

🔗 动手实战学Docker (/courses/498)

存储管理

1. 课程说明

课程为纯动手实验教程，为了能说清楚实验中的一些操作会加入理论内容。理论内容我们不会写太多，已经有太多好文章了，会精选最值得读的文章推荐给你，在动手实践的同时扎实理论基础。

实验环境中可以联网，不受实验楼网络限制。

2. 学习方法

实验楼的 Docker 课程包含 14 个实验，每个实验都提供详细的步骤和截图，适用于有一定 Linux 系统基础，想快速上手 Docker 的同学。

学习方法是多实践，多提问。启动实验后按照实验步骤逐步操作，同时理解每一步的详细内容。

如果实验开始部分有推荐阅读的材料，请务必先阅读后再继续实验，理论知识是实践必要的基础。

3. 本节内容简介

在本节内容中，我们将讨论 Docker 中管理数据的几种方式。

本节中，我们需要依次完成下面几项任务：

1. 使用 volumes
2. 使用 bind mounts
3. 使用 tmpfs

4. 存储

4.1 概述

通过之前的学习，我们学习了有关于容器和镜像的一些知识。对于数据来说，我们可以将其保存在容器中，但是会存在一些缺点：

- 当容器不再运行时，我们无法使用数据，并且容器被删除时，数据并不会被保存。

- 数据保存在容器中的可写层中，我们无法轻松的将数据移动到其他地方。
- 🔗 动手实战学Docker (/courses/498)

针对上述的缺点而言，有些数据信息，例如我们的数据库文件，我们不应该将其保存在镜像或者容器的可写层中。Docker 提供三种不同的方式将数据从 Docker 主机挂载到容器中，分别为卷（volumes），绑定挂载（bind mounts），临时文件系统（tmpfs）。很多时候，volumes 总是正确的选择。

- volumes，卷存储在 Docker 管理的主机文件系统的一部分中（/var/lib/docker/volumes/）中。完全由 Docker 管理
- bind mounts，绑定挂载，可以将主机上的文件或目录挂载到容器中
- tmpfs，仅存储在主机系统的内存中，而不会写入主机的文件系统

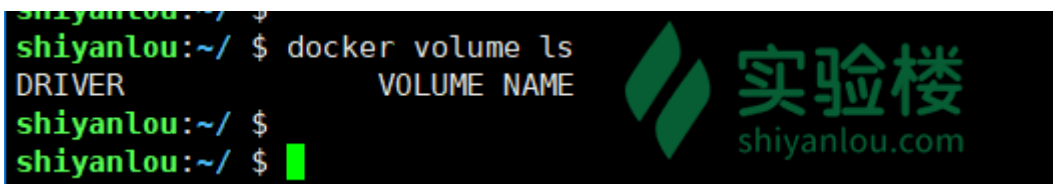
无论使用上述的哪一种方式，数据在容器内看上去都是一样的。它被认为容器文件系统目录或单个文件。

4.2 卷列表

对于三种不同的存储数据的方式来说，卷是唯一完全由 Docker 管理的。它更容易备份或迁移，并且我们可以使用 Docker CLI 命令来管理卷。

列出本地可用的卷列表可以使用如下命令：

```
$ docker volume ls
```



```
shiyanlou:~/ $  
shiyanlou:~/ $ docker volume ls  
DRIVER          VOLUME NAME  
shiyanlou:~/ $  
shiyanlou:~/ $
```

由于此时我们并未创建有相应的卷，所以显示为空。

创建卷

创建卷我们可以直接使用如下命令：

```
$ docker volume create
```

上述命令会创建一个数据卷，并且会随机生成一个名称。创建之后我们可以查看卷列表：

```
$ docker volume ls
```

```
shiyanolou:~/ $ docker volume create
b3df029b741f873e797058d95eef247af7d2acc015bf402cbb52d044dd749951
shiyanolou:~/ $
shiyanolou:~/ $ docker volume ls
DRIVER          VOLUME NAME
local           b3df029b741f873e797058d95eef247af7d2acc015bf402cbb52d044dd749951
shiyanolou:~/ $
```

