Linux user namespaces:
a blessing and a curse

Ignat Korchagin
@ignatkn

CLOUDFLARE

$ whoami

- Linux team at Cloudflare
- Systems security and performance
- Low-level programming

CLOUDFLARE

Agenda

- How containers contain: Linux namespaces
- Unprivileged user namespaces
- Linux application sandboxing (Google Chrome example)
- Linux bugs under Linux namespaces
- Policing user namespaces
- Fine grained user namespace creation

@ignatkn

CLOUDFLARE

How containers contain

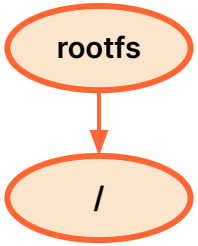
Linux namespaces

CLOUDFLARE

Mount tree

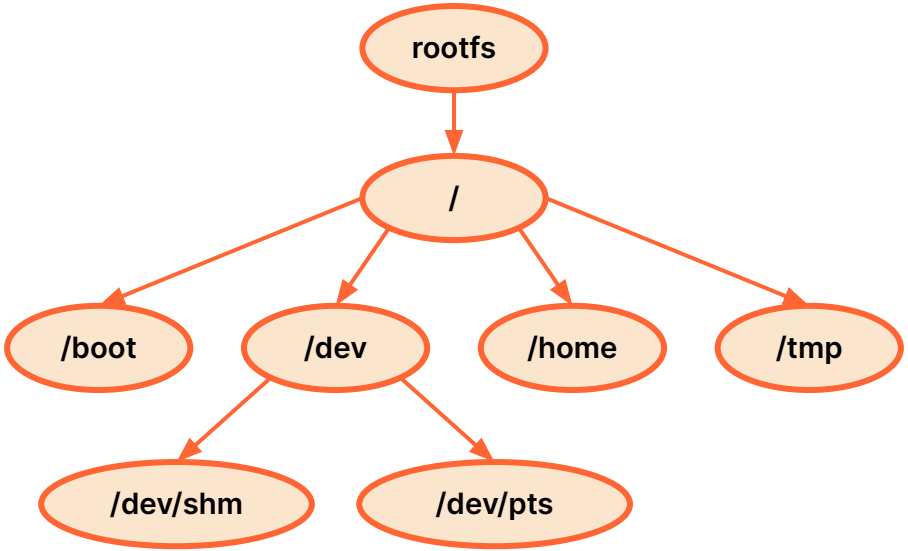
CLOUDFLARE

Mount tree



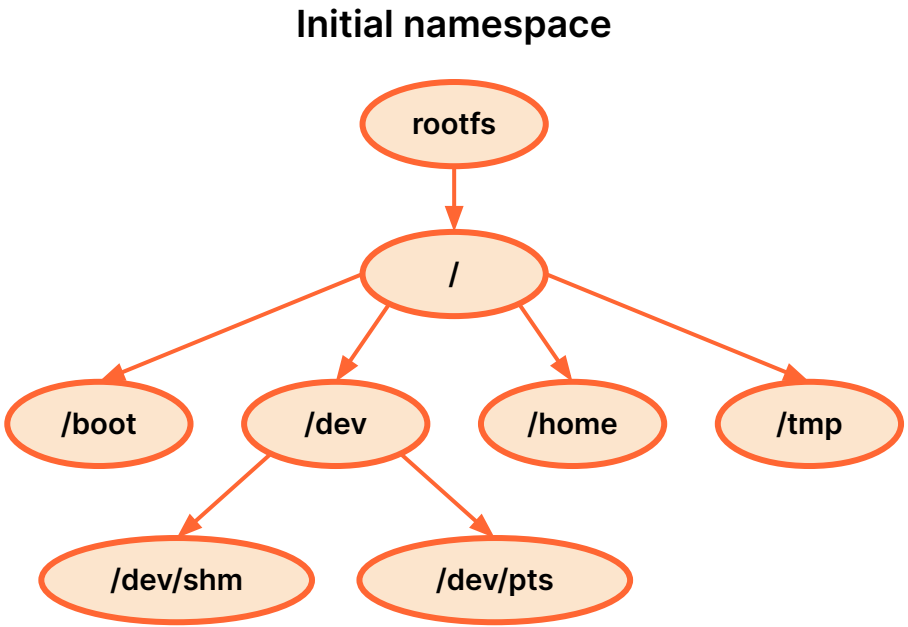
CLOUDFLARE

Mount tree



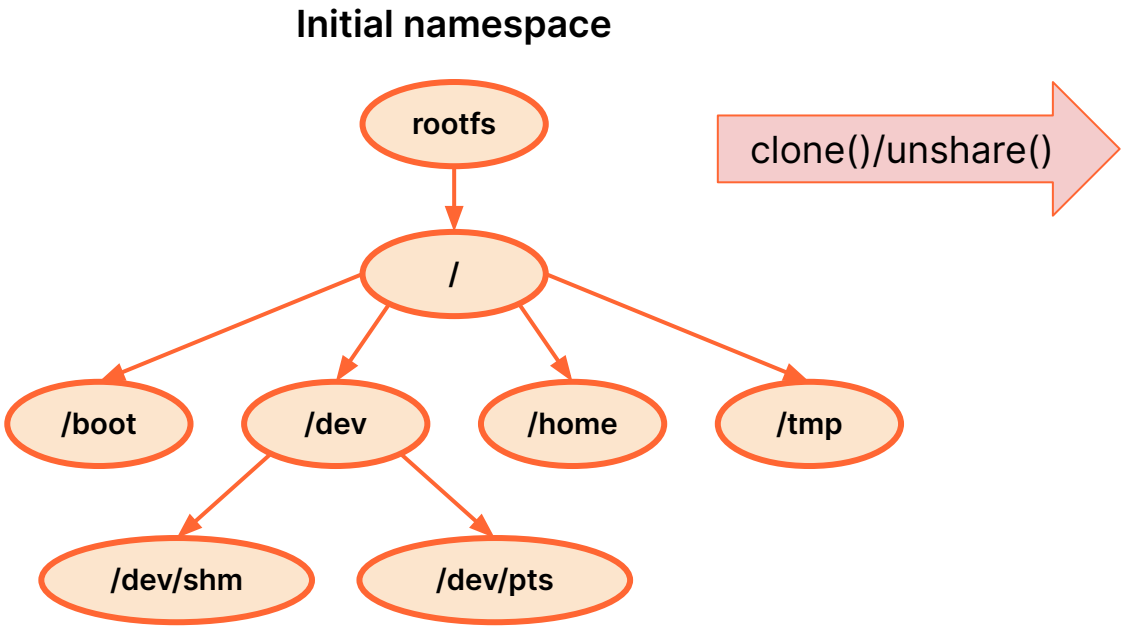
CLOUDFLARE

Mount namespaces



