20 数据存储：Docker 数据存储的三种模式

更新时间：2020-08-31 10:16:07

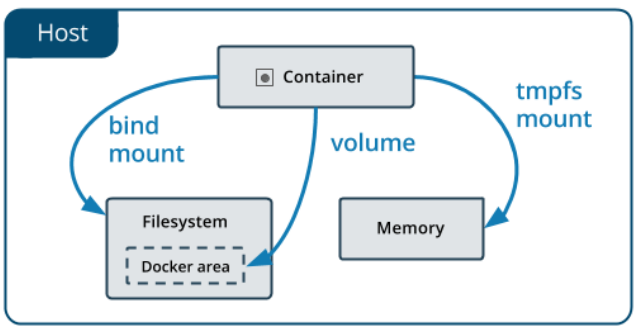
> 每个人的生命都是一只小船，理想是小船的风帆。——张海迪

我们知道在 Docker 容器中创建出来的文件默认都是存储在一个可写的容器文件层，也就是说一旦容器停止运行，这些数据就丢失了。

如果我们想要在 Docker 容器停止之后创建的文件依旧存在，也就是将文件在宿主机上保存。那么我们有两种方式：***volumes***、***bind mounts***。如果 Docker 是运行在 Linux 系统上，那么我们还可以使用 ***tmpfs***；对应在 Windows 系统上，可以使用 ***named pipe***。我们这里主要讨论 ***volumes***、***bind mounts*** 和 ***tmpfs***。

1. 概览

在开始详细讨论每一种存储模式之前，我们先来看一下这三种模式的区别，如下图。



Volumes 会把文件存储到宿主机的指定位置，在 Linux 系统上这个位置为 `/var/lib/docker/volumes/`。这些文件只能由 Docker 进程进行修改，是 docker 文件持久化的最好的方式。

bind mounts 可以将文件存储到宿主机上面任意位置，而且别的应用程序也可以修改。

tmpfs 只会将数据存储到宿主机的内存中，并不会落盘。

下面我们详细看一下这三种方式。

2. volumes

Volumes 由 Docker 创建和管理。我们可以通过命令 `docker volume create` 显式地创建 volume，也可以由 Docker 进程在需要的时候自动创建，比如服务初始化的时候。

当我们创建一个 volume 之后，这个 volume 的数据会存储在宿主机的指定目录。然后我们可以将这个 volume 挂载到容器内部，然后我们就可以在容器内部的对应挂载点访问这个 volume。这种挂载的方式和 ***bind mounts*** 很相似，区别在于 volume 只能由 Docker 进程管理。

同一个 volume 可以同时挂载到多个容器内部。Docker 并不会在没有容器使用 volume 的时候自动删除该 volume，我们可以通过命令 `docker volume prune` 来移除指定的 volume。

当我们挂载 volume 的时候，可以指定一个名字，如果没有指定名字，那么系统将会分配一个随机的名字（系统保证名字唯一）。volume 也支持 ***volume driver***，通过 ***volume driver*** 我们可以将数据存储到远端的机器或者云平台上面，感兴趣的可以自己查阅相关文档。

Volume 相比 ***bind mounts*** 的优点包括：

- 相更容易做数据备份和迁移
- 可以使用 Docker CLI 或者 Docker API 管理 volume
- Volume driver 可以让我们使用远端存储
- 可以在容器间共享和重用

除此之外，相比将数据保存在容器的写入层，volume 是一种更好的数据持久化方式。因为 volume 不会增加容器的大小，同时 volume 的数据存活独立于容器的生命周期。

如何使用 volumes

首先我们创建一个名字叫做 `my-vol` 的 volume。

```
[root@docker ~]# docker volume create my-vol
my-vol
[root@docker ~]# docker volume ls
DRIVER              VOLUME NAME
local               0d677566872e112e6792b7dd1e71f4b5c26fec701de4d43fe401fd1d5bd93afd
local               12a0226aff0e607425bd2f8ed6544154ec276feda24dee39255e377b978d4014
local               22340dc6d144f4f4be30c93afc1186734f8559acb20aeeb861fa929d4c26e30b
local               a7fe694cc0abea99d1e455e31d25a49a523e4ff661f4172d48e3b61ccd00c2c0
local               my-vol
```

