第二章：掌握Docker数据卷

Docker容器数据卷

容器存在的问题

- 容器删除后，在容器中产生的数据默认也会被删除
- 容器与外部主机无法直接交换文件
- 容器与容器之间无法相互交换数据

容器数据卷

- 数据卷可以是宿主机中的一个目录或文件，通过将目录或文件挂载到容器中实现容器数据与宿主机数据立即同步
- 数据卷可以解决容器删除后容器数据丢失的问题，实现数据持久化
- 数据卷可以间接的将外部主机数据传输到宿主机目录，解决容器与外部主机之间数据交换的问题
- 一个数据卷目录可以同时挂载多个容器，解决多容器之间数据交换的问题

配置数据卷

> 在创建启动容器时，使用-v参数设置数据卷
>
> docker run 参数 -v 宿主机目录/文件:容器内目录/文件...

注意事项

> 1.目录必须是绝对路径
>
> 2.如果目录不存在，会自动创建
>
> 3.可以挂载多个数据卷

配置容器数据卷

```
#创建容器并挂载数据卷
[root@localhost ~]# docker run -it --name=centos3 -v
/root/test:/root/test_container centos:7 /bin/bash

#容器内查看数据卷目录
[root@8cf8fcdb6e52 /]# ls /root
anaconda-ks.cfg  test_container

#验证数据同步，在宿主机数据卷目录创建文件
[root@localhost ~]# touch test/xxxx.txt
[root@localhost ~]# ls test/
xxxx.txt
```

```
#验证数据同步，容器查看数据是否同步
[root@8cf8fcdb6e52 /]# ls /root/test_container/
xxxx.txt

#验证数据同步，容器数据卷发生变化，宿主机也会立刻同步
[root@8cf8fcdb6e52 /]# touch /root/test_container/abc.txt
[root@8cf8fcdb6e52 /]# ls /root/test_container/
abc.txt  xxxx.txt

#验证数据同步，宿主机验证数据是否同步
[root@localhost ~]# ls test/
abc.txt  xxxx.txt

#退出并删除容器
[root@8cf8fcdb6e52 /]# exit
[root@localhost ~]# docker rm centos3

#宿主机验证数据卷目录数据是否存在
[root@localhost ~]# ls test/
abc.txt  xxxx.txt

#重新创建容器，并将数据卷目录同步到容器中
[root@localhost ~]# docker run -it --name=centos3 -v
/root/test:/root/test_container centos:7 /bin/bash

#容器验证数据是否同步
[root@9a48ae7d7eb3 /]# ls /root/
anaconda-ks.cfg  test_container
[root@9a48ae7d7eb3 /]# ls /root/test_container/
abc.txt  xxxx.txt
```

容器挂载多个数据卷

```
#创建容器并同时挂载多个数据卷目录
[root@localhost ~]# docker run -it --name=centos4 \
> -v /root/test:/root/test \
> -v /root/test1:/root/test1 \
> centos:7

#容器查看数据卷目录
[root@2e975255d526 /]# ls /root
anaconda-ks.cfg  test  test1
```

多容器挂载同一个数据卷

```
#创建容器并挂载数据卷
[root@localhost ~]# docker run -it --name=centos5 -v
/root/test:/root/test_container centos:7

#创建容器并挂载数据卷
[root@localhost ~]# docker run -it --name=centos6 -v
/root/test:/root/test_container centos:7
```

```
#在容器数据卷创建文件
[root@dec4ecc23b61 test_container]# touch test.txt
[root@dec4ecc23b61 test_container]# ls
abc.txt  test.txt  xxxx.txt

#另一个容器验证数据是否同步
[root@448a180cc6c8 ~]# ls test_container/
abc.txt  test.txt  xxxx.txt

#宿主机验证数据是否同步
[root@localhost ~]# ls test
abc.txt  test.txt  xxxx.txt
```

数据卷总结：

> 1. 把容器内的相应目录的数据持久化到宿主机
> 2. 数据卷可以间接的将外部主机数据传输到宿主机目录，解决容器与外部主机之间数据交换的问题
> 3. 一个数据卷目录可以同时挂载多个容器，解决多容器之间数据交换的问题

