Docker进阶

1. scratch自定义镜像

scratch镜像是一个虚拟镜像,它根本不存在,但还是所有基础镜像的原始模板镜像

* 1. 官方制作Base Image的Dockerfile:

FROM scratch

ADD centos-7-docker.tar.xz /

LABEL name="CentOS Base Image" \

vendor="CentOS" \

license="GPLv2" \

build-date="20170911"

CMD ["/bin/bash"]

· 要想实现基础镜像就需要scratch和centos-7-docker.tar.xz

* 1. 获得scratch镜像

scratch是一个空镜像,不能下载,但可用以下方法获得:

使用 /dev/null创建一个空文件

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

使用docker import导入这个空文件

# docker import scratch.tar scratch

或使用

# tar -cv --file-from /dev/null | docker import - scratch

这样，就有了scratch镜像

* 1. 制作centos-7-docker.tar.xz

centos-7-docker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的方式来创建这个源

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

# yum install --installroot=/root/vroot bash yum coreutils -y

# chroot /root/vroot

# cp -a /etc/skel/.[!.]\* /root/

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

打包,要以虚拟目录作为根打包

# tar -cJf centos-7-docker.tar.xz -C vroot ./

* 1. 万事俱备,开始制作第一个镜像

· 写Dockerfile文件：

# 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-data="20181205"

CMD ["/bin/bash"]

· 打镜像

# docker build -t myos:latest

1. 创建与启动容器

· Docker容器这么火，喜欢技术的朋友可能也会想，如果要实现一个资源隔离的容器，应该从哪些方面下手？

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

* 1. UTS命名空间，主机名隔离

· unshre命令

# man setns

//查看namespace的调用

# unshare --uts

//开启主机名的命名空间

# hostname bbb

# bash

//然后主机名变为bbb，但是只在此子进程内生效

* 1. 文件系统mount与用户user隔离

unshare --mount

# mount -t iso9660 -o loop,ro xxxx.iso /mnt/

//此时，只在此终端(子进程中可以看到该挂载，其他终端无法查看)

* 1. IPC&PID隔离

# ps -ef

# pstree -p

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

# pstree -p

bash(1)——pstree(165)

# kill -9 26517

//报错：no such process

* 1. Network网络隔离

· 虚拟交换机管理工具

# yum install -y bridge-utils

# ip help link

# unshare --net /bin/bash

· 设置虚拟机交换机

(ip命令可以封装隧道，制作虚拟交换机，封装VPN)

# ip link add name docker0 type bridge stp\_state 1

# ip link set dev docker0 up

//设置开启虚拟交换机

# ip addr add 172.17.0.1/16 brd 172.17.255.255 dev docker0

//给交换机设置IP

· 没有网卡怎么办？创建虚拟设备，加入命名空间

# ip link add lnic0 type veth peer name rnic0

//lnic0本地网卡，rnic0远程网卡(塞到容器内)

# mkdir -p /var/run/netns

//创建命名空间文件夹

# ip netns list

# cd /var/run/netns

# echo $$

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

# ip netns list

# ip link set rnic0 netns 27120 name eth0

# ip netns 27120 exec ip addr add 172.17.0.2/16 brd 172.17.255.255 dev eth0

# ip link set dev lnic0 master docker0

# ip link set dev lnic0 up

# systcl -w net.ipv4.ip\_forward=1

# brctl show

# ip netns list

* 1. 手动创建一个容器

· 创建命名空间与根目录，并进入命名空间

# unshare --uts --mount --pid --ipc --fork --mount-proc

· 虚拟目录映射访问内存

# mount --build /proc /var/vroot/proc

· 设置主机名

# hostname vhost

· 进入虚拟目录

# /usr/sbin/chroot/ /var/vroot

· 这么多步骤太麻烦怎么办？

# systemd-nspawn -M xxoo -D /var/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 dev host0 name eth0