### podman容器

podman是一个开源的容器管理工具，操作和docker类似。podman原来是CRI-O项目的一部分，后来被分离成一个单独的项目叫libpod，和docker不同的是podman没有守护进程，直接通过oci runtime来启动容器，命令行操作和docker差不多，docker需要用root权限去操作，而podman则普通用户也可操作

podman官网： https://podman.io/

作者：李茂福

更新日期：2023-09-11

**★podman安装及基本配置**

# yum module install container-tools #安装podman,skopeo,buildah整套管理工具，centos8安装光盘自带有，centos7没有

#podman可以不启动守护进程，也可启动

# systemctl enable podman

# systemctl start podman

★podman默认存储镜像及容器位置： /var/lib/containers/storage

# vi /etc/containers/registries.conf #配置文件，查看默认访问的镜像仓库地址

unqualified-search-registries = ["registry.fedoraproject.org", "registry.access.redhat.com", "registry.centos.org", "docker.io"]

RHEL 8里面的配置为

[registries.search]

registries = ['registry.redhat.io', 'quay.io', 'docker.io']

[registries.insecure]

registries = []

[registries.block]

registries = []

# vi /etc/containers/registries.conf #指定要用的镜像源，文件内容如下：（与上面的写法不可同时使用）

unqualified-search-registries = [ "docker.io" ] #search镜像时访问的地址

[[registry]]

prefix = "cof-lee.com:5443" #不指定镜像名称前缀时，默认带上此前缀

location = "cof-lee.com:5443" #pull拉镜像时访问的地址

insecure = true #信任不安全的ssl证书

**★podman登录仓库及下载镜像**

# podman login -u admin -p passwdxxx --tls-verify=false \

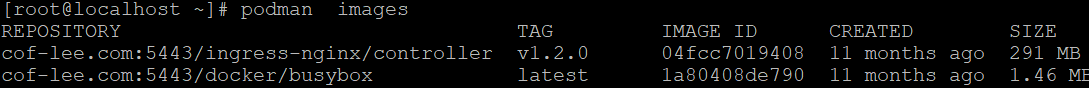
cof-lee.com:5443 #登录镜像仓库，若不指定仓库地址，则默认登录的是unqualified-search-registries指定的仓库地址

Login Succeeded!

# podman pull cof-lee.com:5443/k8s/nginx:v1.19.5 --tls-verify=false #下载镜像

# --tls-verify=false不验证镜像仓库服务器的tls证书，私有镜像则要求前缀同配置里prefix=指定的前缀

#podman images #默认查看当前宿主机上的docker镜像



镜像全名由Repository和Tag组成（中间用冒号连接），如上图，显示有2个镜像：

第一列Repository表示仓库，即镜像主体名，第二列的Tag为版本号，latest表示最新版本，当使用镜像时，若不指定Tag则默认表示使用最新版本 :latest版本。

podman images查看到的镜像的属性有名称，还有镜像ID及大小，镜像ID是唯一的，由12个十六进制字符表示，镜像大小不固定，有大有小，大的可以有10来个GB，小的10来MB甚至几百KB

# podman inspect 镜像id #查看镜像详细信息

# podman rmi 镜像名 #默认是删除本地镜像

#podman tag 原镜像名 新镜像名 #给本地镜像打上新的名称

可以改Repository和Tag名，原来的镜像并不会被重命名，而是复制了一个副本，给副本打上新的名字

# podman push cof-lee.com:5443/k8s/testnginx:v1 --tls-verify=false #上传镜像

**★podman镜像操作**

# podman image ls --format "table {{.ID}}\t{{.Repository}}:{{.Tag}}" #以指定格式显示

# podman inspect 镜像id #查看镜像详细信息

# podman search 软件名 #查找目标软件对应的镜像

#podman pull 镜像名 #下载目标镜像，镜像名要写search里看到的名称

# podman rmi 镜像名 #默认是删除本地镜像

#podman tag 原镜像名 新镜像名 #给本地镜像打上新的名称

可以改Repository和Tag名，原来的镜像并不会被重命名，而是复制了一个副本，给副本打上新的名字

# podman save 镜像名 -o 导出的文件名.tar #导出镜像到单一tar文件

# podman load -i centos7.tar #从tar包导入镜像，导入时不可指定镜像名

# podman images #可查看导入的镜像，可以导入后再修改镜像名称

# podman save imgName1 imgName2 imgNamexx -o xxx.tar #导出多个镜像到一个tar文件中

# podman run -it --name xxx 镜像名 参数 #运行一个镜像，命名为xxx

**★podman运行容器**

**①启动容器**

# podman run 镜像名#运行一个镜像，运行时的环境就是一个容器

# run后面的参数：

-d 后台运行

-i 交互式操作

-t 给容器提供一个shell终端

--name xxx 给容器实例命名，若不指定容器名，则系统会随机给它一个名字

--name=xxx 同上，给容器实例命名

# podman run -d 镜像名 #让容器在后台运行

# podman run -it 镜像名 #让容器在前台运行，并提供一个终端与容器交互，这时如果容器有默认的程序在运行的话，我们就可以和这个程序交互了，若没有默认程序运行或该程序不与用户交互，则我们就什么也做不了

# podman run -it 镜像名 /bin/sh #最后的/bin/sh是传给容器的参数，表示运行这个程序与我们交互，即linux系统的sh，然后就可以在sh命令行操作了

在容器里直接输入exit后，就退出容器了，**容器就停止运行了！**

# podman ps #查看正在后台运行的容器

# podman ps -a #查看所有运行过的容器及正在后台运行的容器

★容器的id与镜像的id是不一样的

# podman stop 容器id #停止正在运行的这个容器，容器id可以换成容器名

# podman start 容器id #启动一个已经停止的容器

# podman restart 容器id #重启一个容器

**②进入容器并与之交互**

如果容器在后台运行，即run -d的，如何再进入这个容器并与之交互呢？

# podman exec -it 容器id #进入正在后台运行的容器，并与之交互

....

