Docker存储&网络&API

前面我们学习了docker的具体使用，我们已经明白，docker体系里重要的四个对象，docker所有的功能，围绕着它们展开。

- 镜像 images
- 容器 container
- 网络 network
- 数据卷 volume

镜像原理、容器实例的创建，使用，超哥已经给大家讲解完毕了。那么网络、数据卷，是docker剩余的重点内容。

我们先看docker的数据卷内容，在一直以来，我们使用虚拟机，那么我们的应用数据，也全都是放在虚拟机的文件系统中，这种方式并不安全，比如当虚拟机损坏，我们即使可以通过快照进行系统恢复，但是后来的数据也丢失了。

因此为了保证数据独立性，我们对于数据的管理，一般都是第三方独立存放，这种操作若是对虚拟机操作，是很繁琐的，但是我们这里学的是docker管理，docker都已经默认提供好了这些功能，我们只需要用简单的命令参数，就能实现文件系统的挂载。

能这么方便，还是得益于docker底层的Union FIle System技术，通过这个UnionFS技术，完成了docker如下的数据管理

- 能够在宿主机上挂载目录、映射容器内的数据
- 自动创建独立的目录，持久化存放数据
- 容器内数据共享

这几种方式的数据共享、或者持久化的文件、或者文件夹，被称作数据卷（Volume）。

Docker数据管理

数据，是企业核心命脉，是应用程序的产出，我们讲程序部署在容器里，那么必然要管理好数据，docker容器内是一个沙箱环境，与外界隔离，那么如何与容器内的数据交互，这就是我们这节要学习的。

数据卷Data Volumes

我们使用docker容器，也需要关注容器内的存储

Data Volumes是一个可供一个或多个容器使用的特殊目录（就是你在宿主机上，的一个文件夹，这个数据，可以共享给多个容器去用）

- 数据卷可以在容器内共享和重用
- 数据卷的修改会立即生效
- 数据卷的更新，不回影响镜像
- 数据卷会一直存在，即使容器被删除

数据卷的用法类似于Linux的mount挂载操作

镜像中被指定为挂载点的目录

其中原有文件会被隐藏，显示挂载的数据卷，linux的mount功能

> 光盘 mount linux的/mnt

挂载方式

docker提供了三种，不同场景的文件系统挂载，volume

- bind mount
- volume
- tmpfs mount，基于内存的临时挂载，容器挂掉后，数据丢失

- bind mount，这是直接将 宿主机的目录 挂载到 容器内的文件系统，此时宿主机的目录、和

容器内的目录、是互通的。

- Volume，由docker自动分配宿主机的一个目录，挂载到容器内的目录。
- Tmpfs Mount，挂载宿主机内存的一部分到容器内的文件系统，这种方式数据是不持久的，容器停止，数据丢失。

实际的操作docker 数据卷的管理

bind mount模式

通过创建容器时，添加 -v --volume参数，容器内部，容器外的目录，或文件

```
必须是绝对路径

-v host-path:container-path

# 运行nginx容器，查看数据卷挂载的使用，你可以通过宿主机的目录，直接修改容器内nginx程序的页面

[root@yc_docker01 ~]# docker pull nginx:1.12

docker run -d -p 9090:80 --name chaoge_nginx -v /my_docker_nginx:/usr/share/nginx/html   nginx:1.12


# 2.尝试访问，发现403禁止访问
# 查看容器的具体信息，找到数据卷相关
[root@yc_docker01 ~]# docker inspect chaoge_nginx

        "Mounts": [

            {
                "Type": "bind",
                "Source": "/my_docker_nginx",
```

```
                "Destination": "/usr/share/nginx/html",
                "Mode": "",
                "RW": true,
                "Propagation": "rprivate"
            }
        ],
```

