配置无根环境使用Podman&配置无根环境的网络

原创 夜风轻快 已于 2022-08-26 15:45:42 修改 阅读量1.2k 收藏 点赞数

分类专栏: Docker 文章标签: docker linux 运维 容器

Podman的Rootless环境运行容器

文章目录

一、配置rootless用户运行容器
- 更改默认的OCIRuntime为crun
- 安装slirp4netns和fuse-overlayfs
- /etc/subuid和/etc/subgid的配置
- usermod
- 用户配置文件
- 授权文件
- rootless用户是无法看见root用户的镜像容器
- 存储卷
- 映射端口

二、配置rootless用户的网络
- 有根容器网络和无根容器网络之间的区别
- 防火墙
- 基本网络设置
- Bridge(桥接模式)
  - 创建网桥

一、配置rootless用户运行容器

参考自：在无根环境中Podman的基本设置和使用

更改默认的OCIRuntime为crun

在允许没有root特权的用户运行 Podman 之前，管理员必须安装或构建Podman并完成以下配置

cgroup V2Linux内核功能允许用户限制普通用户容器可以使用的资源，如果使用cgroupV2启用了运行Podman的Linux 发行版，则可能需要更改默认的OCI运行时。某些较旧的版本runc不适用于cgroupV2，必须切换到备用OCI运行时crun。

bash AI写代码 免登录复制

```bash
[root@localhost ~]# yum -y install crun
[root@localhost ~]# vim /usr/share/containers/containers.conf
431 # Default OCI runtime
433 runtime = "crun"         //取消注释crun
434 #runtime = "runc"        //注释runc

//运行一台容器用来查看OCI运行时是否是crun
[root@localhost ~]# podman run -d --name web -p 80:80 httpd
9603ad989cbcf751510a385c99fa8d2a53ac10dadd9b95c52f199993215acb3e
[root@localhost ~]# podman inspect web | grep crun
        "OCIRuntime": "crun",
            "crun",

```

收起 ^

安装slirp4netns和 fuse -overlayfs

在普通用户环境中使用Podman时，建议使用fuse-overlayfs而不是VFS文件系统，至少需要版本0.7.6。现在新版本默认就是了。

bash AI写代码 免登录复制

```bash
[root@localhost ~]# yum -y install slirp4netns
```

```
[root@localhost ~]# vim /etc/containers/storage.conf
77 mount_program = "/usr/bin/fuse-overlayfs"         //取消注释
```

/etc/subuid和/etc/subgid的配置

Podman要求运行它的用户在/etc/subuid和/etc/subgid文件中列出一系列UID，shadow -utils或newuid包提供这些文件

这个文件的格式是 `USERNAME:UID:RANGE`

- `/etc/passwd` 的输出中或中列出的用户名 `getpwent` 。也就是映射到系统上的用户
- 为用户分配的初始 UID。
- 为用户分配的 UID 范围的大小。

bash AI写代码 免登录复制

```
[root@localhost ~]# yum -y install shadow-utils
//每个用户的值必须唯一且没有任何重叠。该值从100000开始，65536是值的范围，值的范围可自定义。
//起始值+值范围＝下个用户的值
[root@localhost ~]# useradd fish
[root@localhost ~]# cat /etc/subuid
yf:100000:65536
fish:165536:65536
[root@localhost ~]# cat /etc/subgid
yf:100000:65536
fish:165536:65536

//启动非特权ping
[root@localhost ~]# sysctl -w "net.ipv4.ping_group_range=0 200000"      //临时启动
//永久启动
[root@localhost ~]# vim /etc/sysctl.conf
[root@localhost ~]# tail -1 /etc/sysctl.conf
net.ipv4.ping_group_range=0 200000              //意思是只让0-200000范围内的用户id使用podman
//打印sysctl的配置，验证是否生效
[root@localhost ~]# sysctl -p
net.ipv4.ping_group_range = 0 200000
```

