docker 2.容器

什么是容器?

镜像（Image）和容器（Container）的关系，就像是面向对象程序设计中的**类（安装包）**和**实例（应用程序）**一样，镜像是静态的定义，容器是镜像运行时的实体。容器可以被创建、启动、停止、删除、暂停等。

容器的实质是进程，但与直接在宿主执行的进程不同，容器进程运行于属于自己的独立的 命名空间。因此容器可以拥有自己的 `root` 文件系统、自己的网络配置、自己的进程空间，甚至自己的用户 ID 空间。容器内的进程是运行在一个隔离的环境里，使用起来，就好像是在一个独立于宿主的系统下操作一样。这种特性使得容器封装的应用比直接在宿主运行更加安全。

前面讲过镜像使用的是分层存储，容器也是如此。每一个容器运行时，是以镜像为基础层，在其上创建一个当前容器的存储层，我们可以称这个为容器运行时读写而准备的存储层为**容器存储层**。

容器存储层的生存周期和容器一样，容器消亡时，容器存储层也随之消亡。因此，任何保存于容器存储层的信息都会随容器删除而丢失。

按照 Docker 最佳实践的要求，容器不应该向其存储层内写入任何数据，容器存储层要保持无状态化。所有的文件写入操作，都应该使用 数据卷（Volume）、或者绑定宿主目录，在这些位置的读写会跳过容器存储层，直接对宿主(或网络存储)发生读写，其性能和稳定性更高。

数据卷的生存周期独立于容器，容器消亡，数据卷不会消亡。因此，使用数据卷后，容器可以随意删除、重新 `run`，数据却不会丢失。

容器的启动方式:

1.docker run 启动一个容器 (docker run -ti centos /bin/bash)
2.docker-compose 批量启动容器

容器常用操作相关命令:

attach 接管容器输入输出等流（docker attach xxx）

commit 从容器打包出一个新的镜像，包括容器文件的修改(docker commit xxx)

cp 宿主机跟容器间相互拷贝文件(docker cp [OPTIONS] CONTAINER:SRC_PATH DEST_PATH)

create 创建容器(docker create xxx)

run 创建并启动容器(docker run xxx)

exec 用容器执行命令 (docker exec xxx echo 123456)

export 导出容器(docker export xxx)

inspect 查看容器的配置(docker inspect xxx)

kill kill掉容器(docker kill xxx)

logs 查看容器的日志(docker logs xxx)

pause 挂起容器(docker pause xxx)

port 查看容器映射端口(docker port xxx)

ps 查看有多少容器启动了(docker ps)

rename 重命名一个容器(docker rename xxx newname)

restart 重启容器(docker restart xxx)

rm 移除容器(docker rm xxx)

start 启动容器(docker start xxx)

stats 查看容器的资源使用情况(docker stats xxx)

stop 停掉容器(docker stop xxx)

top 查看容器进程相关信息(docker top xxx)

unpause 将挂起的容器恢复(docker unpause xxx)

update 更新容器配置(docker update)

容器的启动，停止，移除:

docker run --name centos_test1 -ti centos /bin/bash
docker stop centos_test1
docker rm centos_test1

数据卷映射:

docker run --name centos_test1 -ti -v ~/Desktop/docker-share/data/container:/test/ centos /bin/bash

端口映射:

docker run --name mysql_test1 -p 3310:3306 -e MYSQL_ROOT_PASSWORD=root 192.168.1.253:5000/mysql:5.6.36

容器日志：

docker logs mysql_test1