Docker

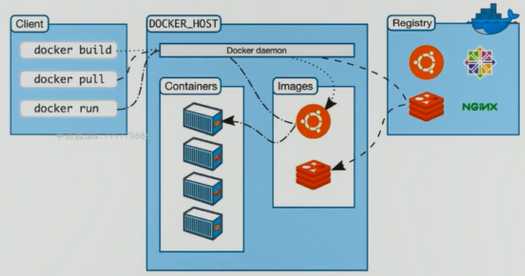
# 简介

* docker是一个开源的应用容器引擎，让开发者可以打包他们的应用以及依赖包到一个可移植的容器中，然后发布到任何流行的linux机器上，也可以实现虚拟化
* 容器是完全使用沙箱机制，相互之间不会有任何接口（类似IPhone）。几乎没有性能开销，可以很容易的在机器和数据中心里运行。
* 不依赖任何语言、框架、系统
* 开发语言：GO



# 概念

* 镜像：其实就是模板，跟我们常见的ISO镜像类似，是一个样板
* 容器：使用镜像常见的应用或系统
* 仓库：存放镜像的地方，分为公开仓库和私有仓库



# 特点

1. 操作启动快

运行时的性能可以获取极大提升，管理操作（启动、停止、开始、重启等等）。都是以秒或毫秒为单位的

1. 轻量级虚拟化

你会拥有足够的“操作系统”，仅需添加或减小镜像即可。在一台服务器上可以部署100-1000个Containers容器。但是传统的虚拟化，你虚拟10-20个虚拟机就不错了

1. 开源免费

开源免费低成本。由Linux内核支持并驱动

# 目的

解决环境的不一致。最简单的理解，我给你镜像，你直接拿过来部署，就拥有了我的一切，你再也不用为如何搭建正确能“跑”的环境忧伤了。

一次封装，到处运行

# 基本概念

* 镜像(image)：就好比是一个模板，可以通过这个模板来创建容器服务
* 容器(container)：利用容器技术，独立运行一个或者一组应用，通过镜像来创建
* 仓库(repository)：存放镜像的地方，分为公有仓库和私有仓库

# 安装

1. 确认内核版本：uname -r。要求：返回的值大于3.10即可
2. 执行脚本：curl -sSL https://get.daocloud.io/docker | sh
3. 安装：yum install docker-ce -y
4. 启动：systemctl start docker
5. 测试：docker -v

|  |
| --- |
| 1. 卸载旧的版本  yum remove docker \  docker-client \  docker-client-latest \  docker-common \  docker-latest \  docker-latest-logrotate \  docker-logrotate \  docker-engine    2. 下载安装包  yum install -y yum-utils  3. 设置镜像的仓库  yum-config-manager \  --add-repo \  https://download.docker.com/linux/centos/docker-ce.repo  更新yum软件包索引  yum makecache fast  4. 安装docker  yum install docker-ce docker-ce-cli containerd.io  5. 启动docker  systemctl start docker  6. 测试是否安装成功  docker version  7. hello world  docker run hello-world  8. 查看下载的镜像  docker images |

第7步会出错，因为要修改镜像加速器

docker在本地没有找到hello-world镜像，也没有从docker仓库中拉取镜像，出项这个问题的原因：是应为docker服务器在国外，我们在国内

无法正常拉取镜像，所以就需要我们为docker设置国内阿里云的镜像加速器；

|  |
| --- |
| 1、新建vi /etc/docker/daemon.json  2、赋权：chmod 777 daemon.json  3、写入  {  "registry-mirrors": ["https://alzgoonw.mirror.aliyuncs.com"]  }  4、重启：systemctl restart docker  5、查看状态，保证处于running：systemctl status docker  6、重新执行命令：docker run hello-world |

# 卸载

|  |
| --- |
| 1. 卸载依赖  yum remove docker-ce docker-ce-cli containerd.io  2. 删除资源  rm -rf /var/lib/docker #docker的默认工作路径 |

# 常用命令

## 帮助命令

|  |
| --- |
| docker version  docker info  docker --help |

## 镜像命令

docker images

参数：

-a：列出本地所有镜像

-q：只显示镜像id

--digests：显示镜像摘要

--no-trunc：显示完整的镜像信息

|  |
| --- |
|  |

docker search 镜像名

主要关注stars（受欢迎程度）

|  |
| --- |
|  |

从 Docker Hub 查找所有镜像名包含 tomcat，并且受欢迎数大于 1000 的镜像

docker search -f stars=1000 tomcat

|  |
| --- |
|  |

docker pull 镜像名

远程拉镜像，类似github的pull

|  |
| --- |
|  |

docker rmi 镜像名id

|  |
| --- |
| docker rmi -f hello-world #删除一个  docker rmi -f hello-world tomcat #删除多个  docker rmi -f ${docker images -qa} #删除所有镜像 |