DRIVER 为 local 表示 volume 都是本地存储。通过 ls 我们可以看到所有的 volume 列表，除了我们创建出来的 my-vol，其他几个都是随时生成的 name。我们前面说过所有的 volume 都位于宿主机的一个指定目录，也就是 `/var/lib/docker/volumes` ，我们来看一下。

```
[root@docker ~]# ls /var/lib/docker/volumes/
0d677566872e112e6792b7dd1e71f4b5c26fec701de4d43fe401fd1d5bd93afd  22340dc6d144f4f4be30c93afc1186734f8559acb20aeeb861fa929d4c26e30b  metadata.db
12a0226aff0e607425bd2f8ed6544154ec276feda24dee39255e377b978d4014  a7fe694cc0abea99d1e455e31d25a49a523e4ff661f4172d48e3b61ccd00c2c0  my-vol
```

删除 volume 的命令为 `docker volume rm`，如果要删除我们上面创建出来的 my-vol，可以执行命令。

```
$ docker volume rm my-vol
```

我们暂时先不删除。下面演示如果将 volume 挂载到容器中。可以通过参数 `-v/--volume` 和 `--mount` 来使用 volume。这两个参数在设计之初区别在于：`-v/--volume` 用于单个容器；而 `--mount` 用于 swarm service。但是，在 Docker 17.06 版本之后，对于单个容器应用也可以使用 `--mount` 参数。下面是两个参数的使用示例。

```
$ docker run -d \
  --name devtest \
  -v myvol2:/app \
  nginx:latest
```

```
$ docker run -d \
  --name devtest \
  --mount source=myvol2,target=/app \
  nginx:latest
```

可以看到我们上面挂载了一个没有预先创建的 volume，也就是 myvol2，对于这种情况，docker 会自动帮我们创建出来。

```
[root@docker ~]# docker run -d --name devtest -v myvol2:/app nginx:latest
ea5a78df94f327799e1cb4d809386b8696321b0c1bef262ee743293a4ebf00ce
[root@docker ~]# docker volume ls
DRIVER              VOLUME NAME
local               0d677566872e112e6792b7dd1e71f4b5c26fec701de4d43fe401fd1d5bd93afd
local               12a0226aff0e607425bd2f8ed6544154ec276feda24dee39255e377b978d4014
local               22340dc6d144f4f4be30c93afc1186734f8559acb20aeeb861fa929d4c26e30b
local               a7fe694cc0abea99d1e455e31d25a49a523e4ff661f4172d48e3b61ccd00c2c0
local               my-vol
local               myvol2
```

下面我们见到介绍一下这两个参数的异同。

`-v/--volume` 参数包含三个字段，以冒号分隔，顺序相关：

- 第一个字段是 volume 的名字，单台宿主机上 volume 名字唯一，如果是匿名的 volume，第一个字段可以忽略
- 第二个字段是容器内的挂载点
- 第三个字段是以逗号分隔开的一系列的可选参数

`--mount` 参数包含一系列的 key-value 对，以逗号分隔，比如 `'type=volume,src=<VOLUME-NAME>,dst=<CONTAINER-PATH>,volume-driver=local`

type：挂载介质的类型，可以是 `bind`、`volume` 和 `tmpfs`

source：volume 名字，也可以简写为 **src**。

destination ：容器内的挂载点，可以简写为 **dst**，或者 **target**。

readonly：可选的，如果添加则表示该 volume 是只读的。下面这个例子就是一个只读的例子。

```
$ docker run -d \
  --name=nginxtest \
  --mount source=nginx-vol,destination=/usr/share/nginx/html,readonly \
  nginx:latest
```

volume-opt：其他可选参数。

下面是一个 `--mount` 的例子。

```
$ docker service create \
   --mount 'type=volume,src=<VOLUME-NAME>,dst=<CONTAINER-PATH>,volume-driver=local,volume-opt=type=nfs,volume-opt=device=<nfs-server>:<nfs-path>,"volume-opt=o=addr=<nfs-address>,vers=4,soft,timeo=180,bg,tcp,rw"'
   --name myservice \
   <IMAGE>
```

