3 - Building Blocks of Containers

Prerequisites for lab

- Create a Ubuntu VM instance on DigitalOcean. Please refer the `Setting up the lab` section of the course for details.

Namespaces

Namespaces are a feature of the Linux Kernel, which are used to isolate and virtualize system resources between processes. System resources that can be virtualized are:-

- Mount (mnt)
- used to isolate mount points
- it is similar to `chroot`, but with improved security.
- Process ID (pid)
- used to isolate PIDs
- virtual PIDs can be the same in different namespaces
- each virtual PID is mapped to a different PID on the host system.

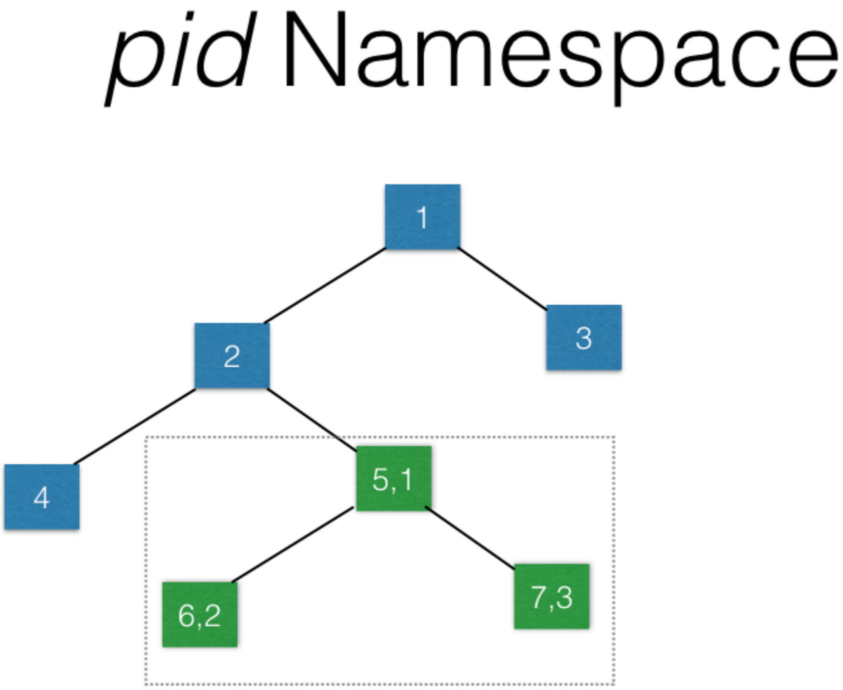


Figure 1: pid namespace

- Network (net)

- /proc/net, IPs, Interfaces and routes are isolated between network namespaces.
- Interprocess Communication (ipc)
- SystemV IPC and POSIX Message Queues can be isolated.
- UTS (hostname)
- used to isolate the hostname and NIS name between namespaces.
- User ID (user)
- user and group IDs are different inside and outside of namespaces and can be duplicated.

1. How to list all the existing namespaces for a process?

```
$ ls -l /proc/<pid>/ns

$ ls -l /proc/1/ns
total 0
16823 dr-x--x--x 2 root root 0 Sep 26 13:23 .
 7772 dr-xr-xr-x 9 root root 0 Sep 26 13:11 ..
 16834 lrwxrwxrwx 1 root root 0 Sep 26 13:23 cgroup -> cgroup:[4026531835]
 16830 lrwxrwxrwx 1 root root 0 Sep 26 13:23 ipc -> ipc:[4026531839]
 16833 lrwxrwxrwx 1 root root 0 Sep 26 13:23 mnt -> mnt:[4026531840]
 16828 lrwxrwxrwx 1 root root 0 Sep 26 13:23 net -> net:[4026531957]
 16831 lrwxrwxrwx 1 root root 0 Sep 26 13:23 pid -> pid:[4026531836]
 16832 lrwxrwxrwx 1 root root 0 Sep 26 13:23 user -> user:[4026531837]
 16829 lrwxrwxrwx 1 root root 0 Sep 26 13:23 uts -> uts:[4026531838]
```

2. How to create new network namespaces and connect them ?

Create two `network namespaces` using `netns add` subcommand of the `ip` command.

```
$ ip netns add ns1
$ ip netns add ns2
```

In the following diagram, we can see that two namespaces have been created.

