CLOUD-day05

云平台部署与管理

# 概述

## 什么是容器？

容器技术已经成为应用程序封装和交付的核心技术

容器技术的核心有以下几个内核技术组成

- Cgroups (Control Groups )-资源管理

- NameSpace-进程隔离

- SElinux安全

由于是在物理机上实施隔离，启动一个容器，可以项启动一个进程一样快速

## 什么是Docker?

Docker是完整的一套容器管理系统

Docker提供了一组命令，让用户更加方便直接地使用容器技术，而不需要过多关心底层内核技术

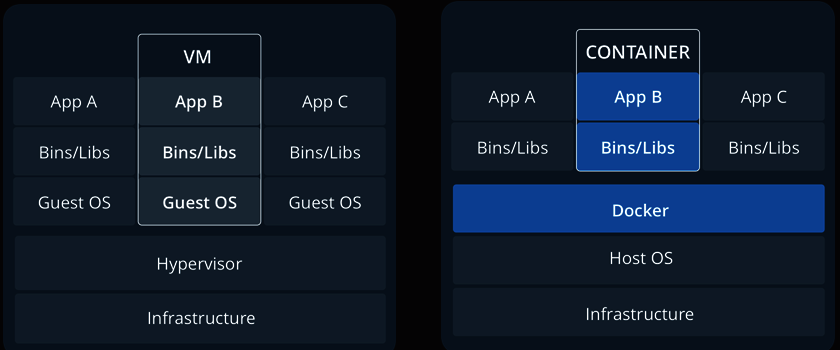
# Docker特性

## Docker优点

相比于传统的虚拟化技术，容器更加简洁高效

传统虚拟机需要给每个VM安装操作系统

容器使用的共享公共库和称序



## Docker缺点

容器的隔离性没有虚拟化强

功用Linux内核，安全性有先天缺陷

SELinux难以驾驭

监控容器和容器排错是挑战

# 部署Docker

部署Docker——>安装软件——>安装前准备&&安装Docker

## 安装软件

### 安装前准备

需要64为操作系统

至少RHEL6.5以上的版本，强烈推荐RHEL7

关闭防火墙（不是必须）

## 安装Docker

软件包列表

- docker-engine

- docker-engine-selinux

## 实验环境准备：

### 创建两台虚拟机：

[root@room9pc01 ~]# cd /var/lib/libvirt/images/

[root@room9pc01 images]#qemu-img create -b node.qcow2 -f qcow2 docker1.img 16G

[root@room9pc01 images]#qemu-img create -b node.qcow2 -f qcow2 docker2.img 16G

[root@room9pc01 images]#vim /etc/libvirt/qemu/docker1.xml

[root@room9pc01 images]#vim /etc/libvirt/qemu/docker2.xml

<memory unit='KiB'>4098782</memory>

<currentMemory unit='KiB'>4098782</currentMemory>

[root@room9pc01 images]#virsh define /etc/libvirt/qemu/docker1.xml

[root@room9pc01 images]#virsh define /etc/libvirt/qemu/docker2.xml

### **启动虚拟机并连接**

**配置静态IP**

**docker1: 192.168.1.12**

**docker2: 192.168.1.13**

[root@docker1 ~]# vim /etc/sysconfig/network-scripts/ifcfg-eth0

DEVICE="eth0"

ONBOOT="yes"

NM\_CONTROLLED="no"

TYPE="Ethernet"

BOOTPROTO="static"

**IPADDR="192.168.1.12"**

NETMASK="255.255.255.0"

GATEWAY="192.168.1.254"

### 配置yum源

[root@room9pc01 ~]# mkdir /var/ftp/docker

**自行准备软件包并拷贝到/var/ftp/docker**

[root@room9pc01 ~]# createrepo /var/ftp/docker

[root@room9pc01 ~]# ls /var/ftp/docker

docker-engine-1.12.1-1.el7.centos.x86\_64.rpm docker\_images.zip

docker-engine-selinux-1.12.1-1.el7.centos.noarch.rpm repodata

**两台docker机器上同样的yum源,并安装docker软件**

[root@docker1 ~]# vim /etc/yum.repos.d/CentOS-1807.repo

**.....**

[docker]

name=docker

baseurl=ftp://176.4.12.147/docker

gpgcheck=0

enabled=1

### 编辑本机DNS解析文件/etc/hosts

[root@docker1 ~]# vim /etc/hosts

... ...

