23f56: Loading layer  32.26kB/32.26kB
Loaded image: postgres_conf:12.5
---------------------------------
Creating file path...
---------------------------------
ls -l /data/docker
total 0
drwxr-xr-x. 2 polkitd input 6 Aug 10 15:16 pgdata
drwxr-xr-x. 2 polkitd input 6 Aug 10 15:16 pglog
```

四、使用docker（可选）

通过脚本，创建了一个名为docker的用户，建议使用docker用户来进行操作，以免出现误操作问题

```
# 使用su切换到docker用户
[root@node1 docker_postgres_setup] su docker
```

通过hello-world了解docker的一些基础指令（可选）

```
# 运行简单的hello-world示例
# 在本地中没有hello-world镜像，会从远程仓库下载，可能会比较慢
[docker@node1 docker_postgres_setup]$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
2db29710123e: Pull complete
Digest: sha256:53f1bbee2f52c39e41682ee1d388285290c5c8a76cc92b42687eecf38e0af3f0
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/
```

运行 docker run hello-world，docker首先会下载名称和标签为 hello-world : latest 的镜像文件，镜像 ID 为 feb5d9fea6a5。接下来，会根据镜像文件创建一个容器，ID为 b90d35f3c038，名称为 wonderful_poitras（自动生成的命名）。然后使用 "/hello" 命令执行该容器，输出文本到控制台中

```
# 查看本地的docker镜像文件
[docker@node1 docker_postgres_setup]$ docker images
REPOSITORY      TAG       IMAGE ID       CREATED          SIZE
postgres_conf   12.5      7ec31b241cc2   5 hours ago      314MB
hello-world     latest    feb5d9fea6a5   10 months ago    13.3kB
# 查看所有状态下的容器
[docker@node1 docker_postgres_setup]$ docker ps -a
CONTAINER ID   IMAGE         COMMAND    CREATED         STATUS
  PORTS     NAMES
b90d35f3c038   hello-world   "/hello"   3 minutes ago   Exited (0) 3 minutes ago
            wonderful_poitras
```

```
# 根据id或者名字删除容器
[docker@node1 docker_postgres_setup]$ docker rm feb5d9fea6a5
或者
[docker@node1 docker_postgres_setup]$ docker rm wonderful_poitras
# 根据id或者仓库:tag删除镜像
[docker@node1 docker_postgres_setup]$ docker rmi feb5d9fea6a5
或者
[docker@node1 docker_postgres_setup]$ docker rmi hello-world:latest
```

五、安装PostgreSQL

在 postgres_build 目录下有四个文件

其中 docker-compose.yml 用于从镜像 postgres_conf : 12.5 创建和启动容器

另外的 Dockerfile、postgresql.conf、updateConfig.sh 用于创建自定义镜像，可以忽略

如果只要部署 pgsql，只需要用到 docker-compose.yml

```
# 进入到postgres安装子目录
[docker@node1 docker_postgres_setup]$ cd postgres_build/
# 查看目录
[docker@node1 postgres_build]$ ll 或者 ls -l
total 40
-rwxrwxr-x. 1 docker docker   366 Aug 10 16:00 docker-compose.yml
-rwxrwxr-x. 1 docker docker   260 Aug 10 11:55 Dockerfile
-rwxrwxr-x. 1 docker docker 26675 Aug 10 11:55 postgresql.conf
-rwxrwxr-x. 1 docker docker   109 Aug 10 11:55 updateConfig.sh
```

使用 docker-compose up 可以一键部署容器，配置都在 yml 文件中写好了

```
[root@node1 postgres_build] docker-compose up
[+] Running 2/2
 ⠿ Network postgres_build_default  Created
             0.6s
 ⠿ Container postgres              Created
             0.6s
Attaching to postgres
postgres  | The files belonging to this database system will be owned by user
"postgres".
postgres  | This user must also own the server process.
# 中间略
postgres  | PostgreSQL init process complete; ready for start up.
postgres  |
postgres  | 2022-08-10 16:07:48.814 CST [1] LOG:  starting PostgreSQL 12.5
(Debian 12.5-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by gcc (Debian 8.3.0-
6) 8.3.0, 64-bit
postgres  | 2022-08-10 16:07:48.815 CST [1] LOG:  listening on IPv4 address
"0.0.0.0", port 5432
postgres  | 2022-08-10 16:07:48.815 CST [1] LOG:  listening on IPv6 address
"::", port 5432
postgres  | 2022-08-10 16:07:48.817 CST [1] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5432"
postgres  | 2022-08-10 16:07:48.835 CST [1] LOG:  redirecting log output to
logging collector process
postgres  | 2022-08-10 16:07:48.835 CST [1] HINT:  Future log output will appear
in directory "/var/log/pglog".

# 使用Ctrl+C退出
^CGracefully stopping... (press Ctrl+C again to force)
[+] Running 1/1
 ⠿ Container postgres  Stopped
             0.7s
canceled
```

启动容器后，控制台会处于 postgres 控制状态中，使用Ctrl+C退出的同时也会关闭容器

使用 docker ps -a 查看容器，可以看到 postgres 已经处于退出状态

```
[root@node1 postgres_build] docker ps -a
CONTAINER ID   IMAGE                COMMAND                  CREATED
STATUS                       PORTS     NAMES
58d7d16e3ad2   postgres_conf:12.5   "docker-entrypoint.s…"   10 minutes ago
Exited (0) 31 seconds ago              postgres
```

如果想在后台启动 postgres 服务，可以使用 -d 选项

此时用 docker ps 查看，可以看到容器处于运行状态，而且将 5432 端口进行了映射

如果要访问这个 pgsql 数据库，则使用 本机IP+5432端口 的方式来进行访问。例如在本文档中，虚拟机的 IP 是 192.168.182.132。那么使用笔记本的 Navicat 连接 192.168.182.132 和 5432 端口的数据库，用户名、密码、初始数据库都为 postgres，就可以连接到数据库内

```
[root@node1 postgres_build]# docker-compose up -d
[+] Running 1/1
 ⠿ Container postgres  Started
             0.7s
[root@node1 postgres_build]# docker ps
CONTAINER ID   IMAGE                COMMAND                  CREATED
STATUS              PORTS                                      NAMES
58d7d16e3ad2   postgres_conf:12.5   "docker-entrypoint.s…"   13 minutes ago   Up
About a minute   0.0.0.0:5432->5432/tcp, :::5432->5432/tcp   postgres
```

如果要启停容器，可以使用容器的名称或者 ID

```
# 启动
$ docker start postgres
或者
$ docker start 58d7d16e3ad2
# 停止
$ docker stop postgres
或者
$ docker stop 58d7d16e3ad2
```

如果要进入容器内部，则使用 docker exec 命令

```
# 通过交互方式进入postgres容器
[docker@node1 postgres_build]$ docker exec -it postgres bash
##### 进入容器内部的控制台 #####
root@843f7970c6b0:/# su postgres
postgres@843f7970c6b0:/$ psql
psql (12.5 (Debian 12.5-1.pgdg100+1))
Type "help" for help.

postgres=# select version();
                          version

------------------------------------------------------------------------
 PostgreSQL 12.5 (Debian 12.5-1.pgdg100+1) on x86_64-pc-linux-gnu, compiled by
gcc (Debian 8.3.0-6) 8.3.0, 64-bit
(1 row)

postgres=# \q
```

```
postgres@843f7970c6b0:/$ exit
exit
root@843f7970c6b0:/# exit
exit
##### 退出容器内部的控制台 #####
[docker@node1 postgres_build]$
```