docker基础

1.1 引言

```
1. 我本地运行没问题
    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。
    运维成本过高的问题。
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。
```

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:
    1. 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，当谁需要某些环境，就直接指派鲸鱼去搬运这个集装箱就可以了。
    2. 命令的标准化：docker提供了一系列的命令，去帮助我们获取集装箱等等操作。
    3. 提供了rest的api：衍生出了很多图形化的界面，rancher。
3. 隔离性:
    docker在运行集装箱的内容时，会在Linux的内核中，单独开辟一片空间，这片空间不会影响到其他程序。
```

- 注册中心。（超级码头，上面放的就是集装箱）
- 镜像。（集装箱）
- 容器。（运行起来的镜像）

1.3 镜像的操作

```
#1. 拉取镜像到本地
docker pull 镜像名称[:tag]
docker pull daocloud.io/library/tomcat:8.5.15-jre8
#2. 查看全部本地的镜像
docker images
#3. 删除本地镜像
docker rmi 镜像的标识
# 4. 镜像的导入导出
# 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
dockerload -i 镜像文件
# 修改镜像名称
docker tag 镜像id 新镜像名称:版本
```

1.4 容器的操作

```
#1. 运行容器
# 简单操作
docker run 镜像的标识|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 镜像的标识|镜像名称[:tag]
# -d 代表后台运行
# -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]
#-q: 查看全部容器，包括没有运行的
#-a: 只查看容器的标识
#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. 进入到容器的内部
docker exec -it 容器id bash
#5. 删除容器（删除容器前， 需要先停止容器）
docker stop 容器id
docker stop $(docker ps -qa)
docker rm 容器id
docker rm $(docker ps -qa)
#6. 启动容器
docker start 容器id
```

1.5 数据卷

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。
>
> 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
```

选择语言

```sh
# 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。这个路径下是空的。
docker run -v 路径:容器内部的路径 镜像id
```

sh

应用

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。

```sh
# 1. 创建一个Dockerfile文件，并且指定自定义镜像信息。
# Dockerfile文件中常用的内容
from:  指定当前自定义镜像依赖的环境
copy:  将相对路径下的内容复制到自定义镜像中
workdir:  声明镜像的默认工作目录
cmd:  需要执行的命令（在workdir下执行的，cmd可以写多个，只以最后一个为准）
# 举个栗子，自定义一个tomcat镜像，并且将ssm.war部署到tomcat中
from daocloud.io/library/tomcat:8.5.15-jre8
copy ssm.war /usr/local/tomcat/webapps
```

```sh
# 2. 将准备好的Dockerfile和相应的文件拖拽到Linux操作系统中，通过Docker的命令制作镜像
docker build -t 镜像名称[:tag] .
```

6.1 下载Docker-Compos\

```sh
# 1. 去github官网搜索docker-compose，下载1.24.1版本的Docker-Compose
https://github.com/docker/compose/releases/download/1.24.1/docker-compose-Linux-x86_64

# 2. 将下载好的文件，拖拽到Linux操作系统中
。。。

# 3. 需要将DockerCompose文件的名称修改一下，基于DockerCompose文件一个可执行的权限
mv docker-compose-Linux-x86_64 docker-compose
chmod 777 docker-compose

# 4. 方便后期操作，配置一个环境变量
# 将docker-compose文件移动到了/usr/local/bin ， 修改了/etc/profile文件，给/usr/local/bin配置到了PATH中
mv docker-compose /usr/local/bin
vi /etc/profile
    export PATH=$JAVA_HOME:/usr/local/bin:$PATH
source /etc/profile
```

```
# 5. 测试一下
# 在任意目录下输入docker-compose
```

```
[root@localhost ~]# docker-compose
Define and run multi-container applications with Docker.

Usage:
  docker-compose [-f <arg>...] [options] [COMMAND] [ARGS...]
  docker-compose -h|--help

Options:
  -f, --file FILE             Specify an alternate compose file
                              (default: docker-compose.yml)
  -p, --project-name NAME     Specify an alternate project name
```