这种由系统随机生成名称的创建卷的方式被称为**匿名卷**，直接使用该卷需要指定卷名，即自动生成的 ID，所以创建卷时一般手动指定其 name，例如我们创建一个名为 volume1 的卷。

```
$ docker volume create volume1
```

```
shiyanolou:~/ $ docker volume create volume1
volume1
shiyanolou:~/ $
shiyanolou:~/ $ docker volume ls
DRIVER          VOLUME NAME
local           b3df029b741f873e797058d95eef247af7d2acc015bf402cbb52d044dd749951
local           volume1
shiyanolou:~/ $
```

用卷启动一个容器

创建卷之后，我们可以用卷来启动一个容器，这里首先需要学习 docker container run 命令的两个参数：

- -v 或 --volume
 - 由三个由冒号 (:) 分隔的字段组成，[HOST-DIR:]CONTAINER-DIR[:OPTIONS]。
 - HOST-DIR 代表主机上的目录或数据卷的名字。省略该部分时，会自动创建一个匿名卷。如果是指定主机上的目录，需要使用绝对路径。
 - CONTAINER-DIR 代表将要挂载到容器中的目录或文件，即表现为容器中的某个目录或文件
 - OPTIONS 代表配置，例如设置为只读权限(ro)，此卷仅能被该容器使用(z)，或者可以被多个容器使用(z)。多个配置项由逗号分隔。
 - 例如，我们使用 -v volume1:/volume1:ro,z。代表的意思是将卷 volume1 挂载到容器中的 /volume1 目录。ro,z 代表该卷被设置为只读(ro)，并且可以多个容器使用该卷(z)
- --mount
 - 由多个键值对组成，键值对之间由逗号分隔。例如：type=volume,source=volume1,destination=/volume1,ro=true。
 - type，指定类型，可以指定为 bind，volume，tmpfs。

- source, 当类型为 volume 时, 指定卷名称, 匿名卷时省略该字段。当类型为 bind, 指定路径。可以使用缩写 src。

👉 动手实战学 Docker (/courses/498)

- destination, 挂载到容器中的路径。可以使用缩写 dst 或 target。
- ro 为配置项, 多个配置项直接由逗号分隔一般使用 true 或 false。

针对上述创建的卷 volume1, 用其来运行一个容器就可以使用如下命令:

```
$ docker container run -it --name shiyanlou003 -v volume1:/volume1 --rm ubuntu bash
```

或者我们也可以使用 --mount, 其语法格式如下:

```
$ docker run -it --name shiyanlou004 --mount type=volume,src=volume1,target=/volume1 --rm ubuntu bash
```

从命令中, 可以很明显的得出, --mount 的可读性更好。所以, 推荐大家使用 --mount

上述操作, 我们分别运行了两个容器, 并分别挂载了一个卷, 还可多次使用该参数挂载多个卷或目录。并且对于这两个容器来说, 由于我们使用的是同一个卷, 所以他们将共享该数据卷, 但是对于多个容器共享数据卷时, 需要注意并发性。大家可以分别连接到两个容器中, 操作数据, 验证其是同步的, 这里就不再详细演示了。

4.3 bind-mounts

对于数据卷来说, 其优点在于方便管理。而对于绑定挂载 (bind-mounts) 来说, 通过将主机上的目录绑定到容器中, 容器就可以操作和修改主机上该目录的内容。这既是其优点也是其缺点。

例如, 我们将 /home/shiyanlou 目录挂载到容器中的 /home/shiyanlou 目录下, 使用的命令如下:

```
$ docker run -it -v /home/shiyanlou:/home/shiyanlou --name shiyanlou005 --rm ubuntu bash
```

而如果使用的是 --mount, 相应的语句如下:

```
$ docker run -it --mount type=bind,src=/home/shiyanlou,target=/home/shiyanlou --name shiyanlou006 --rm ubuntu bash
```