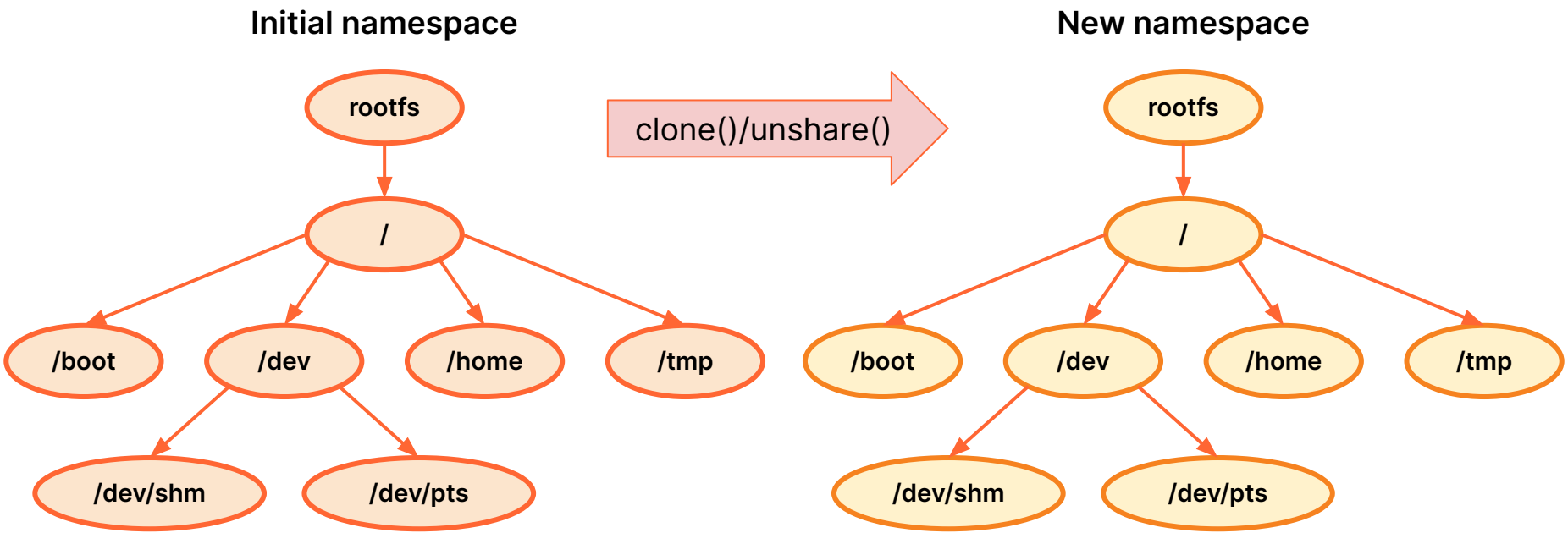
CLOUDFLARE

Mount namespaces



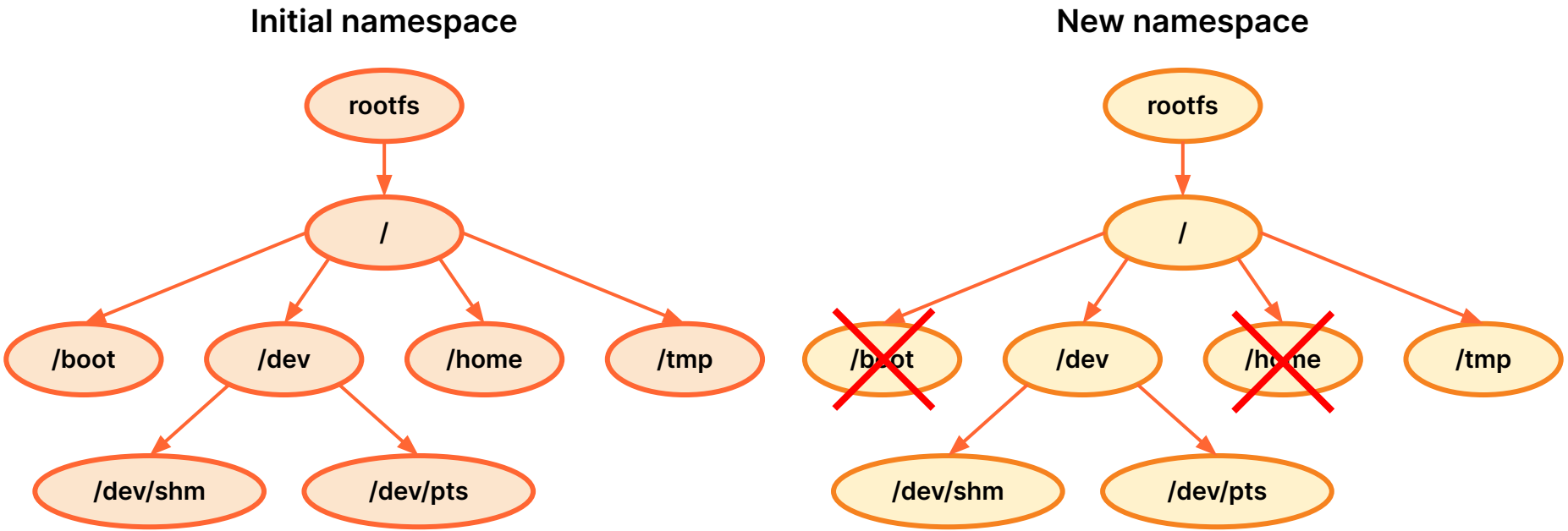
CLOUDFLARE

Mount namespaces



CLOUDFLARE

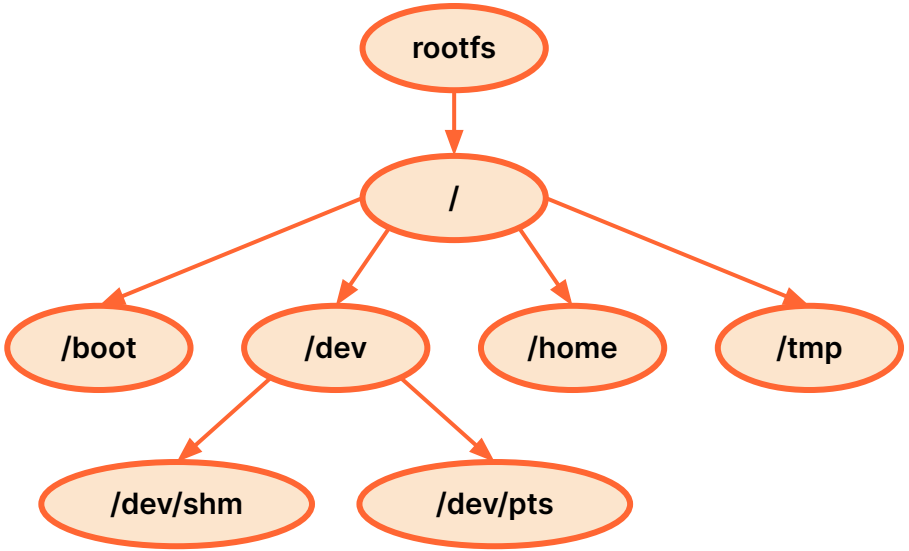
Mount namespaces



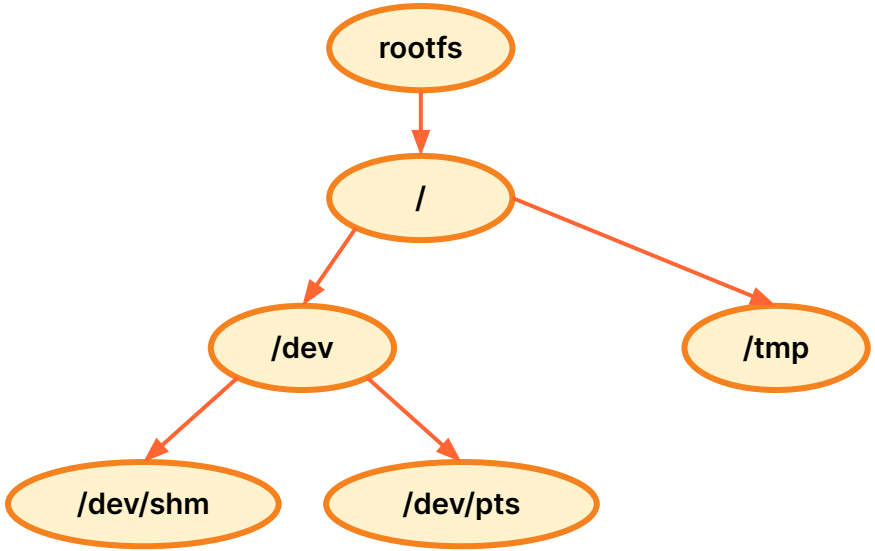
CLOUDFLARE

