2021.3.10 Dockr 数据的持久化

一．Docker 数据持久化

查看 docker 命令帮助（重点）

```
[root@server04 ~]# docker --help
```

1.数据持久化介绍

再 docker 中若想要实现容器数据的持久化（所谓的数据持久化即数据不随着 Container 的结束而销毁），需要把数据从宿主机挂载到容器中。目前 docker 提供了三种不同的方式将数据从宿主机挂载到容器中

（1）Volumes：docker 会管理宿主机文件系统的一部分资源，默认位于/var/libdocker/volumes 目录中：（最常用的方式）

实验操作：

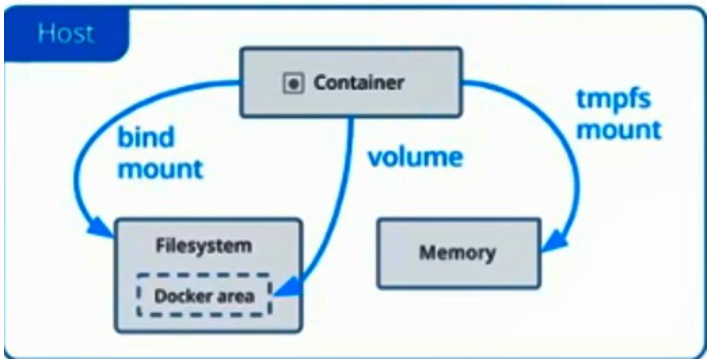
```
[root@server04 ~]# docker run -it -v /opt/ centos:7 /bin/bash   #基于 centos 启动一个容器，将/opt/文件做一个持久化

[root@5bb7b5b68e84 /]#  touch /opt/2.txt   #创建一个空文件，并退出

[root@server04~]#ls/var/lib/docker/volumes/5e3dc853faaff7ed33737983d6e1452f2f9b70012e41f62fef1c260585b5c85a/_data/  #查_data 下看是否具有 2.txt（注意 ID 是数据卷 ID）
```

（2）bind mounts：意为着可以指定储存在宿主机系统的任意位置：（比较常用的方式），但是 bind mounts 在不同的宿主机系统之间是不可移植的，比如 windows 和 linuz 的存储结构是不一样的，bind mounts 所指向的 host 目录也不能一样。这也是为什么 bind mounts 不能出现在 dockerfile 中的原因，因为会导

致 dockerfile 无法移植

（3）tmpfs：挂载存储在宿主机系统的内存中，而不是写入宿主机的文件系统

（一般不会用的方式）



2. volume 的基本使用

1）管理卷

创建一个自定义容器卷

```
[root@server04 ~]# docker volume  create nginx-data   #创建一个数据卷
nginx-data
```

2）查看所有的容器卷

```
[root@server04 ~]# docker volume ls
DRIVER    VOLUME NAME
local     5e3dc853faaff7ed33737983d6e1452f2f9b70012e41f62fef1c260585b5c85a
local     bcafa0d675985c9b763e5ea1848510a8c081d3bbe7869d16bd472f8a4d0b1229
local     bd6c96a7e25eb464624eb6e4af30d8336b9d964550b35945b9d88b09f795bb7b
local     d95e6f676edbcea6fe7569c56fb710166d6852650f0c393b4d1f6a1c77e4b38e
local     nginx-data
```

注意一下： "Mountpoint": "/var/lib/docker/volumes/nginx-data/_data", #文件的存储位置

3）使用自定义数据卷

有了自定义容器卷，我们可以创建一个使用这个数据卷的容器

```
docker run -d -it --name=nginx -p 8000:80 -v nginx-data:/usr/share/nginx/html nginx
[root@server04 ~]# ls /var/lib/docker/volumes/nginx-data/_data/    #查看是否使用
成功
```

```
50x.html  index.html
```