- Create a Veth pair with two interfaces `tap1` and `tap2`. Veth stands for Virtual ETHernet. It is a simple tunnel driver that works at the link layer and looks like a pair of ethernet devices interconnected with each other.

```
$ ip link add tap1 type veth peer name tap2
```

In the following diagram, we can see that a `veth` pair has been created.

Attach one endpoint of the `veth` pair to one of the namespaces.

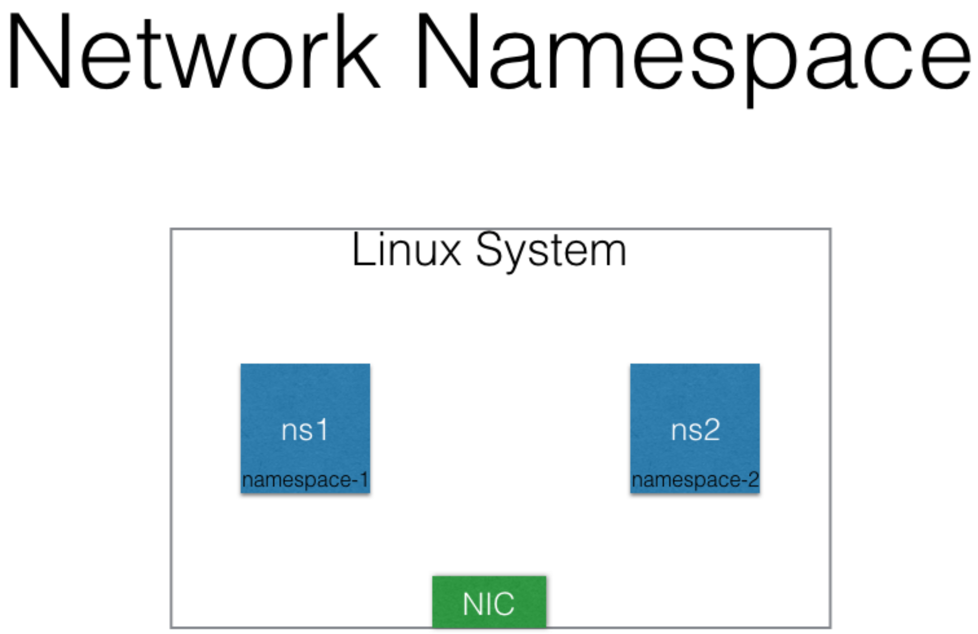
```
$ ip link set tap1 netns ns1
$ ip link set tap2 netns ns2
```

after which, our setup would look like in the following diagram:-

Bring up the link for both interfaces.

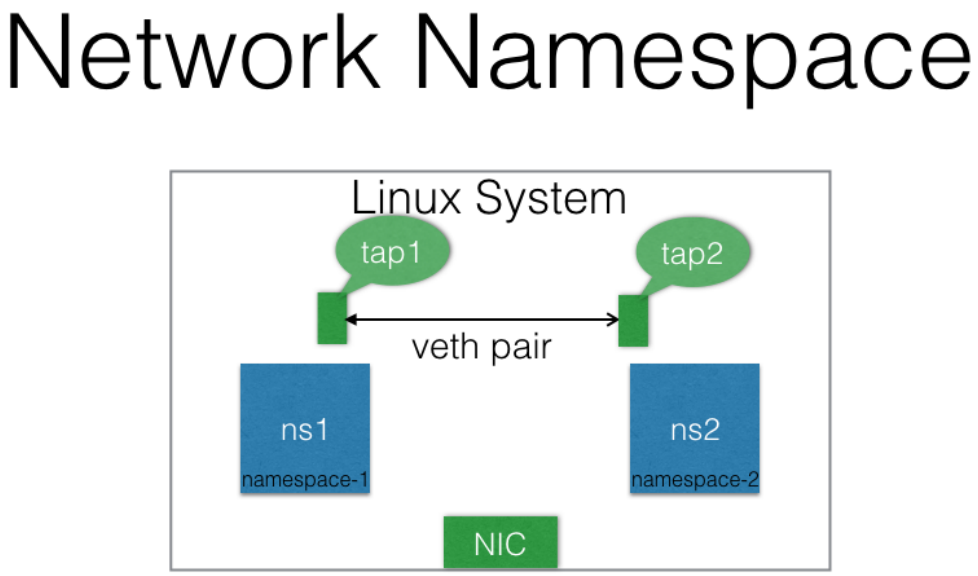
```
$ ip netns exec ns1 ip link set dev tap1 up
$ ip netns exec ns2 ip link set dev tap2 up
```

Assign the IP address to each one of the interfaces, and then `ping` the other endpoint.



