H2 一、Docker介绍

H3 1.1 Docker解决痛点

> 1. 我本地运行没问题啊【环境不一致】。
> 2. 哪个哥们又写死循环了，怎么这么卡【在多用户的操作系统下，会相互影响】
> 3. 淘宝在双11的时候，用户量暴增【大量增加实体机，运维成本过高问题】
> 4. 学习一门技术，学习安装成本过高【关于安装软件成本过高】

H3 1.2 Docker的思想

> 1. 集装箱：
>
>   会将所有需要的内容放到不同的集装箱中，谁需要这些环境就直接拿到这个集装箱就可以了
>
> 2. 标准化：
>
>   - 运输的标准化：Docker有一个码头，所有上传的集装箱都放在了这个码头上，当谁需要某一个环境，就直接指派大海豚搬运这个集装箱就可以了
>   - 命令的标准化：Docker提供了一系列的命令，帮助我们去获取集装箱等等操作。
>   - 提供了REST的API：衍生出了很多的图形化界面，如Rancher
>
> 3. 隔离型：
>
>   Docker在运行集装箱内的内容时，会在Linux的内核中，单独的开辟一片空间，这片空间不会影响到其他程序。

> 1. 注册中心：超级码头，上面放的是集装箱
> 2. 镜像：集装箱
> 3. 容器：运行起来的镜像

H2 二、Docker的基本操作

H3 2.1 Docker的中央仓库【注册中心】

> 1. Docker官方的中央仓库：这个仓库是镜像最全的，但是下载速度较慢
>
>   https://hub.docker.com/
> 2. 国内的网站：网易蜂巢，daoCloud...
>
>   https://c.163yun.com/hub#/home
>
>   http://hub.daocloud.io/ 【推荐使用】
> 3. 在公司内部会采用私服的方式拉去镜像【需要添加配置】

```
# 需要在/etc/docker/daemon.json
{
  "registry-mirrors":["https://registry.docker-cn.com"]
  "insecure-registries":["ip:port"]
}
# 重启两个服务
systemctl daemon-reload
systemctl restart docker
```

H3 2.2 镜像的操作

```
# 1、拉取镜像到本地
docker pull 镜像名称[:tag]       # 如果不写tag版本号会拉去默认版本
# example
docker pull daocloud.io/library/mysql:5.7.4
```

```
# 2、查看本地所有镜像
docker images
```

```
# 3、删除本地镜像
docker rmi 镜像的唯一标识【IMAGE ID】
```

```
# 4、镜像的导入导出（不规范）
# 将本地的镜像导出
docker save -o 导出的路径 镜像id
# 加载本地的镜像文件
docker load -i 镜像文件
# 修改镜像名称
docker tag 镜像id 新镜像名称:版本
```

2.3 容器的操作

```
# 1、运行容器
# 简单操作，采用"镜像名称[:tag]"时，如果有则运行，如果没有则先下载在运行
docker run 镜像标识|镜像名称[:tag]
# 常用的参数
docker run -d -p 宿主机端口:容器端口 --name 容器名称 镜像标识|镜像名称[:tag]
# -d：代表后台运行容器
# -p：宿主机端口:容器端口：为了映射当前Linux的端口和容器的端口
# --name 容器名称：指定容器的名称
```

```
# 2、查看正在运行的容器
docker ps [-qa]
# -a：查看全部的容器，包括没有运行
# -q：只查看容器的标识不查看其他信息
```

```
# 3、查看容器的日志
docker logs -f 容器id
# -f：可以滚动查看日志的最后几行
```

```
# 4、进入到容器内部
docker exec -it 容器id bash
```

```
# 5、删除容器（删除容器前，需要先停止容器）
# 停止制动的容器
docker stop 容器id
# 停止全部的容器
docker stop $(docker ps -qa)
# 删除指定容器
docker rm 容器id
# 删除全部容器
docker rm $(docker ps -qa)
```

```
# 6、启动容器
docker start 容器id
```

三、Docker的应用

3.1 准备SSM工程

> MySQL数据库的连接用户名和密码改变了，修改db.properties

3.2 准备MySQL容器

```
# 运行MySQL容器
docker run -d -p 3306:3306 --name mysql -e MYSQL_ROOT_PASSWORD=root
daocloud.io/library/mysql:5.7.4
```

3.3 准备Tomcat容器

```
# 运行Tomcat容器，前面已经搞定，只需要将SSM项目的war包部署到Tomcat内部即可
# 可以通过命令将宿主机的内容复制到容器内部
docker cp 文件名 容器id:容器内部路径
# 举个离职
docker cp ssm.war fe:/user/local/tomcat/webapps
```

3.4 数据卷

> 为了部署SSM的工程，需要使用到cp的命令将宿主机内的ssm.war文件复制到容器内部【由于新下载的容器没有ssm.war】但是宿主机的ssm.war配置文件改后还需要重新上传容器，或者是在容器中修改，太麻烦
>
> 可以用数据卷解决这个问题
>
> 数据卷：将宿主机的一个目录映射到容器的一个目录中
>
> 可以在宿主机中操作目录中的内容，那么容器内部映射的文件，也会跟着一起改变

```
# 1. 创建数据卷
docker volume create 数据卷名称
# 创建数据卷之后，默认会存放在一个目录下 /var/lib/docker/volumes/数据卷名
称/_data
```

```
# 2. 查看数据卷的详细信息
docker volume inspect 数据卷名称
```

```
# 3. 查看全部数据卷
docker volume ls
```

```
# 4. 删除数据卷
docker volume rm 数据卷名称
```

```
# 5. 应用数据卷
# 当你映射数据卷时，如果数据卷不存在，Docker会帮你自动创建,会将容器内部自带的文件,存储在默认的存放路径中
docker run -v 数据卷名称:容器内部的路径 镜像id
# 直接指定一个路径作为数据卷的存放位置【推荐】
docker run -v 路径:容器内部的路径 镜像id
```

四、Docker自定义镜像

> 中央仓库上的镜像，也是Docker的用户自己上传过去的。