容器存储

前言

- 本章主要介绍容器中的数据如何管理和存放，包括Storage driver、volume和bind mount，并介绍如果通过这些技术实现容器间的数据共享。

目标

- 学完本课程后，您将能够：
  - 描述Storage driver类型
  - 了解volume和bind mount的实现原理
  - 掌握容器数据共享方法

目录

Storage Driver

- Storage driver：管理镜像层和容器层
  - Storage driver处理各镜像层及容器层的处理细节，实现了多层数据的堆叠，为用户提供了多层数据合并后的统一视图。
  - 所有Storage driver都使用可堆叠图像层和写时复制（CoW）策略。
  - docker info命令可查看当前系统上的storage driver。

Storage Driver类型	功能
overlay2	所有当前Linux发行版都支持的首选存储驱动程序。
AUFS	仅在Ubuntu和Debian上支持。
Device Mapper	CentOS和RHEL的推荐存储驱动程序。但当前版本的CentOS和RHEL现在都支持overlay2。
Btrfs	仅在SLES上支持。
ZFS	仅支持Ubuntu 14.04或更高版本。
VFS	主要用于测试目的，不建议用于生产环境。

- Storage driver管理的容器层，一般用于放置无状态的应用的数据，即不需要持久化存储的数据。
- 选择storage driver时，建议优先选择Linux发行版默认的storage dirver。
- 参见：https://docs.docker.com/storage/storagedriver/select-storage-driver/

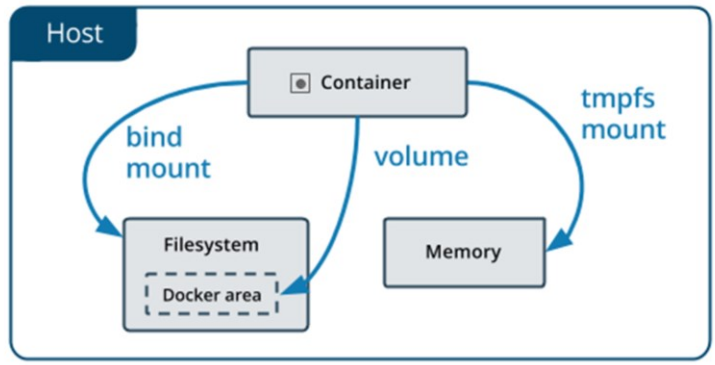
查看Storage Driver

- 使用docker info可查看当前系统使用的Storage driver。

```
[root@localhost ~]# docker info
Containers: 47
 Running: 1
 Paused: 0
 Stopped: 46
Images: 14
Server Version: 18.06.2-ce
Storage Driver: overlay2
 Backing Filesystem: xfs
 Supports d_type: true
 Native Overlay Diff: true
Logging Driver: json-file
Cgroup Driver: cgroupfs
```