usermod

usermod程序可用于为用户分配 UID 和 GID，而不是直接更新文件。

bash　　AI写代码　　免登录复制

```bash
//创建用户，用户必须事先存在
[root@localhost ~]# useradd jerry
[root@localhost ~]# cat /etc/subuid
yf:100000:65536
fish:165536:65536
jerry:231072:65536
[root@localhost ~]# cat /etc/subgid
yf:100000:65536
fish:165536:65536
jerry:231072:65536

//使用usermod修改UID与GID。由于该用户的已有ID，须先删除再分配
//第一个填起始值，第二个结束值。结束值=起始值+值的范围
[root@localhost ~]# usermod --del-subuids 231072-296608 --del-subgids 231072-296608 jerry
//可看到在该文件中jerry用户不存在了，因为没有uid与gid
[root@localhost ~]# cat /etc/subuid
yf:100000:65536
fish:165536:65536
[root@localhost ~]# cat /etc/subgid
yf:100000:65536
fish:165536:65536
//给jerry分配UID和GID
[root@localhost ~]# usermod --add-subuids 300000-302000 --add-subgids 300000-302000 jerry
//可以看到又出现了，并且UID与GID是刚才分配的值
[root@localhost ~]# cat /etc/subgid
yf:100000:65536
fish:165536:65536
jerry:300000:2001
[root@localhost ~]# cat /etc/subuid
yf:100000:65536
fish:165536:65536
```

用户配置文件

三个主要的配置文件是 `container.conf` 、 `storage.conf` 和 `registries.conf` 。用户可以根据需要修改这些文件。

container.conf

text AI写代码 免登录复制

```
//用户配置文件，列出的三个文件不是说必须全都有，而是在其中任意一个文件写入配置都有效
/usr/share/containers/containers.conf
/etc/containers/containers.conf
~/.config/containers/containers.conf  //优先级最高
```

如果它们以该顺序存在。每个文件都可以覆盖特定字段的前一个文件。

配置storage.conf文件

text AI写代码 免登录复制

```
/etc/containers/storage.conf
$HOME/.config/containers/storage.conf
```

在普通用户中 `/etc/containers/storage.conf` 的一些字段将被忽略

bash AI写代码 免登录复制

```
[root@localhost ~]# vim /etc/containers/storage.conf
 5 [storage]
 7 # Default Storage Driver, Must be set for proper operation.
 8 driver = "overlay"       //取消该行注释，默认是没有注释的

//配置使用podman最大的用户数，一般不推荐设置，了解有这种用法即可
[root@localhost ~]# sysctl user.max_user_namespaces=15000       //临时配置
//永久配置
[root@localhost ~]# vim /etc/sysctl.conf
user.max_user_namespaces = 15000    //在文件写入这行
//打印/etc/sysctl.conf配置文件生效的内容
[root@localhost ~]# sysctl -p
```

在普通用户中这些字段默认

python AI写代码 免登录复制 运行

```python
graphroot="$HOME/.local/share/containers/storage"
runroot="$XDG_RUNTIME_DIR/containers"
```

$XDG_RUNTIME_DIR$在大多数系统上默认为‘$/run/user/$UID`

registries.conf

配置按此顺序读入,这些文件不是默认创建的，可以从 `/usr/share/containers` 或复制文件 `/etc/containers` 并进行修改。

bash AI写代码 免登录复制

```bash
/etc/containers/registries.conf
/etc/containers/registries.d/*
HOME/.config/containers/registries.conf
```

授权文件

此文件里面写了docker 账号的密码，以加密方式显示

bash AI写代码 免登录复制

```bash
[root@localhost ~]# podman login
Username: guguniao
Password:
Login Succeeded!
[root@localhost ~]# cat /run/user/0/containers/auth.json
{
        "auths": {
                "docker.io": {
                        "auth": "Z3VndW5pYW86eWVmZW5nMTEyMQ=="
                }
        }
```

```
[root@localhost ~]# podman logout
Removed login credentials for docker.io
[root@localhost ~]# cat /run/user/0/containers/auth.json
{
        "auths": {}
}
```

收起 ^

rootless用户是无法看见root用户的镜像容器

bash AI写代码 免登录复制

```
[root@localhost ~]# podman images
REPOSITORY                              TAG         IMAGE ID      CREATED      SIZE
registry.fedoraproject.org/f29/httpd  latest      25c76f9dcdb5  3 years ago  482 MB
[root@localhost ~]# podman container ls -a
CONTAINER ID  IMAGE                                     COMMAND               CREATED            STATUS                I
9603ad989cbc  registry.fedoraproject.org/f29/httpd:latest  /usr/bin/run-http...  About an hour ago  Up About an hour ago  (
```

bash AI写代码 免登录复制

```
[yf@localhost ~]$ podman images
REPOSITORY  TAG         IMAGE ID    CREATED     SIZE
[yf@localhost ~]$ podman container ls -a
CONTAINER ID  IMAGE       COMMAND     CREATED     STATUS      PORTS       NAMES
```

