## 1.什么是Docker？

Java：一次编译，到处运行在JVM虚拟机上;

Docker：真正实现了一次编译,到处运行;

Docker是一个开源的应用容器引擎,基于GO语言;

Docker可以让开发者打包他们的应用到一个轻量级、可移植的容器中,然后发布到任何流行的Linux机器上,也可以实现虚拟化;

容器完全使用沙箱机制，互相之间不会有任何接口(类似手机中的App),更重要的是容器性能开销极低;

## 2.为什么要用Docker？

Docker支持将软件编译成一个镜像;然后在 镜像中各种软件做好配置，将镜像发布出去， 其他使用者可以直接使用这个镜像。运行中

的这个镜像成为容器,容器的启动速度是 非常快速的；

1.作为一种新兴的虚拟化方式,Docker和传统的虚拟化方式相比有众多的优势;

### a.更高效利用系统资源;

由于容器不需要进行硬件虚拟以及运行操作系统等额外开销，Docker对系统资源的利用率更高。无论是应用执行速度、内存损耗或者文件存储速度，都要比传统虚拟机技术更高效。因此，相比虚拟机技术，一个相同配置的主机,往往可以运行更多数量的应用。

### b.更快的启动时间;

传统的虚拟机技术启动应用服务往往需要数分钟，而Docker容器应用，由于直接运行在宿主内核，无需启动完整的操作系统,因此可以做到秒级、甚至毫秒级的启动时间。大大节约了开发、测试、部署时间;

### c.一致的运行环境;

开发中一个常见的问题是环境一致性问题。由于开发环境、测试环境、生产环境不一致，导致有些bug并未在开发过程中被发现。而Docker的镜像提供了除内核外完整的运行时环境，确保了应用环境一致性。

### d.持续交付和部署;

对开发和维护人员来说，最希望的是一次创建或配置，可以在任意地方正常运行，使用Docker可以通过定制应用镜像来实现持续集成、持续交付、部署。开发任意可以通过Dockerfile来进行镜像构建，并结合集成系统进行集成测试，而运维人员可以直接在生产环境中快速部署改镜像，甚至结合持续部署系统进行自动部署。而且使用Dockerfile使镜像构建透明化，不仅仅开发团队可以理解应用运行环境，也方便运维团队理解应用所运行所需要的条件，帮助更好的在生产环境中部署该镜像。

### e.更轻松的迁移;

由于Docker确保了执行环境的一致性，使得应用的迁移更加容易。Docker可以在很果断平台上运行，无论是物理机、虚拟机、公有云、私有云，甚至笔记本，其运行结果是一致的，因此用户可以轻易的将一个平台上的应用,迁移到另外一个平台上，而不用担心运行环境的变化导致应用无法正常运行。

### f.更加轻松的维护和扩展;

Docker使用了分层存储以及镜像技术,使得应用重复部分的复用更为容易，也使得应用的维护更新更加的简单,基于基础镜像进一步扩展镜像也非常简单。此外，Docker团队和各个开源项目团队维护了一大批高质量的官方镜像,既可以直接在生产环境中使用,也可以作为基础进一步定制,大大降低了应用服务镜像制作成本;

## 3.Docker的应用场景

### 1.应用程序的打包和发布

### 2.应用程序隔离

### 3.自动化测试、持久集成、发布

### 4.部署微服务

### 5.快速搭建测试环境

### 6.提供PaaS平台级产品

## 4.Docker和虚拟机的区别

**1.物理机模型:**

相当于一栋楼一户人家，独立地基，独立花园;

**2.虚拟机模型：**

相当于一栋楼包含了多套房，一套房一户人家，共享基地，共享花园，独立卫生间，厨房、宽带;

**3.Docker 模型:**

一套房隔成多个隔间,每个隔间一个租户,共享基地，共享花园，共享卫生间，厨房、宽带;

