# docker 安装 以及简单项目部署

## 1.0 准备阶段

### 1.0.1 购买服务器

首先我们确保我们的运行是在Liunx环境运行 ,所以我们需要一台liunx服务器，没有小伙伴可以购买阿里云学生机 。当然我们也为了没钱的小伙伴准备了人性化的活动，阿里云高校计划。（当然规定是全日制在校学生才可以）

```
阿里云高校计划传送门：https://developer.aliyun.com/adc/student/
阿里云学生机购买传送门:https://promotion.aliyun.com/ntms/act/campus2018.html
```

### 1.0.2 准备下载连接工具

当买好阿里云服务器后 我们需要去xftp官网申请一下xftp 和xshell （学生免费）

这款工具简而言之就是方便我们连接远程我们的服务器。

冲!我们的准备工作就完成的。

## 2.0 docker 安装

### 2.1 docker官网

```
地址：https://docs.docker.com/engine/install/centos/
```

#### 2.1.1 运行命令

```
//清除服务器上docker
sudo yum remove docker \
                  docker-client \
                  docker-client-latest \
                  docker-common \
                  docker-latest \
                  docker-latest-logrotate \
                  docker-logrotate \
                  docker-engine
-----------------------------------------
//安装工具
sudo yum install -y yum-utils
-----------------------------------------
//配置镜像（官网镜像太慢了）所以我们找到一个阿里的镜像地址 听懂了没有？
sudo yum-config-manager --add-repo http://mirrors.aliyun.com/docker-
ce/linux/centos/docker-ce.repo
-----------------------------------------
sudo yum install docker-ce docker-ce-cli containerd.io
-----------------------------------------
```

```
 //启动docker
sudo systemctl start docker
-----------------------------------------
 //输入命令   查看当前容器
 docker ps
  //出现CONTAINER ID        IMAGE                COMMAND                  CREATED
      STATUS              PORTS                    NAMES
    代表安装docker成功！  下班 byebye。
```

2.1.2 配置阿里镜像加速（这一步可以不用配 配了就会更快更高效 便捷）

```
//阿里镜像加速教程
https://developer.aliyun.com/article/692983
```

2.1.3

```
我们需要去阿里云去开放我们的安全组规则
例如你需要访问你服务器8089端口  这时候你就需要配置服务器安全组规则其中一个8089
当然一个个端口开放比较麻烦  偷懒的小技巧是  开放所有端口（可能有安全问题）
```



docker 安装完毕!

# 3.0 简单项目部署

## 3.0.1 准备工作

```
1.准备对应版本JDK liunx 压缩包  自己去官网就可以下载:
https://www.oracle.com/cn/java/technologies/javase-downloads.html
2.编写一个dockerfile
// dockerfile内容如下
-----------------------------------------------------------------------------------
# Docker image for springboot file run
# VERSION 0.0.1
# Author: wl
```

```
# 建立一个新的镜像文件，配置模板：新建立的镜像是以centos为基础模板
# 因为jdk必须运行在操作系统之上
FROM centos:7

# 作者名  作者邮箱
MAINTAINER wl<1987153442@qq.com>

#设置系统编码
RUN yum install kde-l10n-Chinese -y
RUN yum install glibc-common -y
RUN localedef -c -f UTF-8 -i zh_CN zh_CN.utf8
#RUN export LANG=zh_CN.UTF-8
#RUN echo "export LANG=zh_CN.UTF-8" >> /etc/locale.conf
#ENV LANG zh_CN.UTF-8
ENV LC_ALL zh_CN.UTF-8


# 创建一个新目录来存储jdk文件
RUN mkdir /usr/local/java

#将jdk压缩文件复制到镜像中，它将自动解压缩tar文件（版本不一致自己调换版本）
ADD jdk-11.0.6_linux-x64_bin.tar.gz /usr/local/java/

# 创建软连接 ln -s 源文件 目标文件（主要自己上传jdk版本名字）
# RUN ln -s /usr/local/java/jdk-11.0.6 /usr/local/java/jdk

# 设置环境变量
ENV JAVA_HOME /usr/local/java/jdk-11.0.6
ENV PATH $JAVA_HOME/bin:$PATH

# VOLUME 指定了临时文件目录为/tmp
# 其效果是在主机 /var/lib/docker 目录下创建了一个临时文件，并链接到容器的/tmp
VOLUME /tmp

# 运行jar包
# 运行.sh文件 这里注意 .sh 文件放置于/usr/local/idea 位置以及文件名字 可以自定义
CMD ["sh","-c","/usr/local/idea/wljar.sh"]
-------------------------------------------------------------------------
 3   //编写xxx.sh 文件
    //1.jar 是jar包名字 前面是jar包存在路径
    //.sh文件内容如下
-------------------------------------------------------------------------
    java -jar /usr/local/idea/1.jar
-------------------------------------------------------------------------
    4准备一个jar包
-----------------------------------------------------------------
    5将jar包 jdk 以及 .sh 文件 以及dockerfile 丢到服务器中（依靠xftp）
```