> exit #在容器里输入exit后退出容器

★使用podman exec进入容器后，再exit退出的话，该容器仍在后台运行，并不会停止

# podman rm -f 容器id #强制删除容器，包括正在运行的及已停止的

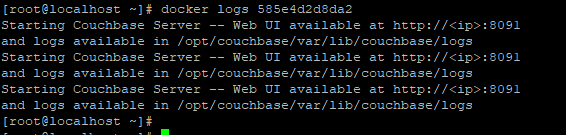
# podman container prune #删除所有已停止运行的容器，停止的容器也是要占用系统的存储空间的，确定不用了的容器就可删除

**③容器信息查看**

# podman inspect 容器id/容器名 #查看容器的详细信息

#容器的信息储存在podman数据目录下的containers子目录下

# podman logs 容器id/容器名 #查看容器里的标准输出信息



# podman logs -f 容器id/容器名 #不断地输出容器里的标准输出信息，

# 就像top命令那样，不退出显示界面，动态输出信息按Ctrl+C退出

**★podman容器随开机自启**

podman容器的开机自启和docker的配置不一样！

# podman run -d --name=hello8 cof-lee.com:5443/k8s/nginx:v1.19.5 #先创建一个容器

# loginctl enable-linger root #使用户注销登录后也能继续运行它管理的服务，且系统重启后此用户不登录也能自启动它管理的服务；取消命令为 loginctl disable-linger root

# loginctl show-user root | grep Linger

Linger=yes

# mkdir -p ~/.config/systemd/user #创建用户服务文件保存目录

# cd ~/.config/systemd/user/

# podman generate systemd --name hello8 --files

#生成相应的服务文件： ~/.config/systemd/user/container-hello8.service

# systemctl --user daemon-reload

Failed to connect to bus: No such file or directory

#原因是此用户是从其他用户su - 切换的，需要使用ssh远程登录此用户才可执行以上命令

# systemctl --user daemon-reload #用户ssh远程登录后再执行此命令

# systemctl --user status container-hello8.service

# systemctl --user enable container-hello8.service

# systemctl --user start container-hello8.service

# cat container-hello8.service #podman生成的systemd服务配置文件如下

[Unit]

Description=Podman container-hello8.service

Documentation=man:podman-generate-systemd(1)

Wants=network-online.target

After=network-online.target

RequiresMountsFor=/run/containers/storage

[Service]

Environment=PODMAN\_SYSTEMD\_UNIT=%n

Restart=on-failure

TimeoutStopSec=70

ExecStart=/usr/bin/podman start hello8

ExecStop=/usr/bin/podman stop -t 10 hello8

ExecStopPost=/usr/bin/podman stop -t 10 hello8

PIDFile=/run/containers/storage/overlay-containers/f8818526c6f1a968165841c105e111dd248b8d2dfcfce6cb9fba49e5c0652f40/userdata/conmon.pid

Type=forking

[Install]

WantedBy=multi-user.target default.target

#最后，就算重启系统，用户不登录，此podman容器也是随系统开机启动的

**★podman端口映射**

容器里的网络环境默认为nat的，也就是所有容器自己会获得一个私有ip，然后它访问外部时是要转为宿主系统的ip，外部没法直接访问容器里的Ip和端口号，得做个dNat端口映射

# podman run -d -p 宿主端口:容器端口 镜像名 #用-p指定端口映射

**例如:**

-p 9999:80 #把宿主上的9999端口映射到容器里的80端口

★端口组映射：（端口数量要一致）

-p 800-810:900-910 #把宿主上的800到810端口组映射到容器里的900到910端口

★ -p xx:xx可有多个，做多个端口的映射：

-p 100:100 -p 800-820:800-820 -p 999:999

★默认是tcp的端口，可在端口后加 /udp表示使用udp协议的端口

-p 53:53/udp

# podman run -d -p 999:80 cof-lee.com:5443/k8s/nginx:v1.19.5 #将宿主机的999端口映射到容器的80端口

# podman port 容器id/容器名 #查看容器的端口映射情况

80/tcp -> 0.0.0.0:999

**★podman目录映射**

容器里的数据是保存的临时的容器存储位置的，如果不小心把容器删除或清理了，那数据就再也没有了，而且数据直接放在容器里，也不方便管理和备份。最好是做个目录映射，把宿主系统上的某个目录映射到容器里的某目录下，然后容器的服务指定存放数据的目录到映射的目录下就行了。这些容器产生的数据就会直接保存到宿主系统的某目录下。

使用:Z选项，自动将selinux的container\_file\_t标签应用到主机目录上

# podman run -d --name xxx -v /hostdir:/usr/poddir/xxx/:Z imagexx

#要在容器运行时指定映射的目录，2个目录中间是一个冒号，冒号左右无空格

#从宿主目录复制某文件到容器里

#从容器里复制文件到宿主系统下

**★podman网络**

安装podman之后，默认有个虚拟网桥cni-podman0

# ip addr

3: cni-podman0: <BROADCAST,MULTICAST,UP,LOWER\_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000

link/ether ba:70:29:53:18:b0 brd ff:ff:ff:ff:ff:ff

inet 10.88.0.1/16 brd 10.88.255.255 scope global cni-podman0

valid\_lft forever preferred\_lft forever

# brctl show cni-podman0

bridge name bridge id STP enabled interfaces

cni-podman0 8000.ba70295318b0 no vethfdb22f9e