该课程适合有 docker 基础的学员

在之前我们的课程中学过了使用 docker 镜像启动运行容器的方法，也学过了如何编排镜像，制作自己的镜像

不过我们都是基于官方的基础镜像制作的

那官方的基础镜像是哪里来的呢？

这里就不得不提 scratch 镜像了

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

我们看官方制作 Base Image 的Dockerfile 是怎么写的

<https://hub.docker.com/search?q=centos&type=image>

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"]

这个基础镜像是基于 scratch 创建的

添加了一个文件 centos-7-docker.tar.xz

要想实现基础镜像就需要 scratch 和 centos-7-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

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

使用 docker import 导入这个空文件到镜像

docker import scratch.tar scratch

或

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

我们已经有了 scratch 镜像，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 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

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

初始化 root 用户登录环境

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

修改默认 yum 源

rm -f vroot/etc/yum.repos.d/\*.repo

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

万事具备，开始制作第一个镜像

docker build –t myos:latest .

创建与启动容器

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

完成一个基本的容器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

1. UTS CLONE\_NEWUTS 主机名与域名
2. IPC CLONE\_NEWIPC 信号量、共享内存
3. PID CLONE\_NEWPID 进程编号
4. Network CLONE\_NEWNET 网络设备
5. Mount CLONE\_NEWNS 挂载点、文件系统
6. 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

文件系统 mount 与用户 user 隔离

unshare --mount

IPC & PID 隔离

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

Network 网络隔离

unshare --net /bin/bash

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

mkdir -p /var/run/netns

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

ip link set rnic0 netns ${Pid} name eth0

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 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 docker0

创建虚拟网卡，并添加到虚拟交换机

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

ip link set dev global\_nic promisc on

ip link set dev global\_nic up

ip link set dev global\_nic master docker0

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

ip netns exec vhost ip route replace default via 172.17.0.1

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

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 --bind /proc /var/vroot/proc

设置主机名

hostname vhost

进入虚拟目录

ip netns exec vhost /usr/sbin/chroot /var/vroot

这么多步骤太麻烦怎么办？

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