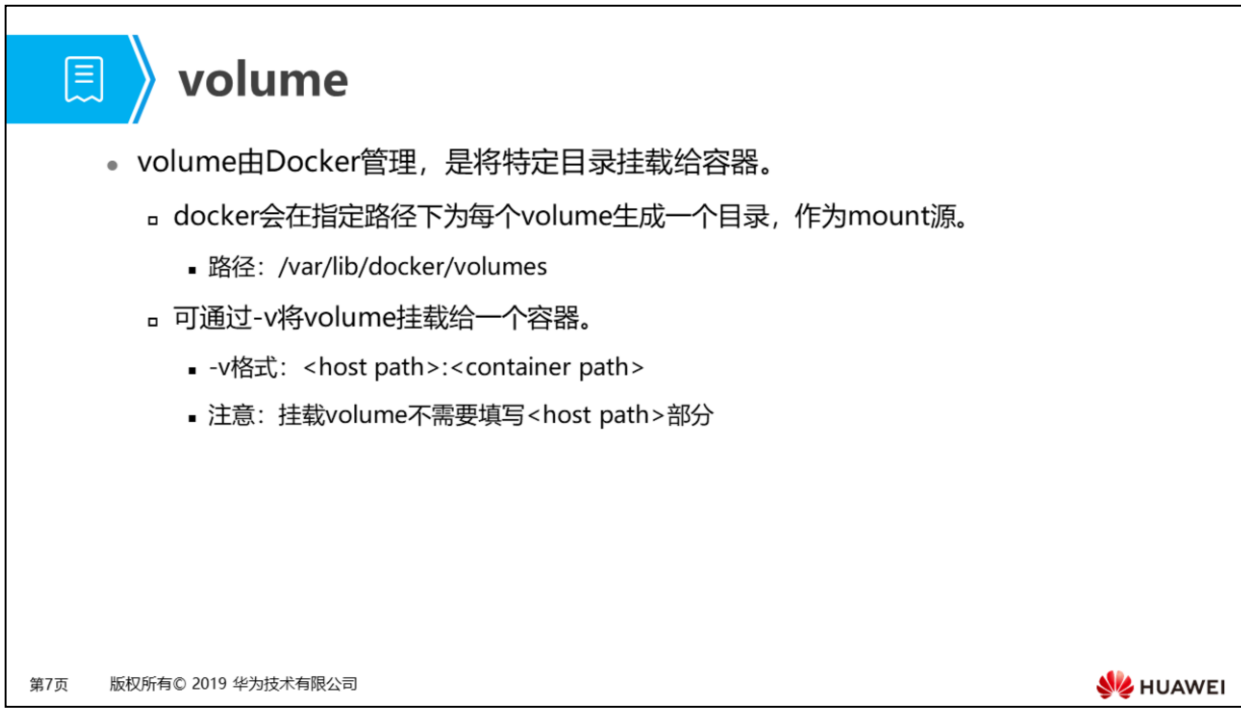
Docker数据管理

- docker容器中持久化数据一般采用两种存储方式：
  - volume
  - bind mount



HUAWEI

- Storage driver管理的容器层，一般用于放置无状态的应用的数据，即不需要持久化存储的数据。对于需要持久化存放的数据，则需要使用volume或bind mount。
  - 无论是volume还是bind mount，其本质上是宿主机文件系统中的目录或文件。
  - 无论是volume还是bind mount，其上存放的数据生命周期独立于容器。即容器删除之后，volume或bind mount上的数据，依旧存在。
- 上图来自Docker官网，本章聚焦讨论volume与bind mount，不讨论tmpfs mount。



- Volume完全由Docker管理：使用卷时，会在主机上的Docker存储目录中创建一个新目录，Docker会管理该目录的内容。
- Volume使用注意事项：
  - Volume的内容存在于容器的生命周期之外：删除容器后，Docker数据卷仍然存在。
  - 挂载volume时，不需要指定mount源，指定mount point即可。Docker会在/var/lib/docker/volumes路径下为每个volume生成一个目录，作为mount源。
  - 若mount point指向容器中的已有目录，则该目录下的数据会被copy到volume中。
  - 若mount point指向容器中的空目录，则会自动创建所需目录。
  - 若启动挂载尚不存在的卷的容器，Docker会为自动创建卷。
  - Volume在使用时，可通过ro参数将容器对volume的权限设置为只读。
- Volume具体描述可参见：https://docs.docker.com/storage/volumes/

volume示例 (1)

- 创建一个卷，挂载给一个httpd容器。

```
[root@localhost ~]# docker run -d -p 8080:80 -v /usr/local/apache2/htdocs httpd
93120e815ee5ad96b4ff1a6610ed3dd8b6b4c09808c593f47b3763f05709c4cc
```

- 使用docker volume ls命令查看卷信息。

```
[root@localhost ~]# docker volume ls
DRIVER              VOLUME NAME
local               e86e179bd904bf434e9aa70d0e0d1b372b16f1853afb8271cebb50fd0eba0014
```