## 5. Docker的核心概念

a.**Docker主机(host)**:安装了docker程序的机器(Docker直接安装在操作系统之上)  
b**.Docker客户端（client）**：连接docker主机进行操作;  
c.**Docker仓库(registry)**：用来保存各种打包好的软件镜像;  
d.**Docker镜像(Images)** :软件打包好的镜像,放在docker仓库;  
e.**Docker容器(container)**：镜像启动后的实例称为一个容器,容器是独立运行的一个或一组应用;

## 6.使用Docker的步骤

1.安装Docker;  
2.去Docker仓库找到需要安装的软件对应的镜像;  
3.使用Docker运行该软件的镜像,这个镜像镜像会产生一个Docker容器,容器就代表这个当前正在运行的应用;  
4.对容器的启动停止就是对软件的启动停止;

## 7.在Linux（cetos7）上安装Docker

1.查看Linux系统的内核版本

必须是3.10及以上

**uname -r**

2.升级软件包及内核;(选做)

**yum update**

3.安装 docker

**yum install docker**

查看docker版本号

**docker -v**

4.启动docker

**systemctl start docker**

5.将docker服务设为开机启动

**systemctl enable docker**

## 8.Docker相关命令

1.停止Docker命令

**systemctl stop docker**

## 9. Docker镜像操作常用命令

**1.检索**

命令：docker search 关键字

eg:docker search redis 说明：我们经常去docker hub上检索镜像的详细信息,如镜像的Tag;

默认去 docker hub 仓库搜索镜像; <https://hub.docker.com/>

**2.拉取**

命令：docker pull 镜像名:tag

说明：:tag是可选的,tag表示标签，多为软件的版本,默认为latest

**3.列表**

命令：docker images

说明：查看所有的本地镜像

**4.删除**

命令：docker rmi image-id

说明：删除指定的本地镜像

## 10. Docker的容器操作

软件镜像–运行镜像—产生一个容器（正在运行的软件）

**a.搜索镜像**

docker search tomcat

**b.拉取镜像**

docker pull tomcat

**c.根据镜像启动容器**

docker run --name mytomacat -d tomcat:latest

**1.运行**

命令：docker run --name container-name -d image-name

eg: docker run --name myredis -d redis

说明：–name:自定义容器名;

-d:后台运行;

image-name:指定镜像模板;

**2.列表**

命令：docker ps(查看运行中的容器) 说明：加上-a 可以查看所有容器

**3.停止：**

命令：docker stop container-name/container-id 说明：停止当前运动的容器

**4：启动**

命令： docker start container-name/container-id

说明：启动容器

**5 删除**

命令：docker rm container-id

说明：删除指定容器

**6端口映射:**

命令：-p 6379:6379 主机端口：容器内部端口

eg: docker run -d -p 6379:6379 -name myredis docker.io/redis

说明：-p:主机端口(映射到)容器内部的端口

**7 容器日志：**

命名： docker logs container-name/container-id

## 11.容器操作相关操作说明

**docker run**

-i:表示运行容器

-t:表示容器启动后会进入命令行,加上这个参数后，容器创建就能登录进云，及分配了一个伪终端.

交互式容器：docker run -it --name -mycentos centos:7 /bin/bash

该类型容器退出容器之后容器就停止.

守护式容器：docker run -di --name -mycentos2 centos:7

登录守护式容器:cocker exec -it 容器名称 /bin/bash

该类型的容器退出后不会停止容器；

-d:创建一个守护式容器在后台一直运行

-p:端口映射 指定容器暴露的端口 9000:9000 （前者是宿主机的端口:后者是容器的端口）

--restart:指定容器重启后的策略

no:容器退出不重启;

no-failure:容器故障退出（返回值非零）时重启；

always:容器退出总是重启；

-v：给容器挂载存储卷，挂载到容器的某个目录 宿主机目录：容器目录

--name:指定容器名字,后续可以通过名字进行容器管理，links特性需要使用名字

--rm=false,指定容器停止后自动删除容器(不支持以docker run -d启动的容器)

为什么要做目录映射：因为容器里的系统不能直接去上传文件;