存储卷

容器与root用户一起运行，则root容器中的用户实际上就是主机上的用户。
UID GID是在/etc/subuid和/etc/subgid等中用户映射中指定的第一个UID GID。
如果普通用户的身份从主机目录挂载到容器中，并在该目录中以根用户身份创建文件，则会看到它实际上是你的用户在主机上拥有的。

bash AI写代码 免登录复制

```
[yf@localhost ~]$ mkdir ~/data
[yf@localhost ~]$ ls
```

```
bin   data  dev   etc   home  proc  root  run   sys   tmp   usr   var
/ # cd /data/
/data # touch 123
/data # ls
123
//在宿主机上查看
[yf@localhost ~]$ cd data/
[yf@localhost data]$ ls
123
[yf@localhost data]$ echo "hello world!" > 123
//在容器里查看
/data # cat 123
hello world!
//我们可以发现在容器里面的文件的属主和属组都属于root，那么如何才能让其属于tom用户呢？继续往下看
/data # ls -l
total 4
-rw-r--r--    1 root     root            13 Aug 16 14:20 123

//只要在运行容器的时候加上一个--userns=keep-id即可
[yf@localhost ~]$ podman run -it -v "$(pwd)"/data:/data --userns=keep-id  busybox  /bin/sh
~ $ cd /data/
/data $ ls -l
total 4
-rw-r--r--    1 yf       yf              13 Aug 16 14:20 123
```

收起 ^

映射端口

使用rootless用户映射容器端口时会报“ permission denied”的错误

bash　　AI写代码　　免登录复制

```
[yf@localhost ~]$ podman run  -d -p 82:80 httpd
Error: rootlessport cannot expose privileged port 82, you can add 'net.ipv4.ip_unprivileged_port_start=82' to /etc/sysctl.
```

注意：rootless用户只能映射>=1024的端口

夜风轻快

bash AI写代码 免登录复制

```
[yf@localhost ~]$ podman run -d -p 1024:80 httpd
64743ca47d56f580b648ee450e1096052e2b303c616417bb51868e2884b2d794
[yf@localhost ~]$ podman port 64743ca47d56
80/tcp -> 0.0.0.0:1024
[yf@localhost ~]$ ss -anlt
State         Recv-Q        Send-Q               Local Address:Port               Peer Address:Port        Process
LISTEN        0             128                        0.0.0.0:1024                    0.0.0.0:*
LISTEN        0             128                        0.0.0.0:80                      0.0.0.0:*
LISTEN        0             128                        0.0.0.0:22                      0.0.0.0:*
LISTEN        0             128                           [::]:22                         [::]:*
```

收起 ^

在/etc/sysctl.conf配置 `net.ipv4.ip_unprivileged_port_start=80` 后可以映射大于等于80的端口

bash AI写代码 免登录复制

```
[root@localhost ~]# echo  'net.ipv4.ip_unprivileged_port_start=80'  >> /etc/sysctl.conf
[root@localhost ~]# sysctl -p
net.ipv4.ping_group_range = 0 200000
user.max_user_namespaces = 15000
net.ipv4.ip_unprivileged_port_start = 80

[yf@localhost ~]$ podman run -d -p 82:80 httpd
d46902105ba54bea7dc30ae0d9c0cfa70c943087168b1e3402fab82bb7ebd118
[yf@localhost ~]$ ss -anlt
State         Recv-Q        Send-Q               Local Address:Port               Peer Address:Port        Process
LISTEN        0             128                        0.0.0.0:1024                    0.0.0.0:*
LISTEN        0             128                        0.0.0.0:80                      0.0.0.0:*
LISTEN        0             128                        0.0.0.0:82                      0.0.0.0:*
LISTEN        0             128                        0.0.0.0:22                      0.0.0.0:*
LISTEN        0             128                           [::]:22                         [::]:*
```

收起 ^

夜风轻快

参考自：Podman官方的网络基本指南

有根容器网络和无根容器网络之间的区别

Podman 容器联网的指导因素之一是容器是否由 root 用户运行。这是因为rootless用户无法在主机上创建网络接口。因此，对于无根容器，默认的网络模式是 slirp4netns。由于权限的限制，slirp4netns 相比 rootful Podman 的联网，缺乏联网的一些特性；例如，slirp4netns 不能给容器一个可路由的 IP 地址。另一端的 rootful 容器的默认联网模式是 netavark，它允许容器有一个可路由的 IP 地址。

