数据管理

如果正在运行中的容器修改了已经存在的文件内容或者生成了新的内容，那么新产生的数据将会被复制到读写层。

Docker的镜像是分层设计的，镜像层是只读的，通过镜像启动的容器添加了一层可读写的文件系统，用户写入的数据都保存在这一层当中。

如果要将写入到容器的数据永久保存，则需要将容器中的数据保存到宿主机的指定目录，目前Docker的数据类型分为两种分别是数据卷和数据卷容器。

参考联合文件系统文章：https://mp.weixin.qq.com/s?__biz=MzI1OTY2MzMxOQ==&mid=2247495888&idx=1&sn=39ed455b12cc2e3ad18f5f72c8e6cc21&chksm=ea77c468dd004d7e42aa4a9c095822e41fd3fa01a08156181615fb66f724bc0d03ff228a39bd&mpshare=1&scene=23&srcid=0725FohVQp49EcJQS0khWpUC&sharer_sharetime=1635037154550&sharer_shareid=ac5ffb01d260d9cefd367fe5efc4eb13#rd

```
[root@docker-server1 ~]# docker inspect nginx:latest
            "Data": {
                "LowerDir": "/var/lib/docker/overlay2/01ad296a438f93e5cb283b6b462ca8da325754c6958c741221979e783f48a82c/diff:/var/lib/docker/overlay2/3639172ee8e2375e9c5d5843d4f3d9da343574bc6ebface8ee3735691fca04ea/diff:/var/lib/docker/overlay2/fdd0aea0a652268ce776642f9dfb67e221015259400579bd4d01da06931ac015/diff:/var/lib/docker/overlay2/1c85c85b62d3d884b51d4c7cebcb7c135c3f8cc00afd1704526b3a5f22f3723f/diff:/var/lib/docker/overlay2/5b2cab96eb3cfea207ddcc5750e9d950437ecdb27f912091d4e467550c2bc775/diff",
                "MergedDir": "/var/lib/docker/overlay2/0e4278c3cbb691146f01f5f69e5cc48199a1b92bda3d78c54cde984bf95921ee/merged",
                "UpperDir": "/var/lib/docker/overlay2/0e4278c3cbb691146f01f5f69e5cc48199a1b92bda3d78c54cde984bf95921ee/diff",
                "WorkDir": "/var/lib/docker/overlay2/0e4278c3cbb691146f01f5f69e5cc48199a1b92bda3d78c54cde984bf95921ee/work"
            },
            "Name": "overlay2"
LoweDir：image镜像层本身（只读）
UpperDir：容器的上层读写层
MergeDir：容器的文件系统，使用Union FS(联合文件系统)将镜像层和容器层合并给容器使用
WorkDir：容器在宿主机的工作目录
[root@docker-server1 ~]#
```

- 在容器中创建一个文件，观察文件

```
root@f59956d6babc:/# touch eagleslab.txt
root@f59956d6babc:/# md5sum eagleslab.txt
d41d8cd98f00b204e9800998ecf8427e  eagleslab.txt
[root@docker-server1 ~]# find / -name eagles*
/var/lib/docker/overlay2/5c16401d71448d83bb51efc896201e326b74f50671380f49a995e3d37e1d2363/diff/eagleslab.txt
```

```
/var/lib/docker/overlay2/5c16401d71448d83bb51efc896201e326b74f50671380f49a995e3d37e1d2363/merged/eagleslab.txt
[root@docker-server1 ~]# find / -name eagles*
/var/lib/docker/overlay2/5c16401d71448d83bb51efc896201e326b74f50671380f49a995e3d37e1d2363/diff/eagleslab.txt
/var/lib/docker/overlay2/5c16401d71448d83bb51efc896201e326b74f50671380f49a995e3d37e1d2363/merged/eagleslab.txt
[root@docker-server1 ~]# md5sum /var/lib/docker/overlay2/5c16401d71448d83bb51efc896201e326b74f50671380f49a995e3d37e1d2363/diff/eagleslab.txt
d41d8cd98f00b204e9800998ecf8427e /var/lib/docker/overlay2/5c16401d71448d83bb51efc896201e326b74f50671380f49a995e3d37e1d2363/diff/eagleslab.txt
[root@docker-server1 ~]# md5sum /var/lib/docker/overlay2/5c16401d71448d83bb51efc896201e326b74f50671380f49a995e3d37e1d2363/merged/eagleslab.txt
d41d8cd98f00b204e9800998ecf8427e /var/lib/docker/overlay2/5c16401d71448d83bb51efc896201e326b74f50671380f49a995e3d37e1d2363/merged/eagleslab.txt