- 使用docker volume inspect命令查看卷挂载信息。

```
[root@localhost ~]# docker volume inspect e86e179bd904bf434e9aa70d0e0d1b372b16f1853afb8271cebb50fd0eba0014
[
    {
        "CreatedAt": "2019-08-11T23:51:20-04:00",
        "Driver": "local",
        "Labels": null,
        "Mountpoint": "/var/lib/docker/volumes/e86e179bd904bf434e9aa70d0e0d1b372b16f1853afb8271cebb50fd0eba0014/_data",
        "Name": "e86e179bd904bf434e9aa70d0e0d1b372b16f1853afb8271cebb50fd0eba0014",
        "Options": null,
        "Scope": "local"
    }
]
```

- 在/var/lib/docker/volume下已自动生成一个带有volume name的文件夹。

- 注：/usr/local/apache2/htdocs是Apache Server中存放静态文件的目录。
- Volume常用操作命令
  - docker volume create：创建卷。
  - docker volume ls：查看卷信息。
  - docker volume rm：删除卷。

volume示例 (2)

- 使用docker inspect命令查看容器中的Mounts信息，注意Type字段的值是volume。

```
"Mounts": [
    {
        "Type": "volume",
        "Name": "e86e179bd904bf434e9aa70d0e0d1b372b16f1853afb8271cebb50fd0eba0014",
        "Source": "/var/lib/docker/volumes/e86e179bd904bf434e9aa70d0e0d1b372b16f1853afb8271cebb50fd0eba0014/_data",
        "Destination": "/usr/local/apache2/htdocs",
        "Driver": "local",
        "Mode": "",
        "RW": true,
        "Propagation": ""
    }
],
```

- 查看volume中的内容，与容器中的内容一样。容器中的数据被copy到了volume中。

```
[root@localhost ~]# curl 127.0.0.1:8080
<html><body><h1>It works!</h1></body></html>
[root@localhost ~]# cd /var/lib/docker/volumes/e86e179bd904bf434e9aa70d0e0d1b372b16f1853afb8271cebb50fd0eba0014/_data
[root@localhost _data]# ls
index.html
[root@localhost _data]# cat index.html
<html><body><h1>It works!</h1></body></html>
```


- 在proc/$$/mountinfo中也能看到挂载信息，$$代表进程号。

volume示例 (3)

- 在宿主机上更新volume内容，发现容器上也同时更新了。

```
[root@localhost ~]# echo "Welcome to Huawei" >
/var/lib/docker/volumes/e86e179bd904bf434e9aa70d0e0d1b372b16f1853afb8271cebb50fd0eba0014/_data/index.html
[root@localhost ~]# curl 127.0.0.1:8080
Welcome to Huawei
```

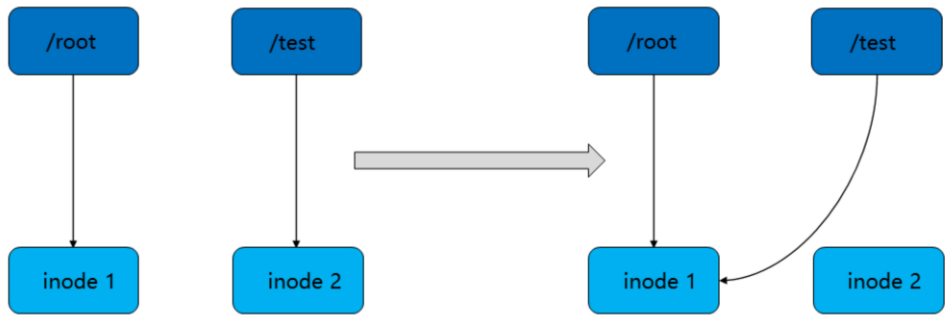
- 通过以上示例可知，volume可以实现容器和宿主机之间的数据共享。

- 销毁容器后，volume依旧存在，其数据可持久化保存。

```
[root@localhost ~]# docker kill 93120e815ee5
93120e815ee5
[root@localhost ~]# docker volume ls
DRIVER              VOLUME NAME
local               e86e179bd904bf434e9aa70d0e0d1b372b16f1853afb8271cebb50fd0eba0014
[root@localhost ~]# cat
/var/lib/docker/volumes/e86e179bd904bf434e9aa70d0e0d1b372b16f1853afb8271cebb50fd0eba0014/_data/index.html
Welcome to Huawei
```

