# 深入理解Docker Volume

本文主要介绍了Docker Volume的原理以及使用方式，是[Docker入门教程](http://dockerone.com/article/111)的延伸。作者通过从数据的共享、数据容器、备份、权限以及删除Volume五方面深入介绍了Volume的工作原理，从实战中帮助读者了解Volume。

很多人还不是很明白Docker Volume的工作原理。在这篇文章中，我会尽最大的努力来解释Volume是如何工作的，并展示一些最佳实践。这篇文章主要是针对那些对Volume不了解的Docker用户，当然有经验的用户也可以通过本文了解一些Volume的细节。

想要了解Docker Volume，首先我们需要知道Docker的文件系统是如何工作的。Docker镜像是由多个文件系统（只读层）叠加而成。当我们启动一个容器的时候，Docker会加载只读镜像层并在其上（译者注：镜像栈顶部）添加一个读写层。如果运行中的容器修改了现有的一个已经存在的文件，那该文件将会从读写层下面的只读层复制到读写层，该文件的只读版本仍然存在，只是已经被读写层中该文件的副本所隐藏。当删除Docker容器，并通过该镜像重新启动时，之前的更改将会丢失。在Docker中，只读层及在顶部的读写层的组合被称为[Union File System](https://docs.docker.com/terms/layer/#union-file-system)（联合文件系统）。  
  
 为了能够保存（持久化）数据以及共享容器间的数据，Docker提出了Volume的概念。简单来说，Volume就是目录或者文件，它可以绕过默认的联合文件系统，而以正常的文件或者目录的形式存在于宿主机上。  
  
 我们可以通过两种方式来初始化Volume，这两种方式有些细小而又重要的差别。我们可以在运行时使用-v来声明Volume：

$ docker run -it --name container-test -h CONTAINER -v /data debian /bin/bash

root@CONTAINER:/# ls /data

root@CONTAINER:/#

上面的命令会将/data挂载到容器中，并绕过联合文件系统，我们可以在主机上直接操作该目录。任何在该镜像/data路径的文件将会被复制到Volume。我们可以使用docker inspect命令找到Volume在主机上的存储位置：

$ docker inspect -f {{.Volumes}} container-test

你会看到类似的输出：

map[/data:/var/lib/docker/vfs/dir/cde167197ccc3e138a14f1a4f...b32cec92e79059437a9]

这说明Docker把在/var/lib/docker下的某个目录挂载到了容器内的/data目录下。让我们从主机上添加文件到此文件夹下：

$ sudo touch /var/lib/docker/vfs/dir/cde167197ccc3e13814f...b32ce9059437a9/test-file

进入我们的容器内可以看到：

$ root@CONTAINER:/# ls /data

test-file

只要将主机的目录挂载到容器的目录上，那改变就会立即生效。我们可以在Dockerfile中通过使用VOLUME指令来达到相同的目的：

FROM debian:wheezy

VOLUME /data

但还有另一件只有-v参数能够做到而Dockerfile是做不到的事情就是在容器上挂载指定的主机目录。例如：

$ docker run -v /home/adrian/data:/data debian ls /data

该命令将挂载主机的/home/adrian/data目录到容器内的/data目录上。任何在/home/adrian/data目录的文件都将会出现在容器内。这对于在主机和容器之间共享文件是非常有帮助的，例如挂载需要编译的源代码。为了保证可移植性（并不是所有的系统的主机目录都是可以用的），挂载主机目录不需要从Dockerfile指定。当使用-v参数时，镜像目录下的任何文件都不会被复制到Volume中。（译者注：Volume会复制到镜像目录，镜像不会复制到卷）

### 数据共享

如果要授权一个容器访问另一个容器的Volume，我们可以使用-volumes-from参数来执行docker run。

$ docker run -it -h NEWCONTAINER --volumes-from container-test debian /bin/bash

root@NEWCONTAINER:/# ls /data

test-file

root@NEWCONTAINER:/#

值得注意的是不管container-test是否运行，它都会起作用。只要有容器连接Volume，它就不会被删除。

### 数据容器

常见的使用场景是使用纯数据容器来持久化数据库、配置文件或者数据文件等。[官方的文档](https://docs.docker.com/userguide/dockervolumes/)上有详细的解释。例如：

$ docker run --name dbdata postgres echo "Data-only container for postgres"

该命令将会创建一个已经包含在Dockerfile里定义过Volume的postgres镜像，运行echo命令然后退出。当我们运行docker ps命令时，echo可以帮助我们识别某镜像的用途。我们可以用-volumes-from命令来识别其它容器的Volume：

$ docker run -d --volumes-from dbdata --name db1 postgres

使用数据容器的两个注意点：

不要运行数据容器，这纯粹是在浪费资源。

* 不要为了数据容器而使用“最小的镜像”，如busybox或scratch，只使用数据库镜像本身就可以了。你已经拥有该镜像，所以并不需要占用额外的空间。

### 备份

如果你在用数据容器，那做备份是相当容易的：

$ docker run --rm --volumes-from dbdata -v $(pwd):/backup debian tar cvf /backup/backup.tar /var/lib/postgresql/data

该示例应该会将Volume里所有的东西压缩为一个tar包（官方的postgres Dockerfile在/var/lib/postgresql/data目录下定义了一个Volume）

### 权限与许可

通常你需要设置Volume的权限或者为Volume初始化一些默认数据或者配置文件。要注意的关键点是，在Dockerfile的VOLUME指令后的任何东西都不能改变该Volume，比如：

FROM debian:wheezy

RUN useradd foo

VOLUME /data

RUN touch /data/x

RUN chown -R foo:foo /data

该Docker file不能按预期那样运行，我们本来希望touch命令在镜像的文件系统上运行，但是实际上它是在一个临时容器的Volume上运行。如下所示：

FROM debian:wheezy

RUN useradd foo

RUN mkdir /data && touch /data/x

RUN chown -R foo:foo /data

VOLUME /data

Docker可以将镜像中Volume下的文件挂载到Volume下，并设置正确的权限。如果你指定Volume的主机目录将不会出现这种情况。  
  
如果你没有通过RUN指令设置权限，那么你就需要在容器启动时使用CMD或ENTRYPOINT指令来执行（译者注：CMD指令用于指定一个容器启动时要运行的命令，与RUN类似，只是RUN是镜像在构建时要运行的命令）。

### 删除Volumes

这个功能可能会更加重要，如果你已经使用docker rm来删除你的容器，那可能有很多的孤立的Volume仍在占用着空间。  
  
Volume只有在下列情况下才能被删除：

* 该容器是用docker rm －v命令来删除的（-v是必不可少的）。
* docker run中使用了--rm参数

即使用以上两种命令，也只能删除没有容器连接的Volume。连接到用户指定主机目录的Volume永远不会被docker删除。  
  
除非你已经很小心的，总是像这样来运行容器，否则你将会在/var/lib/docker/vfs/dir目录下得到一些僵尸文件和目录，并且还不容易说出它们到底代表什么。

### 延伸阅读

以下资源更深入的探究了Volumes机制（译注：以下译文稍后奉上）：

* [疯狂Docker之纯数据容器](http://container42.com/2014/11/18/data-only-container-madness/)
* [深入Docker：Volumes](http://dockerone.com/article/129)（已译）
* [容器数据管理](https://docs.docker.com/userguide/dockervolumes/)

参考：

1. http://dockone.io/article/128