Docker虚拟化技术

# 简介

**Go** 语言开发的云开源项目 ：**“Build，Ship and Run Any App ,AnyWhere”**，通过对组件的封装，分发，部署，运行等生命周期的管理，使用户的及其环境能做到一次封装，到处运行。

Linux 容器技术的出现就解决了这样一个问题，而 Docker 就是在它的基础上发展过来的。将应用运行在 Docker 容器上面，而 Docker 容器在任何操作系统上都是一致的，这就实现了跨平台、跨服务器。只需要一次配置好环境，换到别的机子上就可以一键部署好，大大简化了操作

# 干些什么

\*而容器内的应用进程直接运行于宿主的内核，容器内没有自己的内核，而且也没有进行硬件虚拟。因此容器要比传统虚拟机更为轻便。

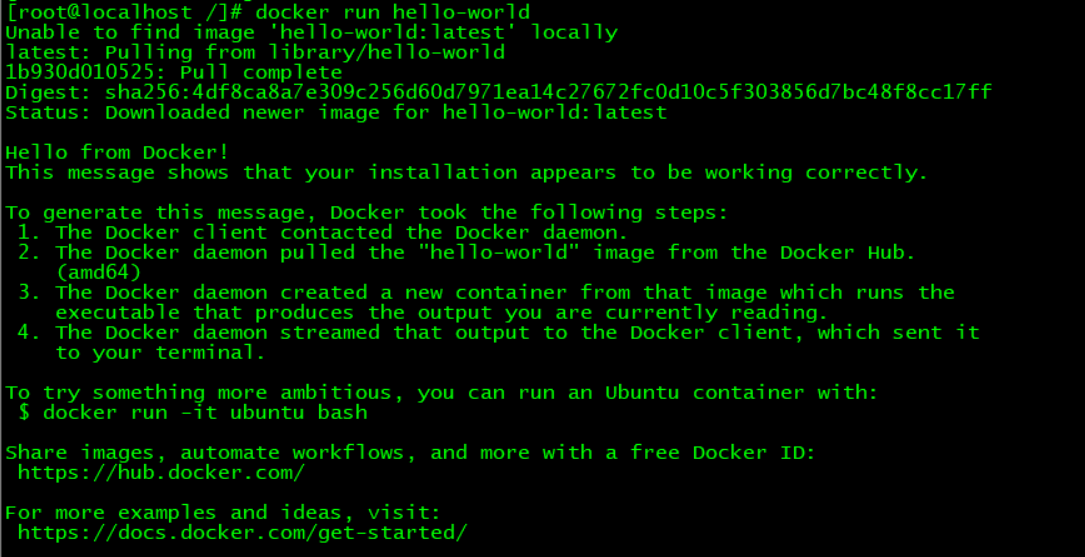
\* 每个容器之间互相隔离，每个容器有自己的文件系统 ，容器之间进程不会相互影响，能区分计算资源。

