容器 (http://39.106.3.150/tags/#1560852708518)

docker 简单运用

Posted by Palmer on 08-17，2019

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镜像中，然后发布到任何流行的 Linux或Windows 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。

Docker是利用Linux内核虚拟机化技术（LXC），提供轻量级的虚拟化，以便隔离进程和资源。LXC不是硬件的虚拟化，而是Linux内核的级别的虚拟机化，相对于传统的虚拟机，节省了很多硬件资源。

NameSpace

LXC是利用内核namespace技术，进行进程隔离。其中pid, net, ipc, mnt, uts 等namespace将container的进程, 网络, 消息, 文件系统和hostname 隔离开。

Control Group

LXC利用的宿主机共享的资源，虽然用namespace进行隔离，但是资源使用没有收到限制，这里就需要用到Control Group技术，对资源使用进行限制，设定优先级，资源控制等。

安装Docker

更新软件包

请确保服务器的软件包已经是最新的。

```
sudo yum update -y
```

安装必要依赖

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

添加软件源信息

```
sudo yum-config-manager --add-repo http://mirrors.aliyun.com/docker-ce/linux/centos/doc
```

更新 yum 缓存

```
sudo yum makecache fast
```

安装 Docker

```
sudo yum install docker-ce docker-ce-cli containerd.io
```

启动 Docker 后台服务

```
sudo systemctl start docker
```

允许当前用户直接运行 docker 命令

需要将当前用户加入 docker 用户组。这样每次运行 docker 命令的时候，就不需要加 sudo。

```
sudo usermod -aG docker your_name
#注意：设置成功之后需要重新登录才会生效。
```

镜像加速

```
# 新建 daemon.json 文件
sudo vim /etc/docker/daemon.json
```

将下面的配置复制进去即可：

```
{
  "registry-mirrors": ["http://hub-mirror.c.163.com"]
}
```

常用命令

```
#拉取镜像
docker pull image_name

#删除镜像
docker rmi image_name

#查找镜像
docker  search  image_name

#查看容器运行
docker  ps

#查看所有容器
docker  ps  -a

#启动，停止容器 ，重启
docker start container_name/container_id
docker stop container_name/container_id
docker restart container_name/container_id

#进入一个容器
docker attach container_name/container_id

#删除容器
docker rm container_name/container_id

#删除镜像
docker rmi image_name/image_id

#查看docker版本
docker version


#安装mysql镜像
docker pull mysql/mysql-server
#运行mysql
docker run --net=host --restart=always --privileged=true -v /usr/docker_dat/mysql/data:

##参数说明
--restart=always 跟随docker启动
--privileged=true 容器root用户享有主机root用户权限
-v 映射主机路径到容器
-e MYSQL_ROOT_PASSWORD=root 设置root用户密码
-d 后台启动
--lower_case_table_names=1 设置表名参数名等忽略大小写

#安装zookeeper
docker run --privileged=true -d --name zookeeper --publish 2181:2181 -d zookeeper:lates

#安装jenkins
docker run -p 8080:8080 -p 50000:50000 --privileged=true  -v /var/run/docker.sock:/var/

#进入jenkins容器
docker exec -it id /bin/bash
#查看密码
cat /var/jenkins_home/secrets/initialAdminPassword
```

```
#在$JENKINS_HOME/hudson.model.UpdateCenter.xml文件中，默认内容如下


<?xml version='1.0' encoding='UTF-8'?>
 <sites>
   <site>
    <id>default</id>
    <url>http://updates.jenkins-ci.org/update-center.json</url>
   </site>
</sites

#这个地址在外国的服务器，因为墙的原因，下载初始化界面所需插件不了，就一直处于等待状态

#把url改为http://mirror.xmission.com/jenkins/updates/update-center.json就解决了
#jenkins插件清华大学镜像地址
#https://mirrors.tuna.tsinghua.edu.cn/jenkins/updates/update-center.json


#配置国内镜像源
#实际在使用过程中，运行 apt-get update，然后执行 apt-get install -y vim，下载地址由于是海外地址，
    mv /etc/apt/sources.list /etc/apt/sources.list.bak
    echo "deb http://mirrors.163.com/debian/ jessie main non-free contrib" >/etc/apt/so
    echo "deb http://mirrors.163.com/debian/ jessie-proposed-updates main non-free cont
    echo "deb-src http://mirrors.163.com/debian/ jessie main non-free contrib" >>/etc/a
    echo "deb-src http://mirrors.163.com/debian/ jessie-proposed-updates main non-free

#更新安装源
apt-get update

#容器参数更新
docker container update --restart=always 容器名字

#创建镜像
docker commit 容器名/容器id 镜像名

#导出镜像
docker save spring-boot-docker  -o  /home/wzh/docker/spring-boot-docker.tar

#导入镜像
docker load -i spring-boot-docker.tar

#使用当前目录的 Dockerfile 创建镜像，标签为 runoob/ubuntu:v1。
docker build -t runoob/ubuntu:v1 .

#使用URL github.com/creack/docker-firefox 的 Dockerfile 创建镜像。
docker build github.com/creack/docker-firefox

#
也可以通过 -f Dockerfile 文件的位置：
$ docker build -f /path/to/a/Dockerfile .

#在 Docker 守护进程执行 Dockerfile 中的指令前，首先会对 Dockerfile 进行语法检查，有语法错误时会返
$ docker build -t test/myapp .

Sending build context to Docker daemon 2.048 kB
Error response from daemon: Unknown instruction: RUNCMD
```

```
#获取容器/镜像的元数据
docker inspect id


#清理容器的网络占用

docker network disconnect --force 网络模式 容器名称

#示例：docker network disconnect --force bridge zookeeper

#简查是否还有同名容器占用

docker network inspect 网络模式

#示例：docker network inspect bridge
```

NEXT POST →
(HTTP://39.106.3.150/ARCHIVES/20190726)

...

撰写评论...

上一页 下一页

FEATURED TAGS **(http://39.106.3.150/tags/)**

java 8 (http://39.106.3.150/tags/#java-8) java8 (http://39.106.3.150/tags/#java8)
redis (http://39.106.3.150/tags/#redis) 监控 (http://39.106.3.150/tags/#1561876610222)
全链路 (http://39.106.3.150/tags/#1561876610220) 容器 (http://39.106.3.150/tags/#1560852708518)
开源框架 (http://39.106.3.150/tags/#1560569459781) Spring (http://39.106.3.150/tags/#spring)
设计模式 (http://39.106.3.150/tags/#1559888728999) linux (http://39.106.3.150/tags/#linux)
SpringBoot (http://39.106.3.150/tags/#springboot) 大数据 (http://39.106.3.150/tags/#1559363598973)
区块链 (http://39.106.3.150/tags/#1559363594390) Java (http://39.106.3.150/tags/#java)

FRIENDS


(https://github.com/PowehiEdge)

你的世界不止在眼前