-v ：代表挂载数据卷，这里使用自定数据卷 nginx-data（类似于光盘挂载的感觉）

即使是容器停止运行或者被删除，数据还是存在于 /var/lib/docker/volumes/nginx-data/_data/的数据卷内的

这样数据就达到永久化了

删除数据卷： docker volume rm 数据卷名

```
[root@server04 ~]# docker volume rm nginx-data   #删除数据卷

[root@server04 ~]# docker volume  ls  #查看是否被删除
```

如果回显是'数据卷正在被使用'那么就要把使用它的镜像和容器都删除

3.Bind mounts 的基本使用（类似于挂载）（不常用的方式）

1）使用卷创建一个容器

8.3.1 使用卷创建一个容器

```
[root@localhost ~]# docker run -d -it --name=nginx -p 800:80 -v
/wwwroot:/usr/share/nginx/html nginx
d7e201c67bdfbd88ac2aa04590889ac4cc3e2f473f90b1b12dd7c5158f1ec306
```

这里指定了将宿主机上的 /wwwroot 目录（如果没有会自动创建）挂载到 /usr/share/nginx/html （这个目录是 yum 安装 nginx 的默认网页目录）。

docker 挂载的默认权限是读写(rw),用户也可以通过 ro 指定为只读

```
[root@localhost ~]# docker run -d -it --name=nginx -p 800:80 -v
/wwwroot:/usr/share/nginx/html:ro nginx
d7e201c67bdfbd88ac2aa04590889ac4cc3e2f473f90b1b12dd7c5158f1ec306
```

```
[root@localhost ~]# docker exec -it nginx /bin/bash
root@d7e201c67bdf:/# ls /usr/share/nginx/html
```

可以看到，与 volumes 不同，bind mounts 的方式会隐藏掉被挂载目录里面的内容（如果非空的话），这里是/usr/share/nginx/html 目录下的内容被隐藏掉了，因此我们看不到。

2）验证绑定

```
[root@localhost ~]# docker inspect nginx
            "HostConfig": {
                "Binds": [
                    "/wwwroot:/usr/share/nginx/html"
                ],
```

3）清理数据卷

```
[root@localhost ~]# docker stop nginx
nginx
[root@localhost ~]# docker rm nginx
nginx
[root@localhost ~]# ls /wwwroot/
index.html
```

同 volumes 一样，当我们清理掉容器之后，挂载目录里面的文件仍然还在，不会随着容器的结束而消失，从而实现数据持久化。

4.数据卷容器（不常用的方式）

8.4、数据卷容器

8.4.1、数据卷容器概述

用户需要在容器之间共享一些持续性更新的数据时，可以使用数据卷容器。数据容器也是一个普通的容器。里边带有设置好的数据卷，专门提供给其他容器挂载使用。 通过 --volumes-from 数据卷容器名来实现。

8.4.2、创建数据卷容器

```
[root@localhost ~]# docker run -it -v /dbdata:/dbdata --name=dbdata centos /bin/bash
[root@56c18602fb79 /]# exit
exit
```

//创建一个数据卷容器，并在其中创建一个数据卷挂载到/dbdata

进入 test1 容器创建文件测试

```
[root@localhost ~]# docker run -it --volumes-from dbdata --name test1 centos /bin/bash
[root@13ab94ee6fde /]# ls
anaconda-post.log  bin  dbdata  dev  etc  home  lib  lib64  lost+found  media  mnt
opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@13ab94ee6fde /]# touch dbdata/crushlinux
```

```
anaconda-post.log  bin  dbdata  dev  etc  home  lib  lib64  lost+found  media  mnt
opt  proc  root  run  sbin  srv  sys  tmp  usr  var
[root@fe011c0bb730 /]# ls dbdata/
crushlinux      //测试文件还在
```

说明：

1. 可以多次使用--volume-from 参数从多个容器挂载多个目录。也可以从其他已经挂载了数据卷的容器来挂载数据卷（类似传递）。
2. 再次强调：如果删除了挂载的容器，数据卷不会被自动删除。如果要删除容器的时候同时删除数据卷，需加上-v 参数。