如果绑定挂载时指定的容器目录是非空的, 则该目录中的内容将会被覆盖。并且如果主机上的目录不存在, 会自动创建该目录。

上述两个操作针对的是目录，而对于挂载文件来说，可能会出现一些特殊情况，涉及到绑定挂载和使用卷的区别。下面我们重现这一操作：

1. 首先在当前目录，即 `/home/shiyanlou` 目录下，创建一个 `test.txt` 文件。并向其中写入文本内容 "test1"：

```
$ echo "test1" > test.txt
```

2. 接着创建一个容器 `shiyanlou007`，将 `test.txt` 文件挂载到容器中的 `/test.txt` 文件，并查看容器中 `/test.txt` 文件的内容：

```
$ docker run -it -v /home/shiyanlou/test.txt:/test.txt --name shiyanlou007 ubuntu /bin/bash
```



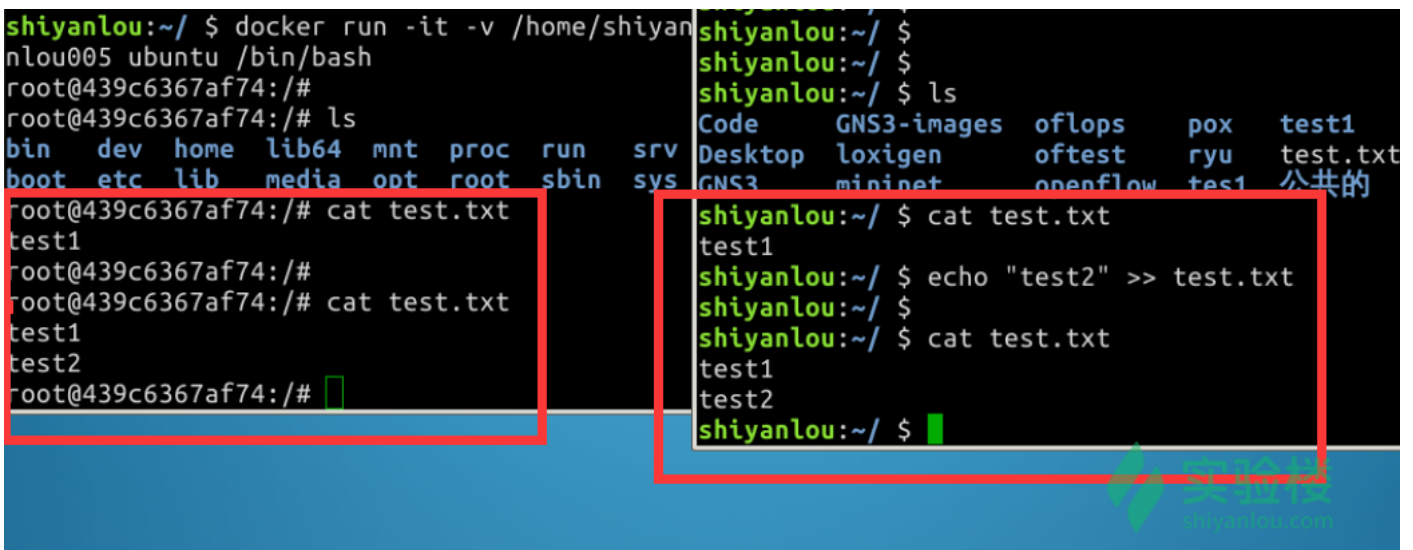
The screenshot shows a terminal window with the following commands and output:

```
shiyanlou:~/ $ docker run -it -v /home/shiyanlou/test.txt:/test.txt --name shiyanlou005 ubuntu /bin/bash
root@439c6367af74:/#
root@439c6367af74:/# ls
bin  dev  home  lib64  mnt  proc  run  srv  test.txt  usr
boot  etc  lib  media  opt  root  sbin  sys  tmp  var
root@439c6367af74:/# cat test.txt
test1
```

The file `test.txt` is highlighted in green in the `ls` output, and the output of `cat test.txt` is highlighted with a red box.