Mount namespaces

Initial namespace

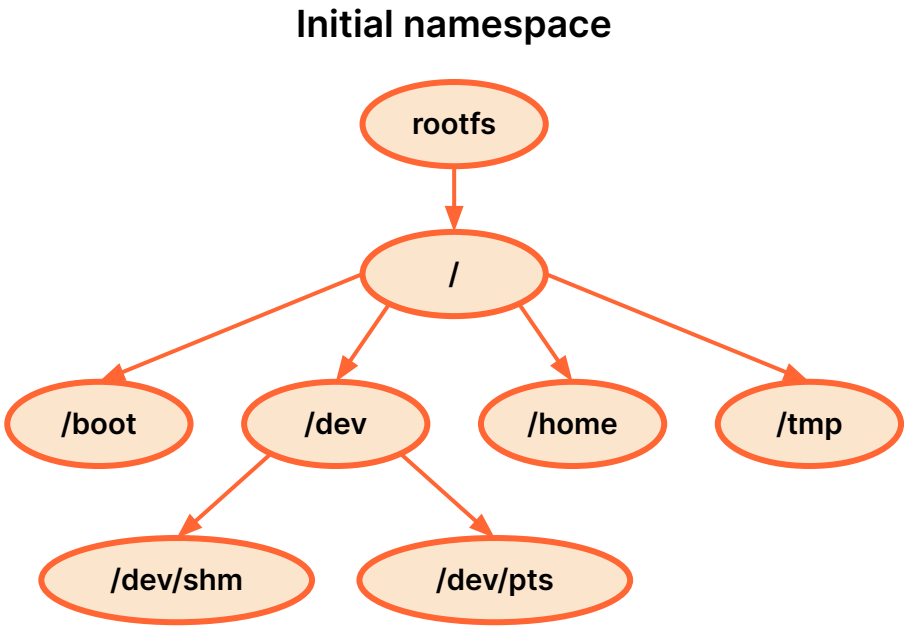


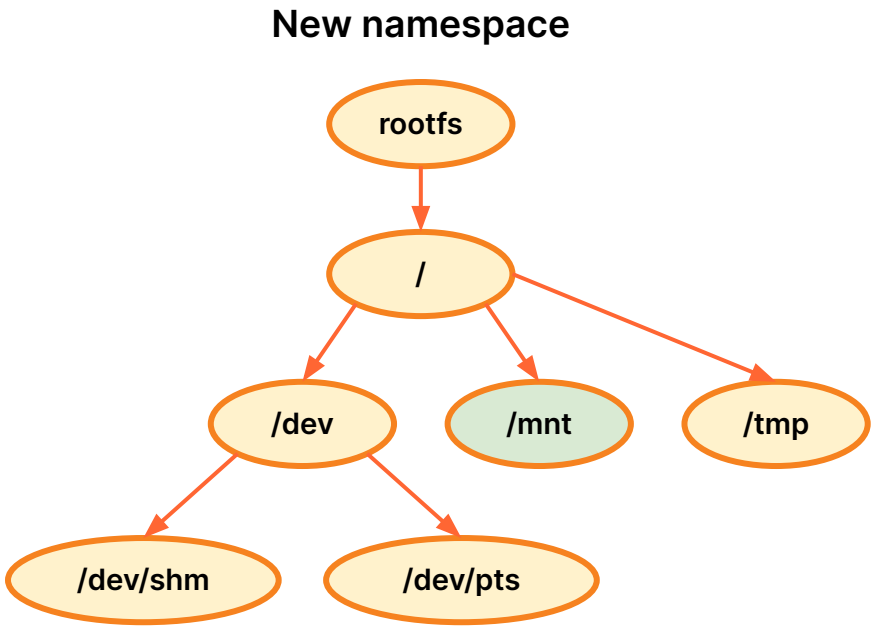
New namespace



CLOUDFLARE

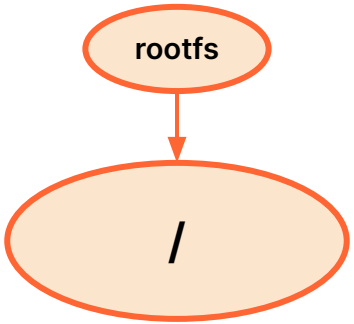
Mount namespaces





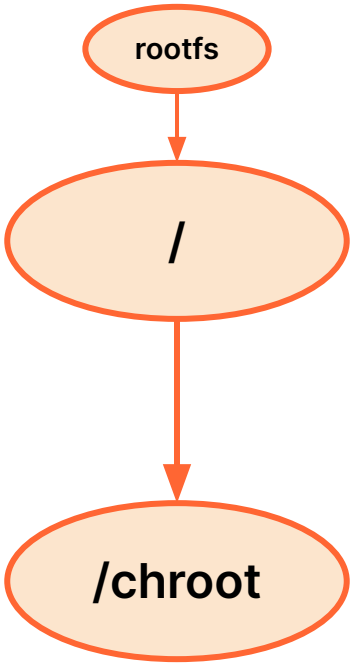