```

- 删除容器后发现文件消失

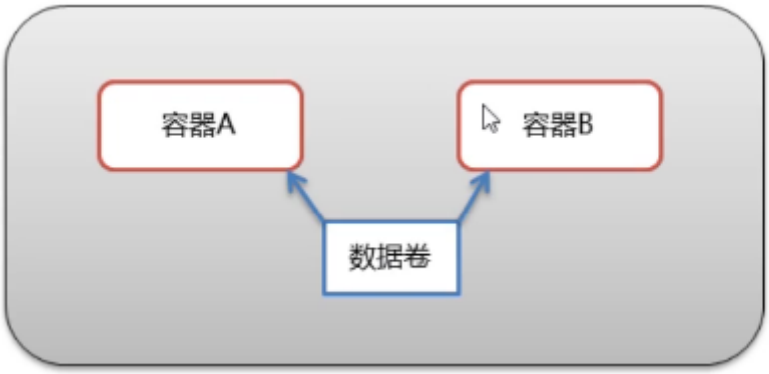
```
[root@docker-server1 ~]# docker ps -a
CONTAINER ID   IMAGE     COMMAND                  CREATED          STATUS          PORTS     NAMES
f59956d6babc   nginx     "/docker-entrypoint.…"   16 minutes ago   Up 16 minutes   80/tcp    upbeat_greider
[root@docker-server1 ~]# docker rm -fv f59956d6babc
f59956d6babc
[root@docker-server1 ~]# find / -name eagles*
```

数据卷

什么是数据卷?

数据卷实际上就是宿主机上的目录或者是文件，可以被直接mount到容器当中使用。

实际生产环境中，需要针对不同类型的服务、不同类型的数据存储要求做相应的规划，最终保证服务的可扩展性、稳定性以及数据的安全性。



数据卷案例

- 创建目录并准备页面

```
[root@docker-server1 ~]# mkdir -p /data/web
[root@docker-server1 ~]# echo 'eaglslab nginx test!' > /data/web/index.html
[root@docker-server1 ~]# cat /data/web/index.html
eaglslab nginx test
```

- 启动两个容器并验证数据

```
[root@docker-server1 ~]# docker run -d -it --name web1 -v /data/web/:/usr/share/nginx/html/ -p 8080:80 nginx
588c494dc9098e0a43e15bce3162c34676dd981609edc32d46bf4beb59b9cf19
[root@docker-server1 ~]# docker run -d -it --name web2 -v /data/web/:/usr/share/nginx/html/ -p 8081:80 nginx
ff6b3731a9ba3e0f91d2c8d89bb6573eb5e5a9b840163bc1122a9e5678d108b7
[root@docker-server1 ~]# curl 192.168.175.10:8080
eaglslab nginx test!
[root@docker-server1 ~]# curl 192.168.175.10:8081
eaglslab nginx test!
[root@docker-server1 ~]# echo 'hello world!' > /data/web/index.html
[root@docker-server1 ~]# curl 192.168.175.10:8080
hello world!
[root@docker-server1 ~]# curl 192.168.175.10:8081
hello world!
```

- 进入到容器内测试写入数据

```
[root@docker-server1 ~]# docker exec -it web1 bash
root@588c494dc909:/# echo 'docker test!' > /usr/share/nginx/html/index.html
[root@docker-server1 ~]# curl 192.168.175.10:8080
docker test!
[root@docker-server1 ~]# curl 192.168.175.10:8081
docker test!
```

