podman

Rootless Networking

Podman Community Meeting - October 5, 2021

Rootless Networking

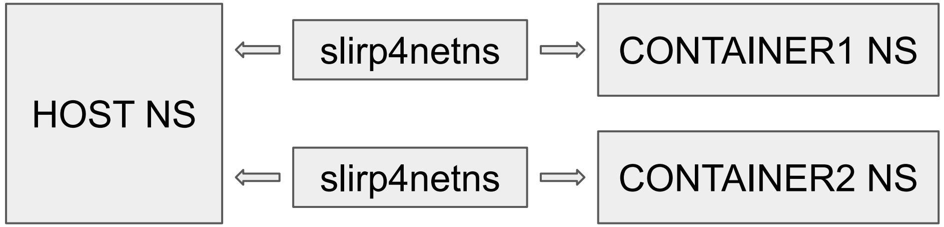
- Networking without privileges

  -> Cannot modify host network namespace

  -> Needs user space application to proxy network connections from and into the container

Slirp4netns

- User-mode networking for unprivileged network namespaces
- Uses the unprivileged “slirp” network stack and connects this to a tap interface in the container namespace
- Provides internet connectivity for the container
- Supports port forwarding from the host to the container



slirp4netns(1)

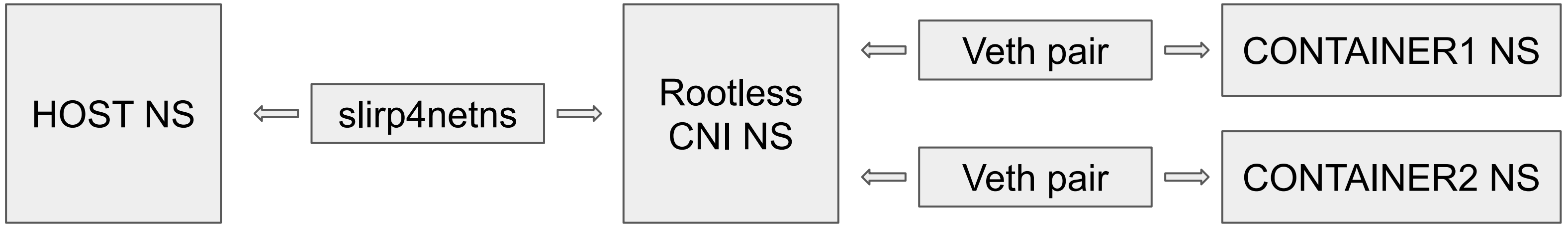
Slirp4netns settings in podman

- podman run --network slirp4netns (default for rootless users)
- allow_host_loopback=true|false: Allow the container process to reach the host loopback IP via 10.0.2.2. Default is false.
- enable_ipv6=true|false: Enable ipv6 support. Default is false.
- port_handler=rootlesskit|slirp4netns: Change the port forwarder, by default rootlesskit is used. Rootlesskit is faster than slirp4netns but it changes the source ip the an internal ip in the container namespace.
- podman run --network slirp4netns:enable_ipv6=true,port_handler=slirp4netns
- Inter container communication only possible via forwarded ports

podman-run(1)

Rootless CNI networking

- Uses extra network namespace to execute the CNI plugins
- Only works for bridge networks, macvlan works in theory but it can only use interfaces inside the rootless CNI NS and not the interfaces from the host
- Inter container communication works out of the box
- Still uses slirp4netns to provide internet connection and only works with the rootlesskit port forwarder



Rootless CNI networking

- podman network create mynet && podman run --network mynet …
- The IP address assigned to the container is not reachable from the host network namespace
- To join the rootless CNI network namespace use podman unshare --rootless-cni, use this to execute commands inside the namespace

```
$ podman run -d --network cni-podman2 nginx
3276a0ecdb2b09ca392262be45f22829421fff27165eea261b1d205f81aea5d1
$ podman container inspect -l --format "{{(index .NetworkSettings.Networks \"cni-podman2\").IPAddress}}"
10.88.3.2
$ curl --max-time 3  10.88.3.2
curl: (28) Connection timed out after 3000 milliseconds
$ podman unshare --rootless-cni curl --max-time 3  10.88.3.2
<!DOCTYPE html>
<html>
...
```

DIY networking

- If you need more advanced setup you can setup your own interfaces.
- Creating interfaces on the host requires root privileges.
- For this to work create a container with *--network=none* and run *podman container init <name>*. This sets up all namespaces.
- Use *podman container inspect --format {{.State.Pid}} <name>* to get the process pid.
- Move a host interface into the container with *sudo ip link set $interface netns $pid*
- The manual setup has to be done every time the container is started
- An example bridge setup by Rudolf Vesely can be found [here]