* 更快速的应用交付和部署
* 更便捷的升级和扩容
* 更简单的系统运维
* 更高效的计算资源利用

它是直接运营在宿主系统的内核上，可以秒级启动，所以docker就是一个缩小版的Linux系统

# Docker安装

验证：



我们在第一次安装运行hello-world实例的时候，会有一个从本地查找的过程，

Unable to find image ‘hello-world:latest’ locally

没有镜像就从我们的内部镜像中获取。

## 运行底层原理：

常用命令：

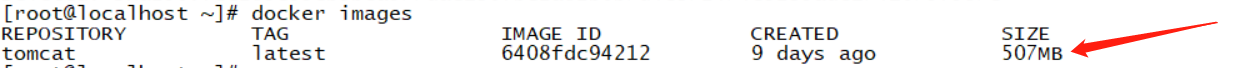
Docker images:列出所有镜像

Docker search 【options】 镜像名字：

Docker pull Tomcat

Docker rmi –f name

如果不写：leatest,不写版本号默认最靠前位置



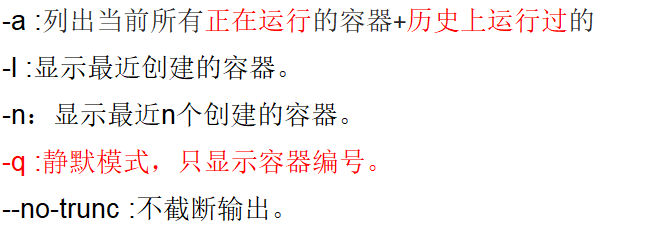
为什么镜像这么大呢？

# Docker 容器命令



Docker run –it [image id]: i :t:以交互伪终端打开

列出当正在运行的容器：docker ps



# 退出命令：

两种退出方式：exit容器停止退出

Ctrl+P+Q 容器不停止退出

# 进入命令

Docker start 容器名称或者容器ID

我们把我们的应用程序和配置依赖打包好形成一个可交付的运行环境，这个打包好的运行环境就是images镜像文件，只有通过这个镜像文件才能后生产一个docker容器。Image文件可以看做事荣器的模板。Docker是根据文件生产容器的实例的。

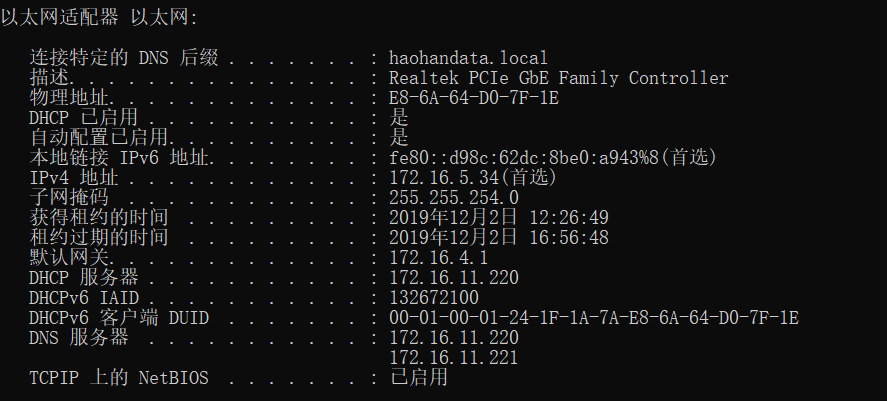
# 附录：

关于centos 7镜像，中的网络配置的坑

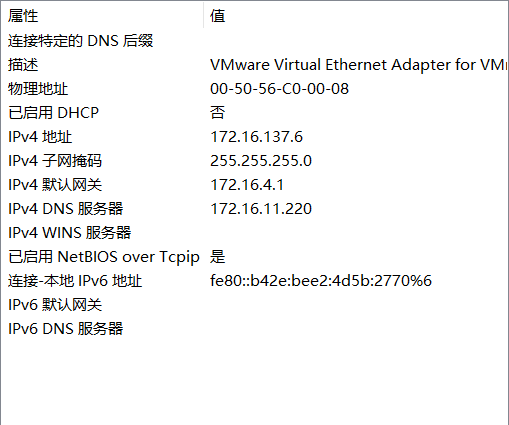
由于我们下载的镜像的版本不同，所以内部配置是不同的。可能会遇到多多少少的坑出现。