为什么看到403，是因为宿主机的这个目录，没有内容

修改rw为ro，只读，禁止容器内修改数据

代码文件，配置文件，依赖全部打包为docker镜像

迭代程序，修改代码 ，只能重新构建镜像 ,很浪费时间，

通过挂载形式，直接修改宿主机的代码内容，直接重启容器，重新读取，加载代码，即可

```
docker run -d -p 9091:80 --name chaoge_nginx2 -v /my_docker_nginx:/usr/share/nginx/html:ro   nginx:1.12

# 你可以修改宿主机的内容

# 而不能修改容器内的内容

# 再次查看容器的信息
[root@yc_docker01 my_docker_nginx]# docker inspect chaoge_nginx2

"Mounts": [
            {
                "Type": "bind",
                "Source": "/my_docker_nginx",
                "Destination": "/usr/share/nginx/html",
                "Mode": "ro",
                "RW": false,
```

```
                "Propagation": "rprivate"
            }
        ],
```

volume模式

```
# 还是通过 -v --volume参数挂载

-v  容器内的目录

# 运行一个挂载的目录
[root@yc_docker01 my_docker_nginx]# docker run -p 8010:80 -d -v /usr/share/nginx/html  nginx:1.12
701e229e510ae79d5c9fc310fc1b2c72f8949d9c784b5951b14db61263f45480

# 查看容器信息
docker inspect 701

  "Mounts": [
            {
                "Type": "volume",
                "Name":
// 这个名字是docker自动生成的数据卷的名字
 "f668a66bec8e3067d0aa2bac69ee6da2cf68cc56a7317aa170de71b27fc33bb6",
                "Source":
"/var/lib/docker/volumes/f668a66bec8e3067d0aa2bac69ee6da2cf68cc56a7317aa170de71b27fc33bb6/_data",
                "Destination": "/usr/share/nginx/html",
                "Driver": "local",
                "Mode": "",
                "RW": true,
```

```
                "Propagation": ""
            }
        ],
```

添加额外的volume参数

```
# 第一个给宿主机的数据卷命名
# 设置只读
docker run -p 8011:80 --name chaoge_nginx02 -d -v
chaoge_docker_data:/usr/share/nginx/html:ro  nginx:1.12

# 查看容器详细信息
[root@yc_docker01 _data]# docker inspect chaoge_nginx02

        "Mounts": [
            {
                "Type": "volume",
                // 就是docker自动生成的数据卷的目录名字
                "Name": "chaoge_docker_data",
                "Source":
"/var/lib/docker/volumes/chaoge_docker_data/_data",
                "Destination": "/usr/share/nginx/html",
                "Driver": "local",
                "Mode": "ro",
                "RW": false,
                "Propagation": ""
            }
        ],

```

共享volume

在多个容器运行时，使用同一套数据卷，docker volume，让多个容器，都使用同一个volume的数据

-v 参数不同

> 使用共享volume的场景，例如：
>
> 运行2个nginx容器 ， 读取同一套html的数据，并且让容器内只读，只能够读取宿主机的volume数据

```
# 容器1
docker run -d  -p 7071:80 --name yuchao_nginx01 -v yuchao_nginx_html:/usr/share/nginx/html:ro  nginx:1.12

# 容器2
docker run -d  -p 7072:80 --name yuchao_nginx02 -v yuchao_nginx_html:/usr/share/nginx/html:ro  nginx:1.12
```

--volumes-from参数

读取某个容器的数据卷

```
# 运行的第一个容器
docker run -d -p 6061:80 --name cc_nginx01 -v cc_nginx_html:/usr/share/nginx/html nginx:1.12

# 第二个容器，直接去读另一个容器的数据卷
# 错误演示
docker run -d -p 6062:80 --name cc_nginx02  nginx:1.12

# 正确演示
# --volumes-from 容器的名字
docker run -d -p 6063:80 --name cc_nginx03  --volumes-from cc_nginx01   nginx:1.12
```

数据卷的查看，删除

```
# 查看当前机器的docker数据卷
docker volume ls

容器创建了数据卷，数据是持久化的，即使容器被删除了，数据卷还存在
当你确认数据无用，可以删除这些数据了

如果有数据卷被容器使用，它是禁止删除的

# 删除命令
# 必须删除使用这个数据卷的容器记录
docker volume rm 存储卷名

# docker提供命令，删除无容器在用的数据卷
# 清理无用数据卷
docker volume prune
```

```
16
17 [root@yc_docker01 _data]# docker volume prune
18 WARNING! This will remove all local volumes not used by at
   least one container.
19 Are you sure you want to continue? [y/N] y
20 Deleted Volumes:
21 chaoge_html
22
23 Total reclaimed space: 1.149kB
24
25
26
```