6.2 Docker-Compose管理MySQL和Tomcat容器

> yml文件以key: value方式来指定配置信息
>
> 多个配置信息以换行+缩进的方式来区分
>
> 在docker-compose.yml文件中，不要使用制表符

```yaml
version: '3.1'
services:
  mysql:                # 服务的名称
    restart: always     # 代表只要docker启动，那么这个容器就跟着一起启动
    image: daocloud.io/library/mysql:5.7.4   # 指定镜像路径
    container_name: mysql   # 指定容器名称
    ports:
      - 3306:3306    #  指定端口号的映射
    environment:
      MYSQL_ROOT_PASSWORD: root    # 指定MySQL的ROOT用户登录密码
      TZ: Asia/Shanghai            # 指定时区
    voulmes:
      - /opt/docker_mysql/data:/var/lib/mysql    # 映射数据卷
```

```yaml
tomcat:
  restart: always
  image: daocloud.io/library/tomcat:8.5.15-jre8
  container_name: tomcat
  ports:
    - 8080:8080
  environment:
    TZ: Asia/Shanghai
  volumes:
    - /opt/docker_mysql_tomcat/tomcat_webapps:/usr/local/tomcat/webapps
    - /opt/docker_mysql_tomcat/tomcat_logs:/usr/local/tomcat/logs
```

yml

6.3 使用docker-compose命令管理容器

> 在使用docker-compose的命令时，默认会在当前目录下找docker-compose.yml文件

```
# 1. 基于docker-compose.yml启动管理的容器
docker-compose up -d
```

```
# 2. 关闭并删除容器
docker-compose down
```

```
# 3. 开启|关闭|重启已经存在的由docker-compose维护的容器
docker-compose start|stop|restart
```

```
# 4. 查看由docker-compose管理的容器
docker-compose ps
```

```
# 5. 查看日志
docker-compose logs -f
```

6.4 docker-compose配置Dockerfile使用

> 使用docker-compose.yml文件以及Dockerfile文件在生成自定义镜像的同时启动当前镜像，并且由docker-compose去管理容器

docker-compose.yml

```yml
# yml文件
version: '3.1'
services:
  ssm:
    restart: always
    build:                  # 构建自定义镜像
      context: ../          # 指定dockerfile文件的所在路径
      dockerfile: Dockerfile    # 指定Dockerfile文件名称
    image: ssm:1.0.1
    container_name: ssm
    ports:
      - 8081:8080
    environment:
      TZ: Asia/Shanghai
```

yml

Dockerfile文件

```
from daocloud.io/library/tomcat:8.5.15-jre8
copy ssm.war /usr/local/tomcat/webapps
```

```sh
# 可以直接启动基于docker-compose.yml以及Dockerfile文件构建的自定义镜像
docker-compose up -d
# 如果自定义镜像不存在，会帮助我们构建出自定义镜像，如果自定义镜像已经存在，会直接运行这个自定义镜像
# 重新构建的话。
# 重新构建自定义镜像
docker-compose build
# 运行前，重新构建
docker-compose up -d --build
```

sh

七. Docker CI、CD

7.1 引言

> 项目部署
>
> 1. 将项目通过maven进行编译打包
> 2. 将文件上传到指定的服务器中
> 3. 将war包放到tomcat的目录中
> 4. 通过Dockerfile将Tomcat和war包转成一个镜像，由DockerCompose去运行容器
>
> 项目更新了
>
> 将上述流程再次的从头到尾的执行一次

7.5 实现持续交付持续部署

7.5.1 安装Jenkins

> 官网：https://www.jenkins.io/

```yaml
version: "3.1"
services:
  jenkins:
    image: jenkins/jenkins
    restart: always
    container_name: jenkins
    ports:
      - 8888:8080
      - 50000:50000
    volumes:
      - ./data:/var/jenkins_home
```

yaml

> 第一次运行时，会因为data目录没有权限，导致启动失败

```
chmod 777 data
```

> 访问http://192.168.199.109:8888

访问速度奇慢无比。。。。。

> 输入密码

```
jenkins    |
jenkins    | Jenkins initial setup is required. An admin user has been created and a password generated.
jenkins    | Please use the following password to proceed to installation:
jenkins    |
jenkins    | cdafe9d7cf154ed2885e30da67571729
jenkins    |
jenkins    | This may also be found at: /var/jenkins_home/secrets/initialAdminPassword
jenkins    |
```

> 手动指定插件安装
>
> publish ssh...
>
> git param...