bind mount机制

- bind mount是将宿主机上已有的目录或文件mount到容器中。



bind mount实际上是一个inode替换的过程

  HUAWEI

- Linux绑定挂载（bind mount）机制的主要作用是，允许将一个目录或者文件（不是整个设备）挂载到一个指定的目录上。且，在该挂载点上进行的任何操作，只是发生在被挂载的目录或者文件上，而原挂载点的内容则会被隐藏起来且不受影响。
- bind mount使用注意事项：
  - 容器运行过程中，对bind mount目录中改动的数据，将被保存。删除容器后，bind mount中的数据仍然存在。
  - bind mount可以挂载一个目录到容器，也可以挂载一个文件到容器。但必须要指定目录或文件的路径，即mount源，当然也必须指定mount point。这也限制了容器的可移植性。
  - 若将bind mount绑定到容器上的某非空目录下，则会隐藏容器目录下的现有内容。若不希望容器的整个目录被覆盖，可单独挂载某个文件。
  - 若mount源指向的文件或目录在宿主机上不存在。则会自动创建。
  - bind mount时，可通过ro参数将容器对数据的权限设置为只读。（设置ro参数后，容器无法对数据进行修改，但宿主机依旧有权限修改其内容。）
- bind mount具体描述参见：https://docs.docker.com/storage/bind-mounts/

bind mount示例 (1)

- 将/root/htdocs目录下的index.html文件挂载给一个httpd容器。

```
[root@localhost ~]# docker run -d -p 8081:80 -v ~/htdocs:/usr/local/apache2/htdocs httpd
100fe2c2dc3fc7f0f06fee1f292701aa000532839defdc47b307d63a84395993
```

- 分别查看宿主机和容器中的index.html文件，发现两者的内容是一样的。

```
[root@localhost ~]# cat /root/htdocs/index.html
<html><body><h1>Welcome to Huawei</h1></body></html>
[root@localhost ~]# curl 127.0.0.1:8081
<html><body><h1>Welcome to Huawei</h1></body></html>
```

- 更新宿主机上的index.html文件内容，并再次查看容器中的内容。

```
[root@localhost ~]# echo "Huawei Training Center" > /root/htdocs/index.html
[root@localhost ~]# curl 127.0.0.1:8081
Huawei Training Center
```

dokcer inspect查看该容器的Mounts部分，Type字段的值是?

- bind mount也使用-v挂载。

bind mount示例 (2)

- 另开一个终端，docker exec进入容器，发现容器目录/usr/local/apache2/htdocs已经被宿主机上的/root/htdocs覆盖。

```
[root@localhost ~]# cd /root/htdocs/
[root@localhost htdocs]# ls
index.html
[root@localhost htdocs]# docker exec -it 100fe2c2dc3f bash
root@100fe2c2dc3f:/usr/local/apache2# cd htdocs
root@100fe2c2dc3f:/usr/local/apache2/htdocs# ls
index.html
root@ff5ae8487e63:/usr/local/apache2/htdocs#
```

- 销毁容器后再次查看/root/htdocs/index.html文件内容，数据依旧存在，可持久化保存。

```
[root@localhost ~]# docker kill 100fe2c2dc3f
100fe2c2dc3f
[root@localhost ~]# cat /root/htdocs/index.html
Huawei Training Center
```

- 通过以上示例可知，bind mount可以实现容器和宿主机之间的数据共享。

目录

1. 容器存储机制
2. **数据共享**

数据共享

- 主机与容器数据共享：
  - bind mount：将Host上的目录或文件mount到容器中。
  - volume：将Host上的数据copy到容器的volume中。
    - 使用docker cp命令在容器与Host之间复制数据。
    - 使用cp命令将需要共享的数据copy到该volume的目录下。
- 容器间数据共享：
  - bind mount：将Host上的目录或文件mount到多个容器中。
  - volume：将volume挂载到多个容器中。
  - volume container：先通过volume或bind mount将数据挂载到一个container中，其他容器再引用这个container中的数据。