3. 这时新打开一个终端，通过 `echo` 命令向 `/home/shiyanlou/test.txt` 文件追加内容 "test2"，并在容器中查看 `/test.txt` 文件的内容：

```
$ echo "test2" >> test.txt
```



The screenshot shows two terminal windows side-by-side. The left window shows the host terminal with the following commands and output:

```
shiyanlou:~/ $ docker run -it -v /home/shiyanlou/test.txt:/test.txt --name shiyanlou005 ubuntu /bin/bash
root@439c6367af74:/#
root@439c6367af74:/# ls
bin  dev  home  lib64  mnt  proc  run  srv  test.txt  usr
boot  etc  lib  media  opt  root  sbin  sys  tmp  var
root@439c6367af74:/# cat test.txt
test1
root@439c6367af74:/#
root@439c6367af74:/# cat test.txt
test1
test2
root@439c6367af74:/#
```

The `cat test.txt` output is highlighted with a red box. The right window shows the container terminal with the following commands and output:

```
shiyanlou:~/ $
shiyanlou:~/ $
shiyanlou:~/ $ ls
Code      GNS3-images  oflops      pox      test1
Desktop   loxigen      oftest      ryu      test.txt
GNS3      mininet      openflow    tes1     公共的
shiyanlou:~/ $ cat test.txt
test1
shiyanlou:~/ $ echo "test2" >> test.txt
shiyanlou:~/ $
shiyanlou:~/ $ cat test.txt
test1
test2
shiyanlou:~/ $
```

The `cat test.txt` output in the container is highlighted with a red box.

4. 这时无论是在容器中还是主机上都能查看到该文件的内容。接下来在主机上查看 `test.txt` 的 inode 号，并使用 `vim` 编辑该文件，添加 "test3"，并查看该文件的内容：

```

shiyanolou:~/ $ docker run -it -v /home/shiyanolou0085 ubuntu /bin/bash
root@439c6367af74:/# ls
bin dev home lib64 mnt proc run srv sys usr
boot etc lib media opt root sbin sys
root@439c6367af74:/# cat test.txt
test1
root@439c6367af74:/#
root@439c6367af74:/# cat test.txt
test1
test2
不能查看到 test3
root@439c6367af74:/# cat test.txt
test1
test2
root@439c6367af74:/#

shiyanolou:~/ $ cat test.txt
test1
test2
shiyanolou:~/ $ ls -i test.txt
822079 test.txt
shiyanolou:~/ $ vim test.txt
shiyanolou:~/ $ cat test.txt
test1
test2
test3
shiyanolou:~/ $
shiyanolou:~/ $ ls test.txt
test.txt
shiyanolou:~/ $ ls -i test.txt
822108 test.txt
shiyanolou:~/ $ vim 编辑后 inode 改变
shiyanolou:~/ $

```

如上图所示，在主机上使用 vim 编辑后，通过 vim 做出的修改不能在容器中查看到。这是因为 vim 编辑保存文件的时候，会将文件内容写入到一个新的文件中，保存好后，删除掉原来的文件，并将新文件重命名，从而完成保存的操作。但是我们标识文件是通过 inode，这在第一周的内容中有讲解到，因此 Docker 绑定的主机文件，依旧是 vim 编辑之前的 inode，即旧文件。所以容器中看到的，依然是旧的内容。

对于数据卷来说，由 docker 完全管理，而绑定挂载，则需要我们自己去维护。我们需要自己手动去处理这些问题，这些问题并不仅仅是上面演示的内容，还可能有用户权限，SELINUX 等问题。

4.4 tmpfs

tmpfs 只存储在主机的内存中。当容器停止时，相应的数据就会被移除。

```
$ docker run -it --mount type=tmpfs,target=/test --name shiyanolou008 --rm ubuntu bash
```

5. 总结

本节实验主要使用三种不同的方式将数据从 Docker 主机挂载到容器中，分别为卷（volumes），绑定挂载（bind mounts），临时文件系统（tmpfs）。

请务必保证自己能够动手完成整个实验，只看文字很简单，真正操作的时候会遇到各种各样的问题，解决问题的过程才是收获的过程。

*本课程内容，由作者授权实验楼发布，未经允许，禁止转载、下载及非法传播。

上一节：Docker 镜像管理 (/courses/498/labs/1705/document)

🔍 动手实战学 Docker (/courses/498) Docker 网络管理 (/courses/498/labs/1707/document)