## 12.安装DockerUI：portainer

1.查看DockerUI:portainer镜像;

docker search portainer

2.拉取portainer镜像

docker pull docker.io/portainer/portainer

3.启动portainer

docker run -d -p 9000:9000 --restart=always -v /var/run/docker.sock:/var/run/docker.sock --name ben-portainer portainer/portainer

## 13.Docker安装软件遇到的问题

### 1. 腾讯云CentOS安装软件, 外部无法访问;

a.解决方案:

腾讯云默认情况下只允许ssh的22端口通过，其余端口一律封锁,解决方法就是进到腾讯云控制台：选择安全组：开放端口;

b.相关文档: <https://blog.csdn.net/weixin_38423249/article/details/80554583>

### 2.安装tomcat访问主页404;

a.解决方案：

访问不到主页的原因是因为存放主页的文件夹webapps是空的,docker的tomcat将这些文件放在了 webapps.dist文件夹中,我们只需要将空的webapps文件夹删除,然后将ebapps.dist重命名为webapps就可以了;

b.相关文档:

<https://www.cnblogs.com/chzushl/p/13385580.html>

## 14.安装MYSQL

## 15.安装Redis

## 16.Docker部署Springboot项目

1.0 在Linux上只安装Docker;

1.切换到宿主机 :/usr/local/docker 如果没有docker就创建一个新的目录mkdir docker;

2.创建一个存放项目的文件夹 mkdir itour;

3.进入文件夹 cd /usr/local/docker/itour/

4.使用rz命令 上传springboot项目的jar;

如果找不到该命令，安装rz命令：yum install lrzsz

5.新建Dockerfile文件

Dockerfile文件,类似于window中bat文件，可以执行一些命令;

#Docker image for springboot file run

#基础镜像使用java

FROM java:8

#作者

MAINTAINER wangtao <wwangtaoc11@163.com>

#指定临时文件目录为/tmp

VOLUME /tmp

#将jar包添加到容器中并更名为itour.jar

ADD itour-online-web.jar itour.app

#运行jar

ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/itour.jar"]

6.构建容器：

docker build -t itour .

7.启动容器

docker run -name itour -p 8899:8899 itour

8.运行成功及可以访问

相关文档:

<https://www.jianshu.com/p/397929dbc27d>

<https://www.imooc.com/article/44582>

Dockerfile相关文档：

<https://www.runoob.com/docker/docker-dockerfile.html>

<https://blog.csdn.net/cxzhq2002/article/details/91371225>

<https://www.cnblogs.com/edisonchou/p/dockerfile_inside_introduction.html>

<https://www.cnblogs.com/panwenbin-logs/p/8007348.html>

## 17.安装Tomcat

1.拉取tomcat镜像

docker pull tomcat

2.运行镜像(创建一个守护式容器（如果对一个需要长期运行的容器来说，我们可以创建一个守护式容器,容器的名称不能重复）)

docker run -d --name tomcat-8080 -p 8080:8080 -v /usr/local/webapps/:/usr/local/tomcat/webapps/ tomcat

3.进入守护式容器内部;

docker exec -it tomcat-8080 bash

3.1切换到/usr/local/tomcat/webapps/

cd /usr/local/tomcat/webapps/

3.2创建一个目录(test或root目录(tomcat的默认目录))

mkdir test

3.3将index.html 复制到容器中

rz 选择index.html 页面;

4.4安装rz上传命令

yum install lrzsz

5.5直接访问

http://服务器地址:端口+目录/index.html

4.关于Docker中安装Tomcat后不能访问及访问主页404的解决办法

https://blog.csdn.net/weixin\_38423249/article/details/80554583

https://www.cnblogs.com/chzushl/p/13385580.html

## 18.Docker部署war项目

1.Linux安装docker

1. 运行tomcat

docker run -d --name tomcat-8080 -p 8080:8080 -v /usr/local/webapps/:/usr/local/tomcat/webapps/ tomcat

2.切入到/usr/local/webapps/

3.使用rz命令选择war包;