## 容器命令

* 有镜像才能有容器
* docker是利用容器独立运行一个或一组应用。容器是用镜像创建的运行实例
* 它可以被启动、开始、停止、删除
* 每个容器都是相互隔离的、保证安全的平台。（集装箱）
* 可以把容器看成一个简易版的linux环境（包括root用户权限、进程空间、用户空间和网络空间等）和运行在其中的应用程序

### 新建容器

语法：docker run [option] IMAGE [command] [args]

option:

--name="容器新名字": 为容器指定一个名称

-d: 后台运行容器，并返回容器ID，也即启动守护式容器

-i：以交互模式运行容器，通常与 -t 同时使用

-t：为容器重新分配一个伪输入终端，通常与 -i 同时使用

-P: 随机端口映射

-p: 指定端口映射，有以下四种格式

ip:hostPort:containerPort

ip::containerPort

hostPort:containerPort

containerPort

|  |
| --- |
| docker run -it 300e315adb2f    说明：300e315adb2f是我的镜像id，可通过docker images查看得到 |

如下方案用得多一些，在新建容器时取个容器名

|  |
| --- |
| docker run -it --name centos\_study centos |

### 列出当前正在运行的容器

docker ps [options]

-a :列出当前所有正在运行的容器+历史上运行过的 （-a all）

-l :显示最近创建的容器。 -l（last）

-n :显示最近n个创建的容器。 docker ps -n 5 #最近创建的5个容器

-q :静默模式，只显示容器编号。docker -ps

--no-trunc :不截断输出

### 进入正在进行的容器并以命令行交互

docker exec -it 容器ID /bin/bash

docker attach 容器ID

#两种方式的区别

attach 直接进入容器启动命令的终端，不会启动新的进程

exec 在容器中打开新的终端，并且可以启动新的进程

docker exec -it 容器id ls -l /

### 从容器内拷贝文件到主机上

docker cp 容器ID:容器内的路径 主机目录

### 当镜像运行之后可以修改容器里面的内容，再提交成一个新的镜像（备份）

docker run -it -p 主机端口:docker容器端口 tomcat

docker run -it -P tomcat #随机分配端口

docker commit -a='作者' -m='镜像描述' 容器ID 新的镜像名/名称:版本

### 退出容器

exit: 容器停止退出

ctrl+p+q: 容器不停止退出

### 启动容器

docker start 容器ID或容器名称

### 重启容器

docker restart 容器ID或容器名称

### 停止容器

docker stop 容器ID或容器名称

### 强制停止容器

docker kill 容器ID或容器名称

### 删除已停止的容器

docker rm 容器ID

docker rm -f $(docker ps -aq) #删除所有容器

docker ps -a -q | xargs docker rm #删除所有的容器

### 其他相关

启动守护式容器

docker run -d 容器名

查看容器日志

docker logs -tf --tail 10 容器id

-t 是加入时间戳

-f 跟随最新的日志打印

--tail 数字 显示最后多少条

查询容器运行进程

docker top 容器id

查询容器内部细节

docker inspect 容器id

# 容器卷

目的：数据持久化（保存数据，实现容器之间的数据共享等）

特点：

1. 数据卷可在容器之间共享或重用数据
2. 卷中的更改可以直接生效
3. 数据卷中的更改不会包含在镜像的更新中
4. 数据卷的生命周期一直持续到没有容器使用它为止

## 命令添加

|  |
| --- |
| docker run -it -v /myData:/dataContainer 镜像名(centos)  docker run -it -v /myData:/dataContainer:ro 镜像名(centos)  ro:只读（read-only） |

## Dockerfile添加

### 基础指令

* FROM:基础镜像,当前新镜像是基于哪个镜像的
* MAINTAINER:镜像维护者的姓名和邮箱地址
* RUN:容器构建时需要运行的命令
* EXPOSE:当前容器对外暴露的端口号
* WORKDIR:指定在创建容器后,终端默认登录进来的工作目录
* ENV:设置环境变量时使用
* ADD:相当于cpoy+解压缩以及处理url
* COPY:拷贝
* VOLUME:数据卷指定
* CMD:指定一个容器启动时要运行的命令,可以有多个CMD,但只有最后一个生效,CMD会被docker run之后的参数替换
* ENTRYPOINT:和CMD类似,但有追加的效果
* ONBUILD:在构建一个被继承的Dockerfile时运行命令,父镜像在被子继承后父镜像的onbuild被触发,类似于触发器.父镜像结束后做点收尾,有点类似触发器中的after