192.168.1.12 docker1

192.168.1.13 docker2

### 安装软件

[root@docker1 ~]# yum -y install docker-engine.x86\_64

[root@docker1 ~]# systemctl restart docker

[root@docker1 ~]# systemctl enable docker

### 查看版本信息

[root@docker1 ~]# docker version



# Docker镜像

## 基本概念

### 什么是镜像？

在Docker中容器是基于镜像启动的

镜像是启动容器的核心

镜像采用分层设计

使用快照的COW技术，确保底层数据不丢失

### Docker hub镜像仓库

<https://hub.docker.com>

Docker官方提供公共镜像的仓库(Registry)

### 镜像操作

docker images () //查看

docker search 名称 //搜索

docker pull 名称 //下载

docker push 名称 //上传

### 导出镜像

[root@docker1 ~]# docker save busybox:latest >busybox.tar

[root@docker1 ~]# scp busybox.tar 192.168.1.13:/root/

### 导入镜像

[root@docker2 ~]# docker load < busybox.tar

### 启动镜像

启动centos镜像生成一个容器

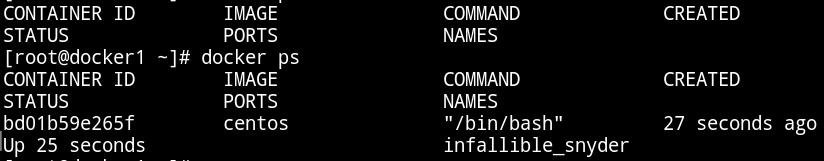
docker run -it 镜像名：标签名 要运行的命令

[root@docker1 ~]# docker run -it centos /bin/bash

### 查看容器信息

开启另一个终端（查看容器信息）

[root@docker1 ~]# docker ps



**新开终端对主机名、IP地址、进程、信号、用户、根文件系统项目进行测试，会发现进入容器前后信息完全变化**

debian | ubuntu

装包：apt-get

管理包：dpkg

# Docker基本命令

## 镜像常用命令

docker images //查看镜像列表

docker history //查看镜像制作历史

docker inspect //查看镜像底层信息

docker pull //下载镜像

docker push //上传镜像

docker rmi //删除本地镜像

(启动过容器的镜像无法直接删除，先删容器，再删除镜像)

docker save //镜像另存为tar包

docker load //使用tar包导入镜像

docker search //搜索镜像

docker tag //修改镜像名称和标签

## 详细用法及解释

### docker images

[root@docker1 ~]# docker images



镜像仓库名称 REPOSITORY

镜像标签 REPOSITORY

镜像ID IMAGE ID

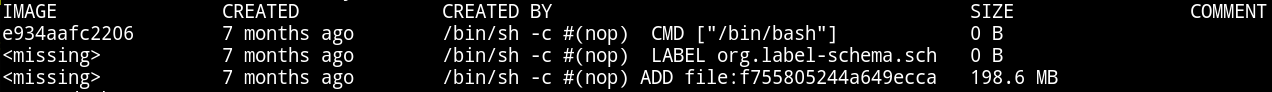
创建时间 CREATED

大小 SIZE

### docker history

- 了解镜像制作过程

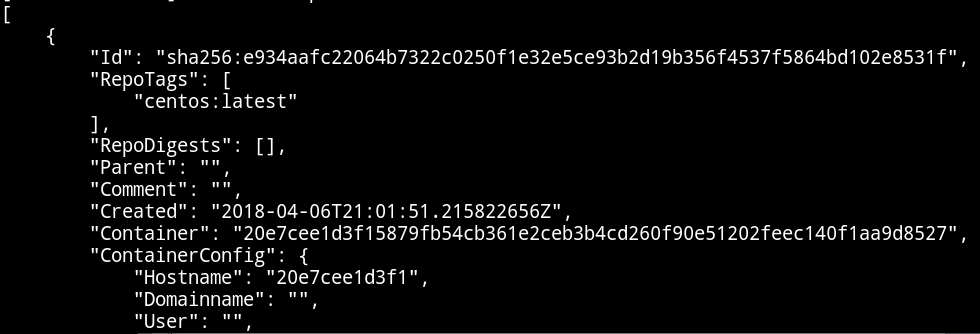
[root@docker1 ~]# docker history centos



### docker inspect

- 了解镜像环境变量、存储卷、标签等信息

[root@docker1 ~]# docker inspect centos



... ....