bind mount实现容器间数据共享

- 启动两个httpd容器，分别命名为httpd1和httpd2，并同时挂载/root/htdocs目录。

```
[root@localhost ~]# docker run --name httpd1 -d -p 1001:80 -v /root/htdocs:/usr/local/apache2/htdocs httpd
164621c418b42fbbc55f749fa0d4e8995e8b87d72ab839b21e81f0e404ae424b
[root@localhost ~]# docker run --name httpd2 -d -p 1002:80 -v /root/htdocs:/usr/local/apache2/htdocs httpd
e7949c78e756adb2313877e78354f6572137dbfc8ea49396cfffe2724844e98a
```

- 查看两个容器的index.html内容，是一致的。

```
[root@localhost ~]# cat /root/htdocs/index.html
Huawei Training Center
[root@localhost ~]# curl 127.0.0.1:1001
Huawei Training Center
[root@localhost ~]# curl 127.0.0.1:1002
Huawei Training Center
```

- 在宿主机上更新index.html内容后，再次查看容器index.html文件内容，已同步更新。

```
[root@localhost ~]# echo update the index > /root/htdocs/index.html
[root@localhost ~]# curl 127.0.0.1:1001
update the index
[root@localhost ~]# curl 127.0.0.1:1002
update the index
```

volume container实现容器间的数据共享 (1)

- 创建一个volume container。

```
[root@localhost ~]# docker create --name vc -v /root/htdocs/:/usr/local/apache2/htdocs httpd
8c2c31093964540a67cce59e0423984d5d29895950cffb3a9a0d4ed1f28d7d32
```

- 创建两个httpd容器，并引用vc中的数据。

```
[root@localhost ~]# docker run --name httpd3 -d -p 1003:80 --volumes-from vc httpd
3523a7051cc59635ce75d8e7db08db65b2fa8845bb62aa7d15189b8b80d8fe17
[root@localhost ~]# docker run --name httpd4 -d -p 1004:80 --volumes-from vc httpd
7e5dca7ad7e0b82ee33acbc98eb8bb5598af628a934dc8009c4361ff9aa424e4
```

- 使用docker inspect命令查看vc、httpd3、httpd4容器的挂载信息，三者的卷挂载信息是一致的。

```
"Mounts": [
    {
        "Type": "bind",
        "Source": "/root/htdocs",
        "Destination": "/usr/local/apache2/htdocs",
        "Mode": "",
        "RW": true,
        "Propagation": "rprivate"
    }
],
```

vc卷挂载信息

```
"Mounts": [
    {
        "Type": "bind",
        "Source": "/root/htdocs",
        "Destination": "/usr/local/apache2/htdocs",
        "Mode": "",
        "RW": true,
        "Propagation": "rprivate"
    }
],
```

httpd3卷挂载信息

- volume contaier就是一个挂载了卷的普通容器。

volume container实现容器间的数据共享 (2)

- 更新index.html文件中的数据，并访问httpd3和httpd4验证数据共享。

```
[root@localhost ~]# echo "test the volume container" > /root/htdocs/index.html
[root@localhost ~]# curl 127.0.0.1:1003
test the volume container
[root@localhost ~]# curl 127.0.0.1:1004
test the volume container
```

知识小考

- 当容器被删除后，有哪些残留文件？怎么删除磁盘上该容器的残留数据？
- bind mount一个目录到容器和bind mount一个文件到容器，有什么区别？
- 阻碍容器迁移的因素有哪些？

实验任务

- 实验任务
  - 请按照实验手册1.5部分完成容器存储部分实验。

思考题

1. docker inspect能查看到容器的哪些信息？（）
   A. 容器IP地址
   B. 容器当前状态
   C. 容器名称
   D. 容器启动时间
2. docker commit时，会将bind mount内容打包到新的镜像中。T or F

- 参考答案：
  - ABCD。
  - F。

本章总结

- Storage driver类型
- volume原理及实现
- bind mount原理及实现
- 数据共享


谢 谢
www.huawei.com