### 3.0.2 docker 运行项目

```
//在dockerfile 所在路径下运行命令：wldemojar 是指代生成镜像的名字 此处可以替换
docker build -t wldemojar .
-----------------------------------------------------------------------------------------
//查看刚才创建镜像
docker images
//看见以下信息说明创建成功
```

```
//wldemojar                                                    latest
279c07755469        4 seconds ago       695MB
-------------------------------------------------------------------------------------
 // 上一步执行中可能出现 .sh permmison defined 权限不足问题
 // 解决办法：进入.sh所在目录执行 下面命令
  chmod u+x *.sh
-------------------------------------------------------------------------------------
 //运行docker run 命令 使容器生效 至此简单部署全部完成    -v是容器挂载  --name后参数是镜像名字
docker run --name thp -it -v /usr/local/idea:/usr/local/idea --privileged=true -d -p
8081:8081 jar
-------------------------------------------------------------------------------------
//容器id
 d69828275dfbc53ab6724d746e5fefaf017f125af79e298d1748eb8d7e34d921
//部署成功 可以访问接口
```

### 3.0.3 一些命令解释


上述命令用到了容器挂载：容器挂载指的就是 容器内部数据 挂载在liunx服务器外部 这种挂载好处就是我们可以在容器外部去操作容器内部数据。
// -p 命令 -p 8080:8080 第一个8080指代jar包中运行端口号 后一个8080 指我们映射出来的端口号

```
//docker ps -aq 查看所有容器
//docker ps 查看当前运行容器
//docker stop 容器id  停止指定id容器（批量停止所有容器）
docker stop $(docker ps -aq)
//docker rm 容器id 删除一个容器 （这儿是批量删除所有容器）
docker rm $(docker ps -aq)
//docker images 查看当前docker存在镜像
dokcer rmi 镜像名字 删除镜像
docker rmi 镜像id
```

## 5.0 docker安装GitLab：

```
-------------------------------------------------------------------------------------
--------
    //下拉最新gitLab
    docker pull gitlab/gitlab-ce:latest
-------------------------------------------------------------------------------------
--------
    # 启动
    docker run --detach \
    --publish 443:443 --publish 80:80  --publish 222:22 \
```

```
    --name gitlab \
    --memory 4g \
    --restart always \
    --volume /srv/gitlab/config:/etc/gitlab \
    --volume /srv/gitlab/logs:/var/log/gitlab \
    --volume /srv/gitlab/data:/var/opt/gitlab \
    gitlab/gitlab-ce:latest

--------------------------------------------------------------------------------------------
--------
修改ssh端口
vi /etc/ssh/sshd_config
# 2.重启sshd服务
systemctl restart sshd

 --publish 暴露了容器的三个端口，分别是https对应的443，http对应80以及ssh对应的22(如果不需要配置
https，可以不暴露)
--memory 限制容器最大内存暂用4G，这是官方推荐的
--volume 指定挂载目录，这个便于我们在本地备份和修改容器的相关数据

--------------------------------------------------------------------------------------------
-----
--hostname gitlab.example.com: 设置主机名或域名
--detach 如果在docker run后面追加-d=true或者-d，那么容器将会运行在后台模式。此时所有I/O数据只能通
过网络资源或者共享卷组来进行交互。
--publish 8443:443：将http：443映射到外部端口8443
--publish 8880:80：将web：80映射到外部端口8880
--publish 8222:22：将ssh：22映射到外部端口8222
--name gitlab: 运行容器名
--restart always: 自动重启
--volume /srv/gitlab/config:/etc/gitlab: 挂载目录
--volume /srv/gitlab/logs:/var/log/gitlab: 挂载目录
--volume /srv/gitlab/data:/var/opt/gitlab: 挂载目录
--privileged=true 使得容器内的root拥有真正的root权限。否则，container内的root只是外部的一个普通
用户权限
--------------------------------------------------------------------------------------------
-------
# 打开挂载的配置目录
vi /srv/gitlab/config/gitlab.rb

####################################################
# 添加外部请求的域名(如果不支持https，可以改成http)
external_url 'https://gitlab.yinnote.com'
# 修改gitlab对应的时区
gitlab_rails['time_zone'] = 'PRC'
# 开启邮件支持
gitlab_rails['gitlab_email_enabled'] = true
gitlab_rails['gitlab_email_from'] = 'gitlab@yinnote.com'
gitlab_rails['gitlab_email_display_name'] = 'Yinnote GitLab'
# 配置邮件参数
gitlab_rails['smtp_enable'] = true
gitlab_rails['smtp_address'] = "smtp.mxhichina.com"
gitlab_rails['smtp_port'] = 25
gitlab_rails['smtp_user_name'] = "gitlab@yinnote.com"
gitlab_rails['smtp_password'] = "xxxxxx"
gitlab_rails['smtp_domain'] = "yinnote.com"
gitlab_rails['smtp_authentication'] = "login"
gitlab_rails['smtp_enable_starttls_auto'] = true
```

```
gitlab_rails['smtp_tls'] = false
###################################################

-------------------------------------------------------------------------------------------------
--------

# gitlab.rb文件内容默认全是注释
$ vim /home/gitlab/config/gitlab.rb
# 配置http协议所使用的访问地址,不加端口号默认为80
external_url 'http://xxxx'

# 配置ssh协议所使用的访问地址和端口
gitlab_rails['gitlab_ssh_host'] = 'xxxxxx'
gitlab_rails['gitlab_shell_ssh_port'] = 222 # 此端口是run时22端口映射的222端口
:wq #保存配置文件并退出




# 参考
https://segmentfault.com/a/1190000019772866?utm_source=sf-related
```

## 4.0结语

这是docker容器技术初探。简简单单容器部署jar包。

不足之处在于 每次部署需要终止容器 删除镜像 替换jar包 或者.sh文件有问题就需要吧镜像容器全部删除

重新运行docker build 以及docker run  命令 比较麻烦 很讨厌 下次分享更简单部署