Network Namespace

Figure 2: Creating Namespaces



Network Namespace

Figure 3: Creating veth pair

Network Namespace

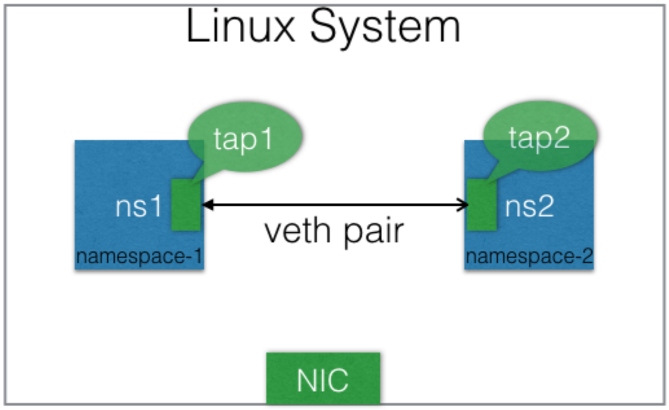


Figure 4: moving interfaced to namespaces

```
$ ip netns exec ns1 ifconfig tap1 192.168.1.1 up
$ ip netns exec ns2 ifconfig tap2 192.168.1.2 up
$ ip netns exec ns2 ping 192.168.1.1
NG 192.168.1.1 (192.168.1.1): 56 data bytes
64 bytes from 192.168.1.1: seq=0 ttl=64 time=0.074 ms
64 bytes from 192.168.1.1: seq=1 ttl=64 time=0.074 ms
^C
--- 192.168.1.1 ping statistics ---
2 packets transmitted, 2 packets received, 0% packet loss
ound-trip min/avg/max = 0.074/0.074/0.074 ms
$ ip netns exec ns1 ping 192.168.1.2
PING 192.168.1.2 (192.168.1.2): 56 data bytes
64 bytes from 192.168.1.2: seq=0 ttl=64 time=0.046 ms
^C
--- 192.168.1.2 ping statistics ---
1 packets transmitted, 1 packets received, 0% packet loss
round-trip min/avg/max = 0.046/0.046/0.046 ms
```

3. How to delete a namespace?

```
# ip netns delete ns1
# ip netns delete ns2
```

Control Groups

Control Groups (cgroups) are features of the Linux kernel which limit, account, and isolate resource usage of the following resources to a process group:-

CPU (cpu/cpuset)

- controls the time period (microseconds per second) a group should have CPU access.
- controls the upper limit of CPU time per second for the group.
- asigns a proportional value of the relative CPU time for a group.
- assigns cores to the group.
- controls the memory nodes, a group can access.

Memory (memory)

- controls the maximum memory limit to a group.
- controls memory swappiness, OOM, etc.

Disk I/O (blkio)

- assigns proportional value of block I/O to a group.
- sets per device hard limits on block I/O access.

Network (net_cls/net_prio)

- tags network packets with a class ID.
- uses `tc` to prioritize tagged packets.
- assigns per interface weighted proportional priority on egress traffic.

Devices

- controls the device access to groups.

Hugepages

- controls huge pages size usage.
- manages per cgroups huge pages matrices.

There is a special kind of `cgroup` called `Freezer`, which suspends/resumes group tasks. We will see an example of this.

1. How to list the available `cgroups` ?

Install the `cgroup-tools` tool

```
$ apt install cgroup-tools -y
```

List the `cgroups` with `lscgroup` command.

```
$ lscgroup
```

2. How to list the `cgroups` associated with a process ?

```
$ cat /proc/<pid>/cgroup

$ cat /proc/1/cgroup
11:cpuset:/
10:freezer:/
9:devices:/init.scope
8:pids:/init.scope
7:blkio:/init.scope
6:cpu,cpuacct:/init.scope
5:net_cls,net_prio:/
4:perf_event:/
3:hugetlb:/
2:memory:/init.scope
1:name=systemd:/init.scope
```

3. How to freeze a process using `cgroups` and then `defreeze` it ?

In this lab, we will use the `freezer` cgroup to freeze and defreeze a process. Once a process is frozen, we cannot do any operation on it. We need to defreeze it to make the process accessible again.

