链滴

【入门篇】Docker 容器化技术

作者：wanming001
原文链接：https://ld246.com/article/1574427500496
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

Docker简介：

概述：

docker是基于go语言实现的虚拟化容器技术，正如他的log描述的一样，容器是完全使用沙箱机制，有独立的运行环境，更重要的是容器性能开销极低

优点：

1. 快速，一致地交付您的应用程序（快速CI/DI）
2. 响应式部署和扩展（基于容器，快速扩展）
3. 在同一硬件上运行更多工作负载（基于虚拟环境）

三个基本概念：

- 镜像(image):docker镜像，就相当于电脑安装系统的那个镜像文件，类似于面向对象程序设计里面类的概念
- 容器(container):docker容器，源于镜像，类似于面向对象程序设计里面的对象，有创建，启动，止，删除等操作，一个镜像可以运行多个容器
- 仓库(repository)：仓库可以看做是镜像的版本控制中心，存放镜像的

架构：

Docker 使用客户端-服务器 (C/S) 架构模式，使用远程API来管理和创建Docker容器

Docker常用命令及相关参数说明：

docker run XXX 容器创建并运行命令(docker create XXX 同docker run 只创建不运行)
-d: 后台运行容器，并返回容器ID；

-P: 随机端口映射，容器内部端口随机映射到主机的高端口
-p: 指定端口映射，格式为：主机(宿主)端口:容器端口
--name="nginx-lb": 为容器指定一个名称；
-h "mars": 指定容器的hostname；
-e username="ritchie": 设置环境变量；
--net="bridge": 指定容器的网络连接类型，支持 bridge/host/none/container: 四种类型；
--volume , -v: 绑定一个卷
示例：docker 运行一个nginx
docker run --name nginx -p 8090:80 -v /data:/data -d nginx:latest

#启动/停止/重启 容器
docker start/stop/restart 容器名称

强制删除正在运行的容器
docker rm -f 容器名称/容器id

docker ps #查看所有正在运行的容器 -a 查看所有的容器

docker inspect 容器名称 #查看容器的信息 docker inspect | grep IPADDRESS 查看容器的ip

docker top 容器名称 #查看容器的进程信息

docker logs --since="2016-07-01" --tail=10 容器名称

docker port 容器名称 #查看容器的端口映射情况

docker commit -a "runoob.com" -m "my apache" 容器id mymysql:v1 #从容器创建一个新的镜。-a 作者 -m 描述 -p提交时停止容器

docker cp /www/runoob 96f7f14e99ab:/www/ #将主机/www/runoob目录拷贝到容器96f7f14e9ab的/www目录下。
docker diff mymysql #检查容器里文件结构的更改。

docker login -u 用户名 -p 密码 #登陆容器仓库

docker logout #登出

docker pull 镜像名称:版本 #不指定版本，默认最新版本latest

docker push myapache:v1 #上传本地镜像myapache:v1到镜像仓库中。

docker search -s 10 java # 搜索镜像名包含java，并且收藏数大于10的镜像

docker images #查看所有的镜像

docker rmi 镜像id #删除镜像

docker history runoob/ubuntu:v3 #查看本地镜像runoob/ubuntu:v3的创建历史。

docker save -o my_ubuntu_v3.tar runoob/ubuntu:v3 #将镜像 runoob/ubuntu:v3 生成 my_ubuntu_v3.tar 文档

docker tag ubuntu:15.10 runoob/ubuntu:v3 #将镜像ubuntu:15.10标记为 runoob/ubuntu:v3 像。

docker load -a xxx.tar #导入镜像包含记录数据

docker build -t 镜像名称. #打包

docker info #查看docker系统信息。

docker version #显示 Docker 版本信息。

docker network create –driver=bridge br0 #创建网络模式为bridge的名字为bro的网络

Docker网络模式：

查看容器的IP

docker inspect 容器ID(容器名称) | grep IPAddress

网络模式： 指定容器运行网络模式 --net=具体的网络模式

- 创建一对虚拟接口，分别放到本地主机和新容器中；
- 本地主机一端桥接到默认的 docker0 或指定网桥上，并具有一个唯一的名字，如 veth65f9；
- 容器一端放到新容器中，并修改名字作为 eth0，这个接口只在容器的名字空间可见；
- 从网桥可用地址段中获取一个空闲地址分配给容器的 eth0，并配置默认路由到桥接网卡 veth65f9。
- 完成这些之后，容器就可以使用 eth0 虚拟网卡来连接其他容器和其他网络。

1. --net=bridge 默认的网桥

2. --net=host 告诉 Docker 不要将容器网络放到隔离的名字空间中，即不要容器化容器内的网络。时容器使用本地主机的网络，它拥有完全的本地主机接口访问权限。

容器进程可以跟主机其 它 root 进程一样可以打开低范围的端口，可以访问本地网络服务比如 D-bus 还可以让容器做一些影响整个主机系统的事情，

比如重启主机。因此使用这个选项的时候要非常小心。如果进一步的使用 --privileged=true，容器被允许直接配置主机的网络堆栈

3. --net=container:NAME_or_ID 让 Docker 将新建容器的进程放到一个已存在容器的网络栈中，新器进程有自己的文件系统、进程列表和资源限制，

但会和已存在的容器共享 IP 地址和端口等网络资源，两者进程可以直接通过 lo 环回接口通信

4. --net=none 让 Docker 将新容器放到隔离的网络栈中，但是不进行网络配置。之后，用户可以自进行配置

查看所有的网络模式

```
docker network ls
NETWORK ID          NAME                DRIVER              SCOPE
b0de84ad03f1        bridge              bridge              local
08e7c8d472c8        harbor_harbor       bridge              local
0b99e2473f11        host                host                local
9cc2e2822015        none                null                local
bc6c919a46a6        vendor-platform-test   bridge           local
```

记住：牛逼的技术虽然不是你的，但是你会了，就是你的