链滴

MySQL 官方 Docker 镜像的使用

作者：jiangyong

原文链接：https://ld246.com/article/1495976290353

来源网站：链滴

Docker镜像是创建容器的基础，我们可以基于官方提供镜像或自己构建的镜像来创建容器，而自己建的镜像往往又基于基于官方基础镜像构建。mysql是Docker及MySQL提供、维护的一个官方镜像我们可以基于该镜像构建自己的MySQL数据库镜像，也可以直接使用这个镜像创建MySQL数据库容。在笔者所参与的项目中，MySQL数据库使用频率较高，因此对该镜像的使用做简单的整理、介绍。

1. 启动一个mysql服务器实例

使用mysql镜像创建或启动MySQL容器时，可以先将镜像下载到本地：

```
$ docker pull mysql
```

也可以直接使用以下命令来启动MySQL实例：

```
$ docker run --name itbilu-mysql -e MYSQL_ROOT_PASSWORD=my-pass -d mysql:5.7
```

这样，我们就创建了一个名为itbilu-mysql的MySQL数据库服务器容器实例。在创建数据库时，通过境变量MYSQL_ROOT_PASSWORD设置数据库的root密码，还通过5.7标签指定了所使用的镜像版。

容器创建完成后，可以通过docker ps命令看到所创建的MySQL容器实例：

```
$ docker ps
CONTAINER ID        IMAGE               COMMAND                  CREATED             STATUS
ORTS                NAMES
872e8133e7ac        mysql:5.7           "docker-entrypoint..."   15 seconds ago      Up 14 seconds
    3306/tcp            itbilu-mysql
```

2. 在其它Docker容器中应用中连接MySQL

在这个镜像中，导出的是MySQL的标准端口3306。这样我们就可以在需要访问MySQL服务器的容器，使用--link参数通过容器链接的方式，将MySQL服务器容器实例连接到包含了需要使用MySQL的用容器中。

使用容器连接的示例如下：

```
$ docker run --name some-app --link itbilu-mysql:mysql -d application-that-uses-mysql
```

注意：以上示例中的application-that-uses-镜像并不存在，仅为操作演示，下同。

除了进行容器连接的方式在其它容器中访问MySQL服务器容器外，还可以通过以下两种方式访问MySL数据库服务器容器：

在创建MySQL服务器容器实例时通过-p或-P参数将数据库服务器端口映射到宿主机，再直接通过宿机进行访问。这种方式较为简单，但需要向外暴露数据库端口。

能过Docker网络(Networking)进行连接。这种方式操作较为复杂，但更为灵活，可以适用于更加复的网络环境。

详细参考如下：

- Docker 网络-端口映射、容器链接、Networking

3. MySQL命令行客户端连接MySQL

在前面创建的MySQL服务器容器中，我们并没有向外暴露访问端口，我们可以通过以下方式启动命行客户端，并基于命令行客户端对数据库服务器进行管理。

运行另一个MySQL交互式容器，该容器会在运行后启动mysql命令行客户端：

```
$ docker run -it --link itbilu-mysql:mysql --rm mysql sh -c 'exec mysql -h"$MYSQL_PORT_330
_TCP_ADDR" -P"$MYSQL_PORT_3306_TCP_PORT" -uroot -p"$MYSQL_ENV_MYSQL_ROOT_PA
SWORD"'
```

如果需要通过非Docker的方式，或远程访问MySQL服务器容器，就可以在创建容器时通过-p或-P与主机进行端口绑定，之后就可以像普通MySQL服务器那样进行访问或操作。

4. 在Shell中访问容器及日志查看

docker exec命令使我们可以在Docker容器内部执行命令，我们可以通过以下方式与mysql容器建立个shell连接：

```
$ docker exec -it itbilu-mysql bash
root@cc9417196c73:/#
```

而MySQL服务器日志，可以直接通过Docker容器日志访问（实时日志查看可以添加-f参数）：

```
$ docker logs itbilu-mysql
```