第三章：掌握Docker应用部署

1.Docker部署MySQL

> 在Docker容器中部署MySQL数据库，并通过外部MySQL客户端管理MySQL数据库

```
#下载MySQL镜像文件（不了解版本的可以从Docker Hub上提前查看MySQL的版本信息）
[root@localhost ~]# docker pull mysql:5.7

#在宿主机创建目录用于存储MySQL数据
[root@localhost ~]# mkdir -p /mysql/conf
[root@localhost ~]# mkdir  /mysql/logs

#创建容器，提前将所需的配置从容器中拷贝到宿主机，所需的目录或文件就是传统部署方式的所在位置，所以需要有传统部署方式的基础
cp 用于将宿主机文件或目录拷贝到容器中，也可将容器中的目录或文件拷贝到宿主机中
命令格式：  docker cp 容器名:目录/文件   宿主机目录        #将容器中文件或目录拷贝到宿主机
命令格式：  docker cp 宿主机目录/文件   容器名:目录        #将宿主机文件/目录拷贝到容器中
[root@localhost ~]# docker run -it --name=mysql mysql:5.7 /bin/bash  #创建容器
[root@localhost ~]# docker cp mysql:/etc/mysql /mysql/conf            #拷贝配置文件目录
[root@localhost ~]# docker cp mysql:/var/log/mysql /mysql/logs        #拷贝日志目录

#删除mysql容器重新创建并挂载数据卷
[root@localhost ~]# docker stop mysql
[root@localhost ~]# docker rm mysql
[root@localhost mysql]# docker run -id -p 3306:3306 --name=mysql \   #映射端口
 -v /mysql/conf:/etc/mysql/ \      #映射配置文件目录
```

```
 -v /mysql/logs:/var/log/mysql \          #映射日志目录
 -v /mysql/data:/var/lib/mysql \          #映射数据目录（不需要提前拷贝，访问数据库时会自动产生数据）
 -e MYSQL_ROOT_PASSWORD=123456 \          #指定数据库密码
 mysql:5.7

#进入MySQL容器
[root@localhost mysql]# docker ps
[root@localhost mysql]# docker exec -it mysql /bin/bash

#进入容器中的MySQL数据库
root@43f0946f3e6c:/# mysql -uroot -p123456
mysql>
mysql> create database db1;
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| db1                |
| mysql              |
| performance_schema |
| sys                |
+--------------------+
5 rows in set (0.00 sec)

#宿主机查看数据卷目录
[root@localhost ~]# ll /mysql/data
```

2.Docker部署tomcat

> 在Docker容器中部署tomcat，并通过外部机器访问tomcat部署的项目

```
#下载tomcat镜像文件（不了解版本的可以从Docker Hub上提前查看tomcat的版本信息）
[root@localhost ~]# docker pull tomcat:10

#在宿主机创建数据卷目录用于存储tomcat容器数据
[root@localhost ~]# mkdir /tomcat

#创建容器，拷贝所需数据到宿主机
[root@localhost ~]# docker run -id --name=tomcat tomcat:10 /bin/bash
[root@localhost ~]# docker cp tomcat:/usr/local/tomcat /tomcat

#停止tomcat容器并删除
[root@localhost ~]# docker stop tomcat
[root@localhost ~]# docker rm tomcat

#创建容器，设置端口映射、目录映射
[root@localhost ~]# docker run -id --name=tomcat -p 8080:8080 \
-v /tomcat/tomcat:/usr/local/tomcat \
tomcat:10

#宿主机创建tomcat网页目录
[root@localhost ~]# mkdir /tomcat/tomcat/webapps/test
```

```
#创建测试页面
[root@localhost ~]# vim /tomcat/tomcat/webapps/test/index.html
<h1>hello tomcat<h1>

#外部主机通过浏览器访问容器tomcat服务
http://192.168.0.41:8080/test/
```