Create a new `cgroup` hierarchy under `/sys/fs/cgroup/freezer/`.

```
$ cd /sys/fs/cgroup/freezer
$ mkdir mycgroup
$ cd mycgroup/
$ ls
cgroup.clone_children cgroup.procs freezer.parent_freezing freezer.self_freezing freezer.state notify_on_
```

As soon as the directory was created under the `freezer` cgroup, some files have been populated automatically by the cgroup sub-system. `tasks` file contains the pids of the processes, which will get affected by this cgroup.

```
$ pwd
/sys/fs/cgroup/freezer/mycgroup
$ cat tasks
```

As expected, the `tasks` file is currently empty, as we have attached the process to our newly created cgroup.

Create a new process and attach it to the `cgroup` created in the previous step.

Open a new terminal and get its PID.

```
$ ps
  PID TTY          TIME CMD
 6664 pts/1    00:00:00 bash
 6693 pts/1    00:00:00 ps
```

Come back to the previous terminal and, in the `tasks` file, append the PID which we got in previous step.

```
$ pwd
/sys/fs/cgroup/freezer/mycgroup
$ echo 6664 >> tasks
```

Freeze all the processes which are attached to the `cgroup` we created earlier.

Run the following command to freeze all processes.

```
$ pwd
/sys/fs/cgroup/freezer/mycgroup
$ echo FROZEN > freezer.state
```

Go back to the other terminal and try to write something on the screen. Nothing comes on the screen, as it is frozen.

Defreeze all the processes which are frozen in the previous step.

Run the following command to defreeze all processes on the terminal from which we froze all the processes.

```
$ pwd
/sys/fs/cgroup/freezer/mycgroup
$ echo THAWED > freezer.state
```

Now, go back to the other terminal and you would see that whatever we typed earlier on the frozen terminal is shown, because the processes are thawed now.

```
$ date
Tue Sep 27 06:51:12 UTC 2016
$ asdf
No command 'asdf' found, did you mean:
 Command 'sadf' from package 'sysstat' (main)
 Command 'sdf' from package 'sdf' (universe)
 Command 'asdfg' from package 'aoeui' (universe)
asdf: command not found
```

UnionFS

UnionFS transparently overlays files and directories of separate filesystems, to create a coherent filesystem. Each one of the participant directories is referred to as a branch. We can set the priority while mounting branches, mount them read-only, etc.

In the following diagram, we can see that `dir1` has two files, `f1` and `f2`. `dir2` also has two files,‘f3* and *f4*. After mounting them using *FUSE* implementation on *UnionFS*, we can see all 4 files on the *union* directory.

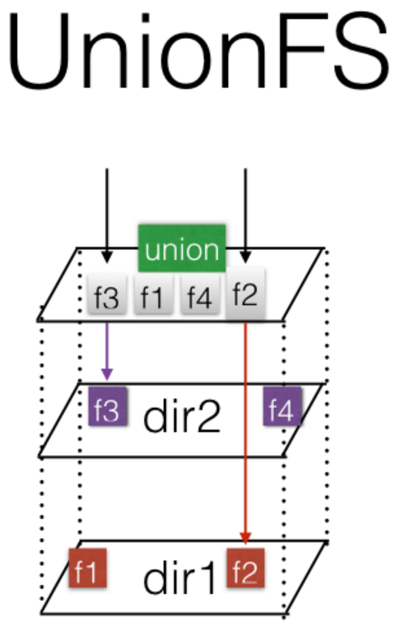


Figure 5: images

1. How to transparently overlay two directories, one on top of another, using UnionFS ?

Install the `unionfs-fuse` package.

```
$ apt install unionfs-fuse
```

Create a `dir1` directory, and then, create two files `f1` and `f2` inside that directory.

```
$ mkdir /root/dir1
$ touch /root/dir1/f1
$ touch /root/dir1/f2
```

Create a`dir2` directory, and then, create two files`f3` and `f4` inside that directory.

```
$ mkdir /root/dir2
$ touch /root/dir2/f3
$ touch /root/dir2/f4
```

Create a directory called `union`.

```
$ mkdir /root/union
```

Mount `dir1` and `dir2` to the `union` directory, using `unionfs-fuse`, and then, list the files.

```
$ unionfs /root/dir1:/root/dir2  /root/union/
$ ls /root/union/
f1  f2  f3  f4
```

References

- https://lwn.net/Articles/531114/
- https://en.wikipedia.org/wiki/Cgroups
- http://www.opencloudblog.com/?p=66
- http://unionfs.filesystems.org/