<https://www.cnblogs.com/caoweixiong/p/12186736.html>



docker基础操作须知:

镜像：应用程序所依赖的操作环境和配置形成的模板，用面向对象的话来说简称类

容器:通过镜像来运行容器，容器相当于类的实例。

常用命令大全:

ip addr/ip link/ ifconfig：查看宿主机里ip地址（宿主机表示是虚拟机)

systemctl start docker/service start docker：启动docker服务

systemctl stop docker:关闭docker服务

service iptables start:启动防火墙

service iptables restart:重启防火墙

service iptables stop:关闭防火墙

docker run：启动container

docker seach xxxx(镜像名称):检索镜像

docker pull xxx(镜像名称）：拉取镜像

docker images:列出镜像

docker  --version:查看docker的版本信息

[root@localhost  ~]:家目录

[root@localhost  /]:根目录

docker exec Container Id:进入容器

docker exec -it NAMES(自定义名称) /bin/bash:进入--name命名的容器

docker exec NAMES(容器的自定义名称）-it bash:进入容器里进行操作

docker attach Container Id:进入到容器的终端

docker inspect IMAGE ID:显示容器或镜像的元数据

[root@（container id）]：比如[root@24527d42d3eb]表示已进入docker里的一个CONTAINER ID的容器内部

[root@24527d42d3eb]:exit(从容器里退出）。

docker ps：查看docker里运行的容器（status为Up....表示运行中）

docker ps -a:查看docker里已停止或正在运行的容器（status为Exit)

docker start CONTAINER ID   ：如果容器的状态为Exit,用它来启动容器

docker stop CONTAINER ID   :停止正在运行的容器，删除一个容器必须先要停止

docker restart :当部署web项目的war文件，到tomcat里的webapps里面了，重启一下docker服务，就自动解压了war文件

docker rm CONTAINER ID ：删除容器

docker rm $(docker ps -a -q)：删除所有已停止的容器;

docekr rmi IMAGE ID:删除镜像

rm -rf xxxx：强制删除某个war.文件或目录、文件

docker cp /xxx.war CONTAINER ID   :/usr/local/tomcat/webapps：从/根目录复制一个打包后的Java web war文件 到tomcat容器里的webapps工作目录里

docker exec CONTAINER ID   ls /usr/local/tomcat:默认的 安装tomcat镜像的webapps目录

docker run -name xxx -p  (自定义端口号,用来访问的):(容器的端口号：比如tomcat的8080，mysql的3306） -d docker.io/tomcat(镜像名称)：表示在docker里运行一个tomcat容器,如果没有镜像，自动pull一个tomcat的镜像

--name 为容器取一个名字

-p 容器要映射的端口号

-d:后台运行

-t：进入终端

-i：获得一个交互式的连接，通过获取container的输入

/bin/bash：在container中启动一个bash shell

举例：docker run --name tomcat -p 8081:8080 -d tomcat

=================================================

springboot 项目 集成docker部署

一.部署方式1.传统 打包jar 本地编写docker File（桌面新建dockfile） dock file是什么使

用阅读 详见up博客

构建镜像dock build -t test.(随便取)

docker images查看

运行镜像

docker run -d -p 8080:8080 -name test hello（-d后台运行-p端口映射 -name是取个名字 test 镜像名称）

docker ps查看

========================================

**3. 编写Dockerfile文件**

# Docker image for springboot file run
# VERSION 0.0.1
# Author: eangulee
# 基础镜像使用java
FROM java:8
# 作者
MAINTAINER eangulee <eangulee@gmail.com>
# 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
# 将jar包添加到容器中并更名为app.jar
ADD demo-0.0.1-SNAPSHOT.jar app.jar
# 运行jar包
RUN bash -c 'touch /app.jar'
ENTRYPOINT ["java","-Djava.security.egd=file:/dev/./urandom","-jar","/app.jar"]

**解释下这个配置文件：**

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。该步骤是可选的，如果涉及到文件系统的应用就很有必要了。/tmp目录用来持久化到 Docker 数据文件夹，因为 Spring Boot 使用的内嵌 Tomcat 容器默认使用/tmp作为工作目录

项目的 jar 文件作为 “app.jar” 添加到容器的

ENTRYPOINT 执行项目 app.jar。为了缩短 Tomcat 启动时间，添加一个系统属性指向 “/dev/./urandom” 作为 Entropy Source

如果是第一次打包，它会自动下载java 8的镜像作为基础镜像，以后再制作镜像的时候就不会再下载了。

**4. 部署文件**

在服务器新建一个docker文件夹，将maven打包好的jar包和Dockerfile文件复制到服务器的docker文件夹下

docker文件夹

**5. 制作镜像**

执行下面命令， 看好，最后面有个"."点！

docker build -t springbootdemo4docker .

-t 参数是指定此镜像的tag名

制作完成后通过**docker images**命令查看我们制作的镜像

**6.启动容器**

[root@localhost docker]# docker run -d -p 8080:8085 springbootdemo4docker
-d参数是让容器后台运行
-p 是做端口映射，此时将服务器中的8080端口映射到容器中的8085(项目中端口配置的是8085)端口

**7. 访问网站**

直接浏览器访问： http://你的服务器ip地址:8080/

=====================================================

docker logs -f --tail 200 adtp

--tail 200只显示最新的两百行

adtp 服务名

===========================================

docker ps -a 查看运行中容器

docker ps -a |grep xxx 加上过滤名称 查找服务

==========================================

docker inspect xxx容器名称 查看容器环境变量

docker inspect xxx容器名称 |grep REDIS 查看容器中环境变量 redis配置

=================================================

容器启动方式

docker run 命令

docker-componse启动

docker -componse xxxxx.yaml(docker 配置文件) up -d

查看docker配置文件（包含环境变量配置）134服务器查看

cd opt | cd adtp