5. 使用自定义MySQL配置文件

默认情况下，MySQL的启动配置文件是/etc/mysql/my.cnf，而/etc/mysql/conf.d目录下的存在任何cnf格式的文件时，都会使用该文件中配置项替换默认配置。

因此，如果要使用自定义配置，可以在宿主机创建一个配置文件，然后在创建容器时通过-v参数，以据卷的方式将自定义配置挂载到mysql容器的/etc/mysql/conf.d目录下。

如，在宿主机中存在/my/custom/config-file.cnf配置文件，这时就可以通过以下方式启动MySQL容：

```
$ docker run --name itbilu-mysql -v /my/custom:/etc/mysql/conf.d -e MYSQL_ROOT_PASSW
RD=my-pass -d mysql:5.7
```

以上示例会启动一个名为itbilu-mysql的MySQL服务器容器，该文件启动时会同时使用/etc/mysql/mycnf及/etc/mysql/conf.d/config-file.cnf中的配置。

不使用cnf文件的配置方式

除使用.cnf文件进行配置外，还可以在启动容器通过参数的形式将配置传递给mysqld。

如，启动一个MySQL服务器容器，并使用UTF-8(utf8mb4)格式的表编码：

```
$ docker run --name itbilu-mysql -e MYSQL_ROOT_PASSWORD=my-secret-pw -d mysql:tag -
character-set-server=utf8mb4 --collation-server=utf8mb4_unicode_ci
```

详细配置参数可以通过以下命令查看：

```
$ docker run -it --rm mysql:tag --verbose --help
```

6. 环境变量

当启动mysql容器时，我们可以向docker run命令传入一或多个环境变量来调整MySQL实例的配置
可设置的环境变量有：

- MYSQL_ROOT_PASSWORD：必须。用于设置MySQLroot用户的密码
- MYSQL_DATABASE：可选。用于指定镜像启动容器时要创建的数据库。如果提供了用户/密码，会将该用户做为此数据库的超级用户。
- MYSQL_USER，MYSQL_PASSWORD：可选。用于创建一个新用户并设置密码。
- MYSQL_ALLOW_EMPTY_PASSWORD：可选。设置为yes时，则可以使用空密码登录
- MYSQL_RANDOM_ROOT_PASSWORD：可选。设置为yes时会为root用户设置一个随机密码（用pwgen），所生成的随机密码会被输出到stdout
- MYSQL_ONETIME_PASSWORD：可选。为root用户指定一个一次性密码，该密码会在用户首次录时强制修改

7. 关于数据存储

在使用　mysql镜像创建MySQL容器时，数据库数据存储可能会有以下两种方式：

数据卷容器：使用Docker默认的数据管理方式来管理数据库的数据存储，在这种方式下，数据库文会被写入数据库的内部。这种方式对于用户非常简单，缺点是很在宿主机上找到所存储的数据。

外部数据卷：在宿主机创建一个数据目录，再将数据目录挂载到容器内部。这种方式可以很方便的在主机上找到并进行数据管理，但需要确保数据目录的存在。

当使用外部数据卷时，假在宿主机有/my/own/datadir目录，我们可以像下面这样启动mysql容器，将目录挂载到容器内：

```
$ docker run --name itbilu-mysql -v /my/own/datadir:/var/lib/mysql -e MYSQL_ROOT_PASS
ORD=my-secret-pw -d mysql:tag
```

其中，-v /my/own/datadir:/var/lib/mysql是数据卷的挂载，表示将宿主机的/my/own/datadir目挂载到容器内的/var/lib/mysql目录。这个目录是MySQL的默认数据目录，当使用自定义配置时，应也做相应的修改。

数据库备份

在非Docker环境下使用的MySQL备份工具，大多数在容器环境下仍然可用，只要其能访问mysqld服器即可。

下面是一个通过docker exec来对mysql容器中的数据库执行备份的示例：

```
$ docker exec some-mysql sh -c 'exec mysqldump --all-databases -uroot -p"$MYSQL_ROOT_
ASSWORD"' > /some/path/on/your/host/all-databases.sql
```