### docker rmi

- 启动容器时删除镜像会提示错误

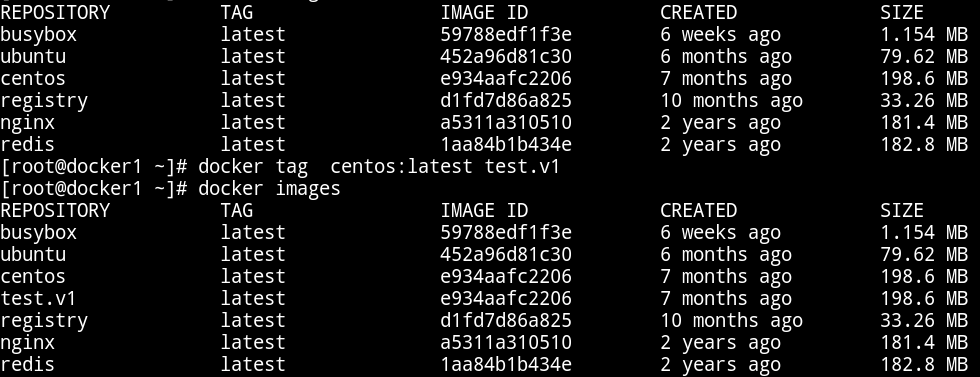
[root@docker1 ~]# docker rmi centos



### docker tag

- 重命名镜像名称（复制）

[root@docker1 ~]# docker tag centos:latest test.v1



## 容器常用命令

docker run //运行容器

docker ps //查看容器列表

docker stop //关闭容器

docker start //启动容器

docker restart //重起容器

docker attach|exec //进入容器

docker inspect //查看容器底层信息

docker top //查看容器进程列表

docker rm //删除容器

## 详细用法及解释

### Docker run -- 使用镜像启动容器

[root@docker1 ~]# docker run -it centos bash //重新开启一个容器

[root@de93517a1746 /]# exit

[root@docker1 ~]# docker run -itd centos bash

40004d9196d4476b7f2cfcc84e7d434779d224dd5acd4d6d854a6029ebd65c7a

[root@docker1 ~]# docker ps



### docker ps --列出容器列表

docker ps //查看正在运行的容器

docker ps -a //查看所有容器列表

docker ps -aq //仅显示容器id

### docker (stop | start | restart)

### 管理容器

docker stop //关闭容器

docker start //开启容器

docker restart //重起容器

示例：



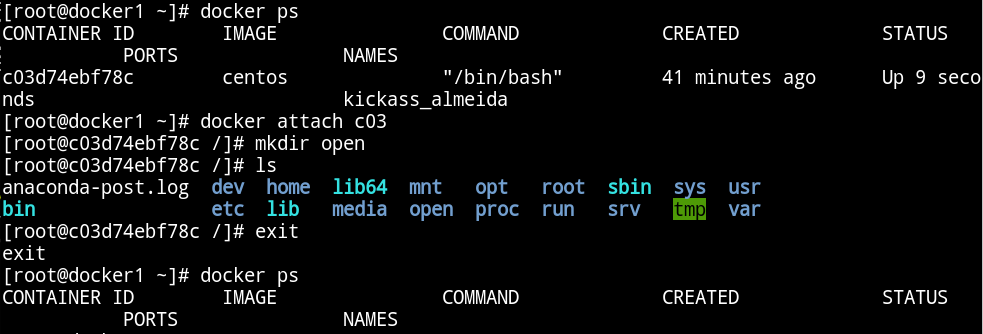
### docker attach | exec

### 进入容器

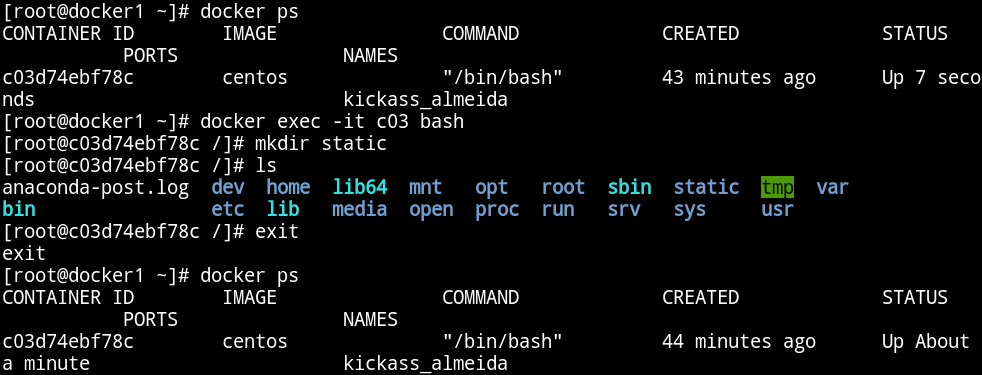
- docker attach //进入容器，exit会导致容器关闭

- docker exec //进入容器，退出时不会关闭容器

- attach示例：



- exec示例：



### docker inspect

查看容器底层信息

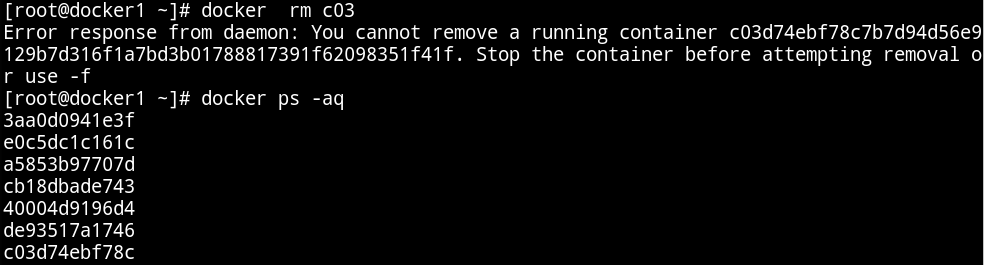


### docker top

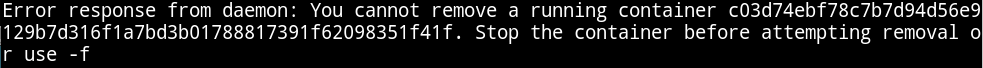


### docker rm --删除容器

- 删除正在运行的容器时会提示错误







[root@docker1 ~]# docker ps -aq

c03d74ebf78c