防火墙

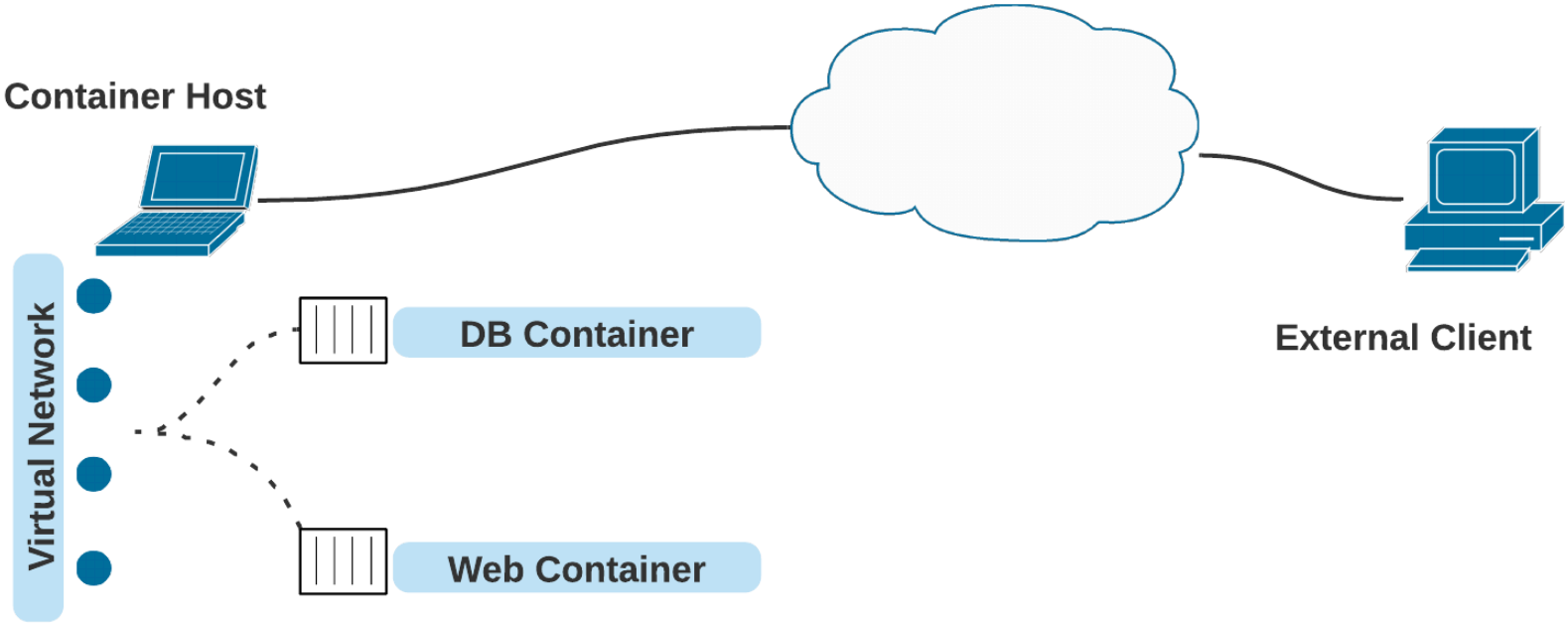
防火墙的作用不会影响网络的设置和配置，但会影响这些网络上的流量。最明显的是到容器主机的入站网络流量，通常通过端口映射传递到容器。根据防火墙实现，我们观察到防火墙端口由于运行具有端口映射的容器而自动打开（例如）。如果容器流量似乎无法正常工作，请检查防火墙并允许容器正在使用的端口上的流量。一个常见的问题是重新加载防火墙会删除 cni/netavark iptables 规则，从而导致 rootful 容器的网络连接丢失。Podman v3 提供了 podman network reload 命令来恢复它，而无需重新启动容器。

基本网络设置

大多数使用 Podman 运行的容器和 Pod 都遵循几个简单的场景。默认情况下，rootful Podman 将创建一个桥接网络。这是 Podman 最直接和首选的网络设置。桥接网络在内部桥接网络上为容器创建一个接口，然后通过网络地址转换 (NAT) 连接到 Internet。我们还看到用户也希望 `macvlan` 用于网络。这 `macvlan` 插件将整个网络接口从主机转发到容器中，允许它访问主机所连接的网络。最后，无根容器的默认网络配置是 slirp4netns。slirp4netns 网络模式功能有限，但可以在没有 root 权限的用户上运行。它创建从主机到容器的隧道以转发流量。

Bridge(桥接模式)

桥接网络被定义为在连接容器和主机的地方创建内部网络。然后这个网络能够允许容器在主机之外进行通信。



上面的插图描绘了一个笔记本电脑用户运行两个容器：一个 web 和 db 实例。这两个容器与主机位于虚拟网络上。此外，默认情况下，这些容器可以启动笔记本电脑外部的通信（例如到 Internet）。虚拟网络上的容器通常通常不可路由，也称为私有 IP 地址。