- 尝试只读挂载

```
# 删除容器的时候不会删除宿主机的目录
[root@docker-server1 ~]# docker rm -fv web1
web1
[root@docker-server1 ~]# docker rm -fv web2
web2
[root@docker-server1 ~]# cat /data/web/index.html
docker test!
# 通过只读方式挂载以后，在容器内部是不允许修改数据的
[root@docker-server1 ~]# docker run -d -it --name web1 -v /data/web/:/usr/share/nginx/html/:ro -p 8080:80 nginx
a395b27958ca0cdcf52a86bd17813dcbcda4ed774895adcc99e85fc114ab84ff
[root@docker-server1 ~]# docker exec -it web1 bash
root@a395b27958ca:/# echo 123 > /usr/share/nginx/html/index.html
bash: /usr/share/nginx/html/index.html: Read-only file system

```

- 文件挂载

```
[root@docker-server1 ~]# docker run -d -it --name web2 -v /data/web/index.html:/usr/share/nginx/html/index.html:ro -p 8081:80 nginx
4b34c957372d314cdb0f85d7e2c65b095615adfe3051ae6b4266b7bacd50f374
[root@docker-server1 ~]# curl 192.168.175.10:8081
docker test!
```

数据卷特点

1. 数据卷是宿主机的目录或者文件，并且可以在多个容器之间共同使用
2. 在宿主机对数据卷更改数据后会在所有容器里面会立即更新
3. 数据卷的数据可以持久保存，即使删除使用该数据卷卷的容器也不影响
4. 在容器里面写入数据不会影响到镜像本身。
5. 需要挂载多个目录或者文件的时候可以使用多个-v参数指定
6. 数据卷使用场景包括日志输出、静态web页面、应用配置文件、多容器间目录或文件共享

数据卷容器

数据卷容器功能是可以让数据在多个docker容器之间共享，即可以让B容器访问A容器的内容，而容器c也可以访问A容器的内容，即先要创建一个后台运行的容器作为Server，用于卷提供，这个卷可以为其他容器提供数据存储服务，其他使用此卷的容器作为客户端。

- 先启动一个卷容器Server

```
[root@docker-server1 ~]# docker run -d -it --name nginx-web -v /data/web/:/usr/share/nginx/html/:ro -p 8081:80 nginx
```

- 启动两个客户端容器

```
[root@docker-server1 ~]# docker run -d -it --name web1 -p 8082:80 --volumes-from nginx-web nginx:latest
ac22faa405ec07c065042465cd7f9d456be891effdd5d13d9571b96ef9c550f7
[root@docker-server1 ~]# docker run -d -it --name web2 -p 8083:80 --volumes-from nginx-web nginx:latest
e084845475b01dedfdae7362f6fbece7b5ab57ff6289c8c9bf08251f5ba448ed
```

- 访问测试

```
[root@docker-server1 ~]# curl 192.168.175.10:8081
docker test!
[root@docker-server1 ~]# curl 192.168.175.10:8082
docker test!
[root@docker-server1 ~]# curl 192.168.175.10:8083
docker test!
```

- 停止卷容器可以创建新容器

```
[root@docker-server1 ~]# docker stop nginx-web
nginx-web
[root@docker-server1 ~]# docker run -d -it --name web3 -p 8084:80 --volumes-from nginx-web nginx:latest
6ebd95c132ee1a9e4b43d1849efc628ca7185187a59d70b3816ff16dd47b6e8e
[root@docker-server1 ~]# curl 192.168.175.10:8084
docker test!
```

- 删除卷容器之后不可以再创建新容器

```
[root@docker-server1 ~]# docker rm -fv nginx-web
nginx-web
[root@docker-server1 ~]# docker run -d -it --name web4 -p 8085:80 --volumes-from nginx-web nginx:latest
docker: Error response from daemon: No such container: nginx-web.
See 'docker run --help'.
# 但是之前已经创建好的容器不会有任何影响
[root@docker-server1 ~]# curl 192.168.175.10:8082
docker test!
```

总结:

在当前环境下，即使把提供卷的容器Server删除，已经运行的容器Client依然可以使用挂载的卷，因为容器是通过挂载的方式访问数据的，但是无法创建新的卷容器客户端，但是再把卷容器Server创建后即可正常创建卷容器client，此方式可以用于线上共享数据目录等环境，因为即使数据卷容器被删除了，其他已经运行的容器依然可以挂载使用

数据卷容器可以作为共享的方式为其他容器提供文件共享，可以在生成中启动一个实例挂载本地的目录，然后其他的容器分别挂载此容器的目录，即可保证各个容器之间的数据一致性。