下面我就写一下net8虚拟网卡的配置流程：

1. 1：查看本机物理机IP信息：ipconfig –all

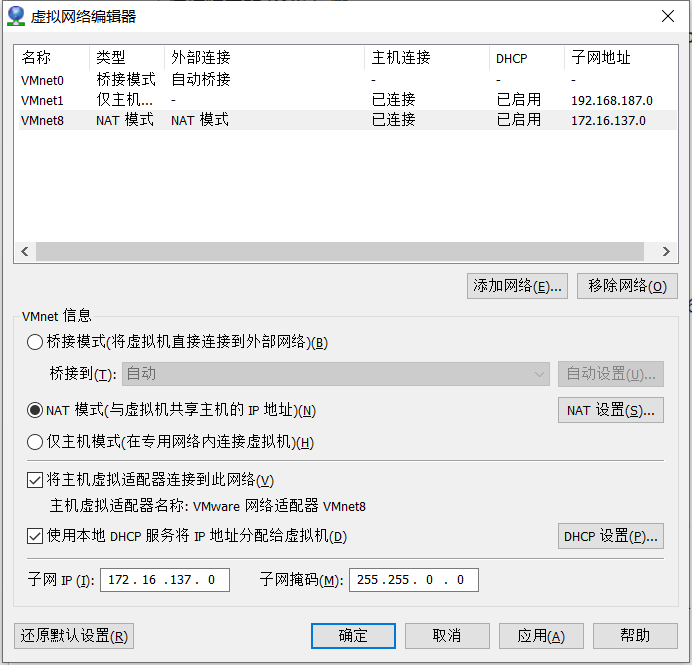


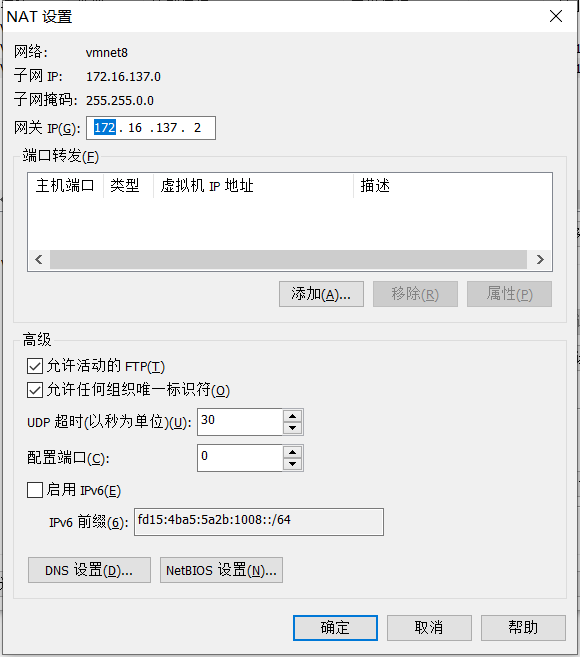
1. 2:配置虚拟网卡的信息：我们选择VMnet8



说明： 我们的虚拟网卡，的网段可以与本机物理网卡的网段不同，目的是为了下次换网的时候，也能正常使用外网。

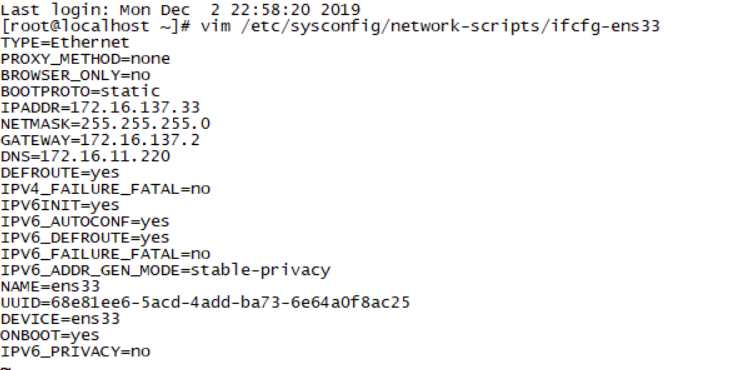
1. 3：打开VM的虚拟网络编辑器，选择VMnet8 模式，进行对网卡信息的修改





1. 4 进入Linux环境内，对内部网卡进行修改

Vim /etc/sysconfig/network-scripts/ifcfg-en33



1. 5重新启动网络

Service network restart

1. 6关闭防火墙

Systemctl stop firewalld.service /// systemctl disable firewalld.service #禁止开机启动

7． Ping [www.baidu.com](http://www.baidu.com)

结果返回报错。Errormessage：Name or service not know

分析报错原因：ping 域名ping不通，说明该域名的IP没有正确解析，那么我们DNS服务器出现了问题，重新配置DNS服务器信息

1. 8 DNS配置文件：vim /etc/resolv.conf

Nameserver 宿主机的DNS地址