总体来说这两个参数支持选项和功能基本一致，区别在于运行一个 service 的时候，只能使用 `--mount`。

最佳实践

volume 的最佳使用场景如下：

- 在多个容器间共享数据。volume 不会随着容器停止而被删除，只能够被显示的删除。
- 使用 volume 来保存一些配置信息，可以达到数据解耦的目的。
- 借助于 volume driver，可以将数据存储到远端机器或者云平台上。
- 数据备份、迁移等场景。我们只需要备份目录 `/var/lib/docker/volumes/<volume-name>`。

3. bind mounts

bind mounts 模式与第一种 ***volume*** 非常类似，区别在于宿主机的文件位置不是固定在 `/var/lib/docker/volumes/` 目录下，而是宿主机上面的任意目录。这也就意味着数据可以被任意程序改动。

另外当容器内部的挂载目录非空时，***bind mounts*** 和 ***volume*** 还有一些行为的差异：

使用 ***volume*** 时，这个容器目录中的文件会被复制到 volume中，也就是说容器目录原有文件不会被 volume覆盖。

使用 ***bind mounts*** 时，容器目录中原有的文件会被隐藏，从而只能读取到宿主机目录下的文件。

如何使用

使用和 volume 的使用非常类似，区别在于对于 ***bind mounts*** ，source 指定的是宿主机的目录，而不是 volume 的名字。

最佳实践

一般情况下，我们要尽可能的使用 volume。下面几种情况可以考虑使用 ***bind mounts***。

- 在宿主机和容器之间共享配置。比如容器默认挂载宿主机的文件 `/etc/resolv.conf` 来实现 DNS 解析。

- 在宿主机和容器之间共享代码或者可执行文件。比如，将一个 maven 的 `target/` 目录挂载到容器内，这样每次我们在宿主机上编译完，容器内部就能得到最新的文件。

3. tmpfs

tmpfs 只支持 Linux，不会将数据持久化到宿主机或者容器内部的文件系统上。在容器的生命周期内，数据将会保存在宿主机的内存里，一旦容器停止，数据将会被删除。和 volume 不同的是，每个容器关联的 tmpfs 不能够共享。

如何使用

可以通过两个参数 `--tmpfs` 和 `--mount` 来使用 ***tmpfs*** 。在设计之初，`--tmpfs` 参数是给单个容器作为参数使用的，`--mount` 参数用在 swarm 中。但是在 Docker 17.06 版本之后，`--mount` 参数也可以用在单个容器，而且 `--mount` 参数也会更加的直观。下面是两种使用示例：

```
$ docker run -d \
  -it \
  --name tmptest \
  --tmpfs /app \
  nginx:latest
```

```
docker run -d \
  -it \
  --name tmptest \
  --mount type=tmpfs,destination=/app,tmpfs-mode=1770 \
  nginx:latest
```

`--tmpfs` 不能指定额外参数，`--mount` 针对 ***tmpfs*** 提供了额外的可选参数：

- tmpfs-size: 指定 tmpfs 的大小，默认不受限制，单位 byte
- tmpfs-mode：Linux 系统的文件模式，比如 700；默认值为 1777，也就是任何用户都可以写。

最佳实践

tmpfs 的最佳使用场景是不希望数据持久化到容器文件系统或者宿主机上。比如出于安全考虑，将一些认证信息存储到 tmpfs 中，或者出于性能考虑，将一些 state 信息存储在内存中，同时又不需要持久话。

4. 总结

本文介绍了宿主机和容器之间数据交互的三种方式：***volumes***、***bind mounts*** 和 ***tmpfs***，并介绍了这三种方式对应的最佳使用场景，希望大家在日常使用中可以对号入座。

}

19 容器间网络通信：link 的技术原理解析

21 数据共享：volume 的使用指南