持久化 - volume

本篇是第五部分“存储篇”的第一篇，在这个部分，我将用三篇内容为你介绍 Docker 存储相关的内容，以及深入理解 Overlay2 存储驱动的工作原理。

通过前面“容器篇”和“镜像篇”的介绍，我们知道 Docker 容器可以近似理解为镜像的运行时实例，默认情况下也算是在镜像层的基础上增加了一个可写层。

所以，一般情况下如果你在容器内做出的修改，均包含在这个可写层中。

当容器被删除后，这些修改也就丢失了，因此有人会使用 `docker commit` 命令将在容器内做出的修改保存到新镜像中。

同时，由于容器间彼此独立，数据共享也是一个问题。如果想要传递数据（文件），一般都会使用 `docker cp` 等功能，完成数据的复制。

基于以上提到的情况，Docker 给我们提供了两种更简便的解决方案： **volume** 和 bind mount。其中的 volume 便是本篇的重点。

volume 简介

在正式开始介绍 Docker volume 的内容前，需要再次明确一下 volume 和 bind mount 的区别。

复制

```
(MoeLove) ➜  ~ docker run --help |grep -E 'volume|mount'
      --mount mount                    Attach a filesystem mount to the container
  -v, --volume list                    Bind mount a volume
      --volume-driver string           Optional volume driver for the container
      --volumes-from list              Mount volumes from the specified container(
```

我们从 `docker run` 支持的参数入手，可以看到主要有两种形式：

- --mount
- --volume

这两种形式功能上略有差别，整体而言，我更推荐使用 `--mount`，虽然 `--volume` 或者 `-v` 形式上比较简单。

对于 volume 和 bind mount 的差别，主要是以下方面：

1. bind mount 是指从主机上挂载文件或者目录至容器中，这些文件或目录是独立于容器或者 Docker 之外的，不受其管理。

例如：

复制

```
(MoeLove) ➜  ~ docker run --rm  -it --mount "type=bind,src=/tmp/k,target=/k"
/ # ls -al /k
total 12
drwxrwxr-x    2 1000     1000            60 Jan 20 15:02 .
drwxr-xr-x    1 root     root          4096 Jan 20 18:25 ..
-rw-rw-r--    1 1000     1000             4 Jan 20 15:02 xxxx.txt
```

2\. volume 则是由 Docker 进行管理的，其数据存储于 Docker 的文件夹下面，默认的位置在

例如：

复制

```
(MoeLove) ➜  ~ docker run --rm  -it --mount "type=volume,src=k-volume,target=/k"
/ # ls -al /k/
total 12
drwxr-xr-x    2 root     root          4096 Jan 20 18:29 .
drwxr-xr-x    1 root     root          4096 Jan 20 18:29 ..
(MoeLove) ➜  ~ docker volume ls
DRIVER              VOLUME NAME
local               k-volume
```

本篇主要聊的内容是使用 Docker volume，这种情况下无论使用 `-v` 或者 `--mount` 选项均可。

volume 的生命周期

了解到了 volume 的主要概念后，我们来看看 volume 的生命周期管理。

增

最简单的办法是使用 `docker volume create` 创建一个具有名字的 volume，例如：

复制

```
(MoeLove) ➜  ~ docker volume create test-volume
test-volume
(MoeLove) ➜  ~ docker volume ls
DRIVER              VOLUME NAME
local               k-volume
local               test-volume
```

也可以选择直接挂载至容器中，Docker 发现本地没有 volume 时，便会自动创建了。

当然，还有另外一种常用但是并不常见的方法，便是在构建镜像的时候使用的 `VOLUME` 语法。例如：

在编写 Dockerfile 时，使用 VOLUME 语法，表示此处数据需要持久化。使用构建完成的镜像启动容器时，会自动创建一个存储卷。例如：

复制

```
(MoeLove) ➜  k cat Dockerfile
FROM alpine

VOLUME /app

# 构建镜像
(MoeLove) ➜  k docker build -q -t local/test-volume .
sha256:a8a1a967ffc5de9ba3930327ae529bc6448c6c1591ff19136bd4e60731ac8aca

# 运行容器
(MoeLove) ➜  k docker run --rm -it local/test-volume sh
/ #

# 查看是否有 volume
(MoeLove) ➜  ~ docker volume ls
DRIVER              VOLUME NAME
local               9343a7021f9c0e03e844f416a9c35c7ed26924c192b2d7580d7365065ee07
local               new-test-volume
```

可以看到已经自动创建了一个持久化卷。

查

`docker volume ls` 这个命令就可以看到当前管理的 Docker volume 啦。例如：

复制

```
(MoeLove) ➜  ~ docker volume ls
DRIVER              VOLUME NAME
local               9343a7021f9c0e03e844f416a9c35c7ed26924c192b2d7580d7365065ee07
local               new-test-volume
```

或者直接在其数据文件中查找：

复制

```
(MoeLove) ➜  ~ sudo ls -al /var/lib/docker/volumes
总用量 100
drwx------.  4 root root  12288  1月 21 02:55 .
drwx--x--x. 15 root root   4096  1月 20 07:14 ..
drwxr-xr-x.  3 root root   4096  1月 21 02:55 9343a7021f9c0e03e844f416a9c35c7ed269
-rw-------.  1 root root 131072  1月 21 02:55 metadata.db
drwxr-xr-x.  3 root root   4096  1月 21 02:41 new-test-volume
```

这个目录中的文件名便是持久化卷的名称，在 metadata.db 中包含了存储卷的一些信息。

当然，也可以直接 `docker inspect` 查看 Docker 容器的具体信息，其中包含了持久化卷的信息。

删除

`docker volume rm` 便可完成其删除操作。例如：

复制

```
(MoeLove) ➜  ~ docker volume ls
DRIVER              VOLUME NAME
local               9343a7021f9c0e03e844f416a9c35c7ed26924c192b2d7580d7365065ee07
local               new-test-volume
(MoeLove) ➜  ~ docker volume rm new-test-volume
new-test-volume
(MoeLove) ➜  ~ docker volume ls
DRIVER              VOLUME NAME
local               9343a7021f9c0e03e844f416a9c35c7ed26924c192b2d7580d7365065ee07
```

也可以使用清理所有空闲 volume 的方式：`docker volume prune` **使用时，请千万小心。**

改

对于持久化卷而言，只要绑定到具体的容器上，并且没有设置 read only 选项，那就可以直接修改其内容了。

总结

本篇，我为你介绍了 Docker 持久化卷相关的知识。在数据需要持久化时，使用 Docker volume 非常方便，但需要注意与 bind mount 的方式区分开。

Docker volume 在使用上，没有过多难点。下一篇我将为你介绍基于 Docker volume 的数据备份和恢复。