当处理在主机外部发起的通信时，外部客户端通常必须寻址笔记本电脑的外部网络接口和给定的端口号。假设主机允许传入流量，主机将知道将该端口上的传入流量转发到特定容器。为此，当容器请求转发特定端口时，会添加防火墙规则来转发流量。

桥接网络是作为 root 创建的 Podman 容器的默认设置。Podman 提供了一个默认的桥接网络，但您可以使用该 `podman network create` 命令创建其他网络。容器可以在使用 `--network` 标志创建时加入网络，或者在通过 `podman network connect` and `podman network disconnect` 命令创建后加入网络。

如前所述，slirp4netns 是无根用户的默认网络配置。但是从 Podman 4.0 版开始，无根用户也可以使用 netavark。无根netavark的用户体验与有根netavark非常相似，只是没有提供默认的网络配置。您只需要创建一个网络，该网络将被创建为一个桥接网络。如果您想从 CNI 网络切换到 netavark，您必须发出

创建网桥

bash AI写代码 免登录复制

```
[yf@localhost ~]$ podman network create
/home/yf/.config/cni/net.d/cni-podman1.conflist
//podman是root的默认网桥。cin-podman1是刚刚新建的网桥。
[yf@localhost ~]$ podman network ls
NETWORK ID    NAME         VERSION     PLUGINS
2f259bab93aa  podman       0.4.0       bridge,portmap,firewall,tuning
b932778640d3  cni-podman1  0.4.0       bridge,portmap,firewall,tuning
```

当运行无根容器时，网络操作将在一个额外的网络命名空间内执行。要加入此命名空间，请使用 `podman unshare --rootless-netns`

运行容器加入网络名称空间

bash AI写代码 免登录复制

```
[yf@localhost ~]$ podman container run -it --name test --network cni-podman1 busybox
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: eth0@if5: <BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1500 qdisc noqueue
    link/ether 9a:e6:ad:45:ca:d2 brd ff:ff:ff:ff:ff:ff
    inet 10.89.0.3/24 brd 10.89.0.255 scope global eth0         //可以看到有IP
       valid_lft forever preferred_lft forever
    inet6 fe80::98e6:adff:fe45:cad2/64 scope link
       valid_lft forever preferred_lft forever
/ # ping baidu.com                                             //ping通外网了
PING baidu.com (39.156.66.10): 56 data bytes
64 bytes from 39.156.66.10: seq=0 ttl=254 time=32.491 ms
64 bytes from 39.156.66.10: seq=1 ttl=254 time=127.453 ms
^C
```

```
[yf@localhost ~]$ podman container inspect test1 | grep -i addr     //也可以用这种方式查看容器的IP
            "IPAddress": "",
            "GlobalIPv6Address": "",
            "MacAddress": "",
            "LinkLocalIPv6Address": "",
                   "IPAddress": "10.89.0.3",
                   "GlobalIPv6Address": "",
                   "MacAddress": "1e:86:11:13:fb:37",
```

收起 ^

rootless用户的容器**没有ip也可以访问外网**，是通过 `slirp4netns` 进行流量转发的。

bash　　AI写代码　　免登录复制

```
[yf@localhost ~]$ podman run -itd --name test2 busybox
23ddf6b711c1653ee9e661492a7c20f86d32fb56a0621ebdbb54c0a887349ed3
[wxh2@podman ~]$ podman inspect -l |grep -i ipaddress
            "IPAddress": "",

//但是没有ip不影响访问外网，它会生成一张tap0的虚拟网卡
[yf@localhost ~]$ podman exec -it -l /bin/sh
/ # ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: tap0: <BROADCAST,UP,LOWER_UP> mtu 65520 qdisc fq_codel qlen 1000
    link/ether ae:ae:1d:87:ec:33 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.100/24 brd 10.0.2.255 scope global tap0
       valid_lft forever preferred_lft forever
    inet6 fe80::acae:1dff:fe87:ec33/64 scope link
       valid_lft forever preferred_lft forever
/ # ping www.baidu.com          //访问百度是完全没有问题的
PING www.baidu.com (112.80.248.75): 56 data bytes
```

```
64 bytes from 112.80.248.75: seq=2 ttl=255 time=43.169 ms
^C
--- www.baidu.com ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 41.726/47.623/57.974 ms
```

收起

显示推荐内容