### 案例

构建一个自己的centos并自动包含了vim和net-tools且默认工作空间为/tmp

|  |
| --- |
| FROM centos  MAINTAINER caichang<caichangfast@qq.com>  ENV mypath /tmp  RUN yum -y install vim  RUN yum -y install net-tools  WORKDIR $mypath  EXPOSE 80  CMD echo "success-------ok"  CMD ["/bin/bash"] |

通过docker history 镜像id可看到底层的执行过程



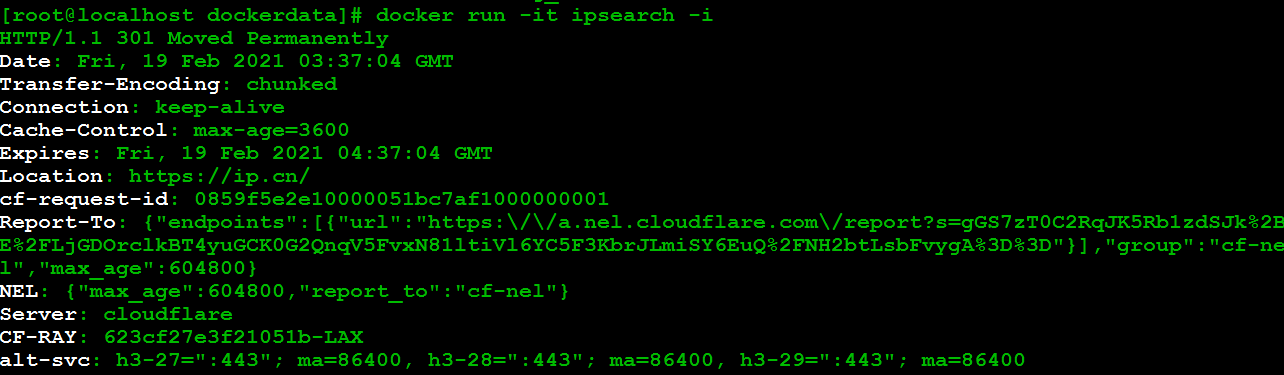
案例2:

ENTRYPOINT的用法赏析,实现ip查询

|  |
| --- |
| FROM centos  MAINTAINER caichang<caichangfast@qq.com>  RUN yum -y install curl  ENTRYPOINT ["curl","-s","http://ip.cn"] |

执行时,可带入-i参数来查询信息头

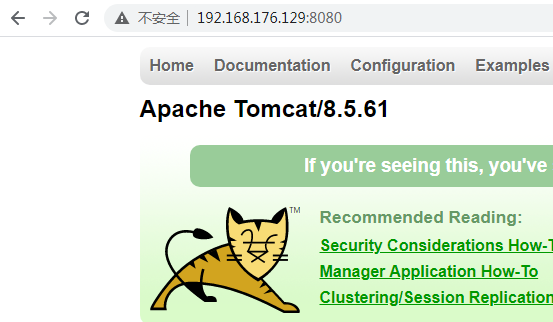
|  |
| --- |
| docker run -it ipsearch -i |



# 部署

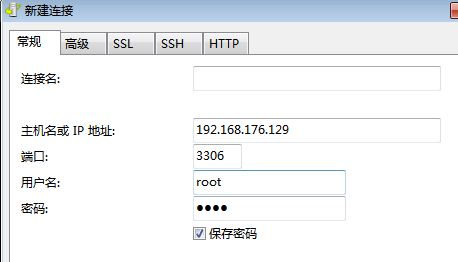
## 部署tomcat

1. 拉取镜像：docker pull tomcat:8.5
2. 创建tomcat容器：docker run -d --name tomcat\_test -p 8080:8080 tomcat:8.5
3. 进入容器：docker exec -it fd26142b93a5 /bin/bash
4. 删除webapps目录；rm -rf webapps
5. 重命名目录：mv webapps.dist/ webapps
6. 启动浏览器，输入：http://linux的ip:8080，即可看到如下图示
7. 可备份一下，使其成为基础的tomcat，以后直接引用就行：docker commit -a='caichang' -m='tomcat基础镜像' c006dd416916 basic\_tomcat:1.0



## 部署mysql

1. docker pull mysql：5.6
2. docker run --name=basic\_mysql -itd -p 3306:3306 -e MYSQL\_ROOT\_PASSWORD=root mysql:5.5
3. 直接可以用navicat来连接就可以



如出现权限问题,开放远程权限即可

1. docker exec -it 3ee6e16f04dd /bin/bash
2. mysql -uroot -p
3. 输入密码：root
4. use mysql;
5. update user set authentication\_string = password('root') where user = 'root';
6. GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY 'root' WITH GRANT OPTION;
7. flush privileges;

# 最后的思考

通过Dockfile完成一键部署项目