# Docker

使用docker镜像启动运行容器的方法,编排镜像,制作自己的镜像.

基于官方的基础镜像制作的.

官方的基础镜像是哪里来的呢?

Scratch镜像是一个神奇的虚拟镜像,之所以说他神奇,是因为他根本不存在,但他还是所有基础镜像发原始模板镜像.

https://hub.docker.com/search?q=centos7type=image

FROM scratch

ADD centos-7-docker.tar.xz / //添加文件到镜像里面

LABEL org.label-schema.schema-version="1.0" \

org.label-schema.name="CentOS Base Image" \ //设置了一个卷标

org.label-schema.vendor="CentOS" \

org.label-schema.license="GPLv2" \

org.label-schema.build-date="20181205"

CMD ["/bin/bash"] //设置了开机启动的解释器

配docker的yum

yum -y install docker-engine

docker version //查看docker 版本

docker search scratch //搜索镜像

有该镜像,下载

docker pull scratch

无法下载

docker images //查看docker 镜像

centos基于scratch创建的

添加了一个文件centos-7.4.1708-docker.tar.xz

scratch是一个空镜像,可以采用如下方法获得(不能下载)

使用/dev/null创建一个空文件(空tar文件)

tar cf scratch.tar --files-from /dev/null

使用docker import导入scratch镜像

docker import scratch.tar scratch

或

tar -cv --files-from /dev/null | docker import - scratch

-代表标准输出

centos-7.4.1708-docker.tar.xz(文件系统,其实就是系统的根目录下的所有文件)

由于系统在未安装之前,我们没有任何命令可以使用,所以只能使用最基本的添加文件方式来创建这个源

yum软件安装到指定根目录 -installroot

最基本的,能登录操作的系统需要的软件包有:

bash 用户登录操作环境,包名bash

yum 安装软件使用 包名yum

ls、pwd等命令 包名coreutils

创建文件夹,安装软件

mkdir vroot //创建虚拟根目录

yum install -y --installroot=/root/vroot bash yum coreutils //往虚拟根目录中安装必要的软件

(如果报错需要密钥,则将gpgcheck改成0)

chroot /root/vroot/ //切换虚拟根为根目录

**初始化用户环境**

修改默认yum源

ls -a /etc/skel/

隐藏文件

\*不包含隐藏文件

cp /etc/skel/.[!.]\* /root/

ls -al /root/

exit

chroot /root/vroot

cd /etc/yum.repos.d/

rm -rf \*

exit

cp /etc/yum.repos.d/local.repo /root/vroot/etc/yum.repos.d/

vim /root/vroot/etc/yum.repos.d/local.repo

cd /root/

打包,要以虚拟目录作为根目录打包

[root@ ~]# tar -cJf centos-7-docker.tar.xz -C vroot ./

开始制作第一个镜像

mkdir ooxx

mv centos-7-docker.tar.xz ooxx/

ls

cd ooxx/

vim Dockerfile

FROM scratch

ADD centos-7-docker.tar.xz /

LABEL org.label-schema.schema-version="1.0" \

org.label-schema.name="CentOS Base Image" \

org.label-schema.vendor="CentOS" \

org.label-schema.license="GPLv2" \

org.label-schema.build-date="20181205"

CMD ["/bin/bash"]

**将官方镜像添加到文件dockerfile中**

docker build -t centos:latest .

docker run -it centos //运行docker容器

yum repolist

yum -y install net-tools //安装网络工具(包含ifconfig)

yum -y install psmisc

docker save centos -o centos.tar //将docker打包备份

**创建与启动容器**

完成一个基本的容器所需要做的六项隔离

Linux内核中就提供了这六种命名空间(namespace)隔离的系统调用

Namespace 系统调用参数 隔离内容

UTS CLONE\_NEWUTS 主机名与域名

IPC CLONE\_NEWIPC 信号量、共享内存

PID CLONE\_NEWPID 进程编号

Network CLONE\_NEWNET 网络设备

Mount CLONE\_NEWNS 挂载点、文件系统

USER CLONE\_NEWUSER 用户和用户组

Linux内核实现namespace的主要目的就是为了实现轻量级虚拟化(容器)服务,在同一个namespace下的进程可以感知彼此的变化,而对外界的进程一无所知,这样就可以让容器中的进程产生错觉，仿佛自己置身于一个独立的系统环境中，以此达到独立和隔离的目的。

unshare命令

**UTS命名空间,主机名隔离**

unshare --uts

hostname bbb

bash //然后主机名变为bbb,但是只在此子进程内生效

**IPC&PID隔离**

unshare --pid --ipc --fork --mount-proc /bin/bash

执行后父进程为bash的子进程,pstree仅能看到该进程和父进程bash

kill信号向量无法发送给被隔离的进程

**Network网络隔离**

unshare --net /bin/bash

仅剩回环地址(lo,即127.0.0.1)

没有网卡

命名空间加chroot构建容器

创建命名空间vhost

ip netns add vhost

ip netns list

设置虚拟交换机

(ip封装VPN,ip封装策略通道,ip )

ip link add name docker0 type bridge stp\_state 1

ip link set dev docker0 up

ip addr add 172.17.0.1/16 brd 172.17.255.255 dev

创建虚拟设备,加入命名空间

ip link add lnic0 type veth peer name rnic0

mkdir -p /var/run/netns

echo $$ 查看PID

ln -s /proc/${Pid}/ns/net /var/run/netns/${Pid}

ip link set rnic0 netns ${Pid} name eth0

**创建虚拟网卡,并添加到虚拟交换机**

虚拟网卡一边连接到虚拟机上,一边连接到docker容器中

ip link add name golbal\_nic type veth peer name ns\_nic

ip link set dev global\_nic procmisc on

ip link set dev global\_nic master docker0

ip link set dev global\_nic up

网卡刚连上后需要等一个广播周期

开启路由转发，让命名空间机器可以访问外部网络

sysctl -w net.ipv4.ip\_forward=1

iptables -t nat -A POSTROUTING -s 172.17.0.0/16 ! -o vbr -j MASQUERADE

ip route replace default via 172.17.0.1 //开启路由转发,设置网关

把虚拟网卡添加进命名空间，并设置 ip 网关

ip link set ns\_nic netns vhost name eth0

ip netns exec vhost ip link set dev lo up

ip netns exec vhost ip link set dev eth0 up

ip netns exec vhost ip addr add dev eth0 172.17.0.2/16 brd 172.17.255.255 dev eth0

文件系统Mount与用户user隔离

unshare --mount //只在此终端(子进程)中可以看到该挂载,其他终端无法查看

命名空间

创建命名空间,与根目录,并进入命名空间

unshare --uts --mount --pid --pic --fork --mount-proc

虚拟目录映射访问内存

mount --bind /proc /var/vroot/proc

设置主机名

hostname vhost

进入虚拟目录

/usr/sbin/chroot /root/vroot

步骤太麻烦怎么办?

systemd-nspawn -M oooxxx -D /root/vroot --network-bridge=docker0

创建一个容器 (沙盒,测试) 名字 根目录 虚拟交换机

ip link set host0 name eth0

ip route replace default via 172.17.0.1