4.浏览器访问

## 19.Eclipse或Idea 集成Docker实现镜像打包一键部署

### 1.Linux环境准备Cetos7(腾讯云)

a.安装Docker

b.安装Mysql

c.安装Redis

### 2.windows 环境准备

a.安装开发工具

b.安装docker插件

c.Linux远程连接工具(xshell)

d.项目准备(Springboot+Maven+mysql+Redis+eureka)

e.打包项目推送镜像到服务器

f.运行镜像

d.浏览器访问

### 3.Docker插件的安装

Idea安装Docker插件

a.打开idea,从File-->Settings-->Plugins-->Install jetBrains plugin 进入插件安装界面;

b.在搜索框中输入docker，可以看到Docker integration,点击右边的install按钮进行安装，安装后重启idea;

**Idea配置docker**

a.配置docker，连接到远程的docker容器

从File-->Setttings-->Builde,Excution,Deployment->Docker打开配置界面;

eclipse 安装Docker插件

a.Help-->Eclipse Marketplace-->搜索 Eclipse Docker Tooling

**eclipse配置docker**

1. window-->show view-->other-->选择docker explorer

相关文档：<https://www.e-learn.cn/topic/1883011>

### 4.Docker插件连接到Docker

1.Docker引擎默认不允许外部访问需要开启远程访问;  
#修改Docker服务文件  
vim /lib/systemd/system/docker.service  
#修改ExecStart这行

ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:2375 -H unix:///var/run/docker.sock  
将文件内的ExecStart注释，新增上行红色字体部分;

#重新加载配置文件  
systemctl daemon-reload  
#重启服务  
systemctl restart docker.service  
#查看端口是否开启  
netstat -nlpt #如果找不到netstat命令，可以进行安装 yum install net-tools  
#检查curl是否生效，注意，必须关闭防火墙或腾讯云安全组开放2375端口，这个端口也可以自定义  
curl http://服务器地址：2375/info  
注意ip地址必须和服务器地址统一;  
相关文档：<https://blog.csdn.net/boling_cavalry/article/details/100049996>

### 5.配置Docker插件

两种方式:

1.直接在pom.xml中配置;

2.在pom中指定Dockfile文件的位置;

直接在pom.xml中配置的方式:

<!-- docker 插件 可以帮助我们在Maven工程中通过简单的配置，自动生成镜像并推送到仓库中-->

<plugin>

<groupId>com.spotify</groupId>

<artifactId>docker-maven-plugin</artifactId>

<version>1.0.0</version>

<configuration>

<!-- 镜像名称 -->

<imageName>${docker.image.prefix}/${project.artifactId}</imageName>

<!-- 指定标签 -->

<imageTags>

<imageTag>latest</imageTag>

</imageTags>

<!-- 基础镜像java 1.8 -->

<baseImage>java</baseImage>

<!-- 制作者提供本人信息 -->

<maintainer>wangtao wwangtaoc11@163.com</maintainer>

<!-- 切换到/root目录 -->

<workdir>/ROOT</workdir>

<cmd>["java","-version"]</cmd>

<entryPoint>["java","-jar","${project.build.finalName}.jar"]</entryPoint>

<!-- 指定远程docker api 地址 -->

<dockerHost>http://106.54.162.159:2375</dockerHost>

<!-- 指定Dockerfile文件的位置

<dockerDirectory>src/main/docker</dockerDirectory> -->

<resources>

<resource>

<targetPath>/ROOT</targetPath>

<!-- 指定需要复制的根目录:${project.build.directory}表示target目录 -->

<directory>${project.build.directory}</directory>

<!-- 指定需要复制的文件 ${project.build.finalName}.jar指的是打包后的文件-->

<include>${project.build.finalName}.jar</include>

</resource>

</resources>

</configuration>

<!-- 以上还需要通过mvn clean package docker:build 手动构建镜像，在<plugin>添加以下插件即可以

在项目打包Package的时候即可自动构建镜像,只需要run就可以访问了;

当执行mvn package时执行mvn clean package docker:build

-->

<executions>

<execution>

<id>build-image</id>

<phase>package</phase>

<goals>

<goal>build</goal>

</goals>

</execution>

</executions>

</plugin>

### 6.执行 mvn package