3.Docker部署nginx

> 在Docker容器中部署nginx，并通过外部主机访问nginx服务

```
#下载nginx镜像文件（不了解版本的可以从Docker Hub上提前查看nginx的版本信息）
#宿主机创建nginx数据卷目录
[root@localhost ~]# mkdir /nginx

#在数据卷目录创建conf目录用于存放nginx的主配置文件
[root@localhost ~]# mkdir /nginx/conf

#创建容器，拷贝所需数据到宿主机（nginx只需要拷贝配置文件即可，其他数据创建容器后自动生成）
[root@localhost ~]# docker cp nginx:/etc/nginx/nginx.conf /nginx/conf

#创建容器，设置端口映射、目录与配置文件映射
[root@localhost ~]# docker run -id --name=nginx -p 80:80 \    #映射端口
-v /nginx/conf/nginx.conf:/etc/nginx/nginx.conf \    #映射配置文件
-v /nginx/logs:/var/log/nginx \                      #映射日志目录
-v /nginx/html:/usr/share/nginx/html                 #映射网页根目录

#宿主机在/html数据卷目录为nginx创建测试
[root@localhost ~]# vim /nginx/html/index.html
hello nginx

#外部主机通过浏览器访问nginx服务
http://192.168.0.41/
```

4.Docker部署Prometheus+Gfafana监控

```
#下载prometheus镜像
[root@localhost ~]# docker search prometheus
[root@localhost ~]# docker pull prom/prometheus

#创建数据卷目录
[root@localhost ~]# mkdir /prom

#创建容器拷贝prometheus配置文件
[root@localhost ~]# docker run -id --name=prom prom/prometheus

#拷贝配置文件到宿主机目录
[root@localhost ~]# docker cp prom:/etc/prometheus/prometheus.yml /prom

#停止容器
```

```
[root@localhost ~]# docker stop prom

#删除容器
[root@localhost ~]# docker rm prom

#创建Prometheus容器并挂载数据卷
[root@localhost ~]# docker run -id --name=prom -p 9090:9090 \
-v /prom/prometheus.yml:/etc/prometheus/prometheus.yml \
prom/prometheus

#查看容器状态并访问Prometheus
[root@localhost ~]# docker ps

http://192.168.0.24:9090

#下载node-exporter用于获取主机指标数据
[root@localhost ~]# docker pull prom/node-exporter

#创建node-exporter容器
[root@localhost ~]# docker run -id --name=node-export -p 9100:9100 prom/node-
exporter

#修改Prometheus.yml配置文件定义监控
[root@localhost ~]# vim /prom/prometheus.yml
...
  - job_name: "node-exporter"
    static_configs:
      - targets: ["192.168.0.24:9100"]

#重启Prometheus
[root@localhost ~]# docker restart prom

#创建grafana容器
[root@localhost ~]# docker run -id --name=grafana -p 3000:3000 grafana/grafana

#访问grafana添加数据源与监控模板
http://192.168.0.24:3000

#部署cadvisor（开源的容器度量和可视化系统）用于获取容器内的指标数据
具体部署地址可参考：https://github.com/google/cadvisor

#下载cadvisor镜像
[root@localhost ~]# docker pull google/cadvisor

#创建cadvisor容器
[root@localhost ~]# docker run \
--volume=/:/rootfs:ro \
--volume=/var/run:/var/run:ro \
--volume=/sys:/sys:ro \
--volume=/var/lib/docker/:/var/lib/docker:ro \
--volume=/dev/disk/:/dev/disk:ro \
--publish=8888:8080 \
--detach=true \
--name=cadvisor \
google/cadvisor

#访问cadvisor
http://192.168.0.24:8888
```

```
#cadvisor为prometheus提供了时序数据格式（该格式为prometheus识别的格式）
http://192.168.0.24:8888/metrics

#修改promethes.yml文件采集cadvisor数据
[root@localhost ~]# vim /prom/prometheus.yml
...
  - job_name: "docker"
    static_configs:
      - targets: ["192.168.0.24:8888"]

#重启prometheus容器
[root@localhost ~]# docker restart prom

#为grafana导入容器监控模板
具体模板可参考地址：https://grafana.com/grafana/dashboards/?search=docker
```