CLOUDFLARE

Traditional chroot



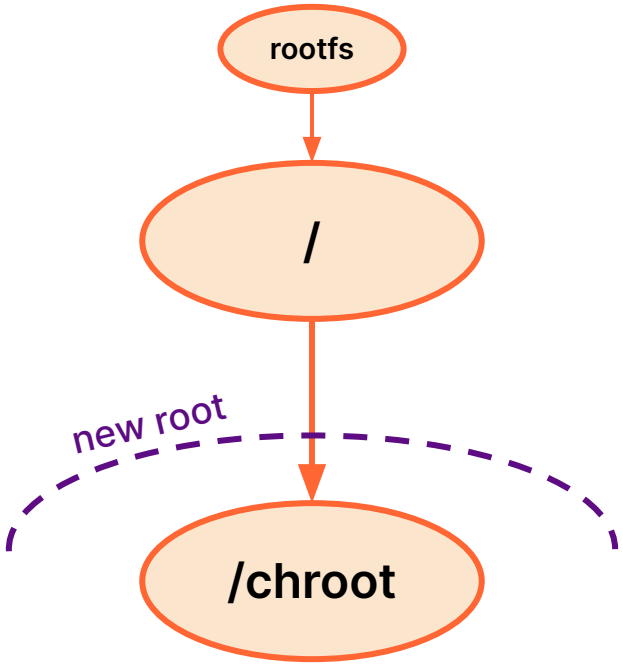
CLOUDFLARE

Traditional chroot



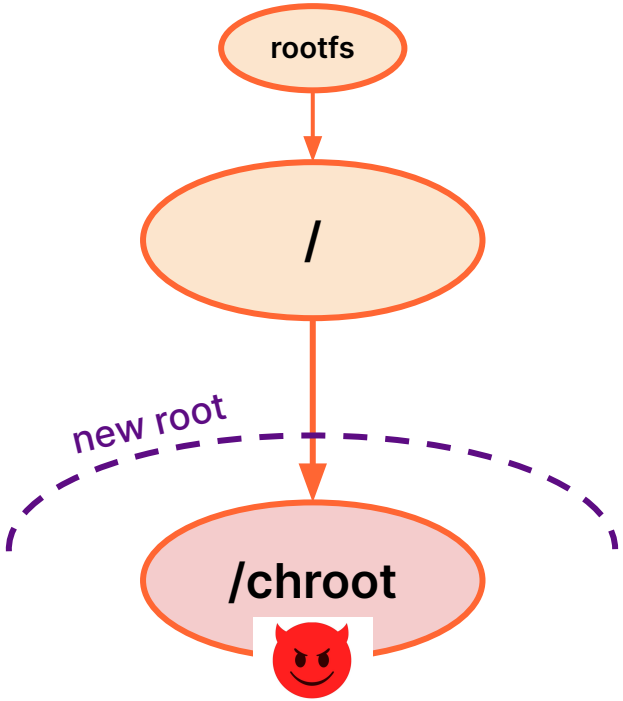
CLOUDFLARE

Traditional chroot



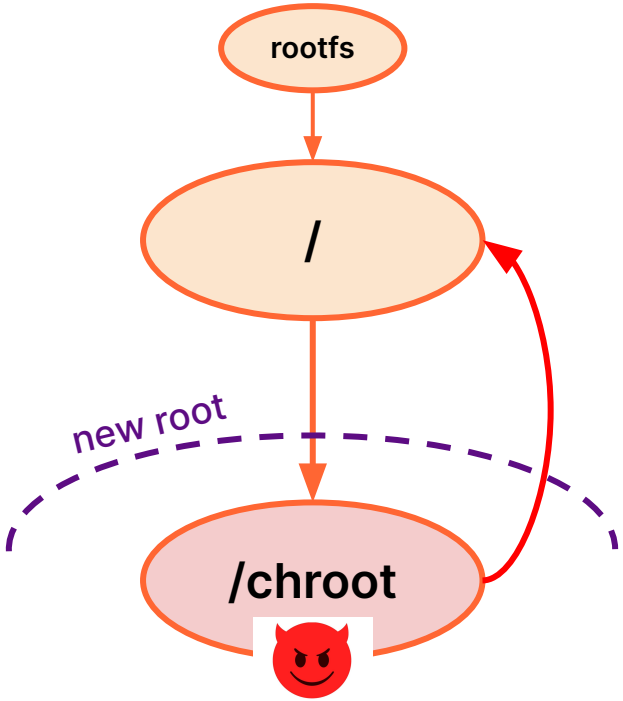
CLOUDFLARE

Traditional chroot



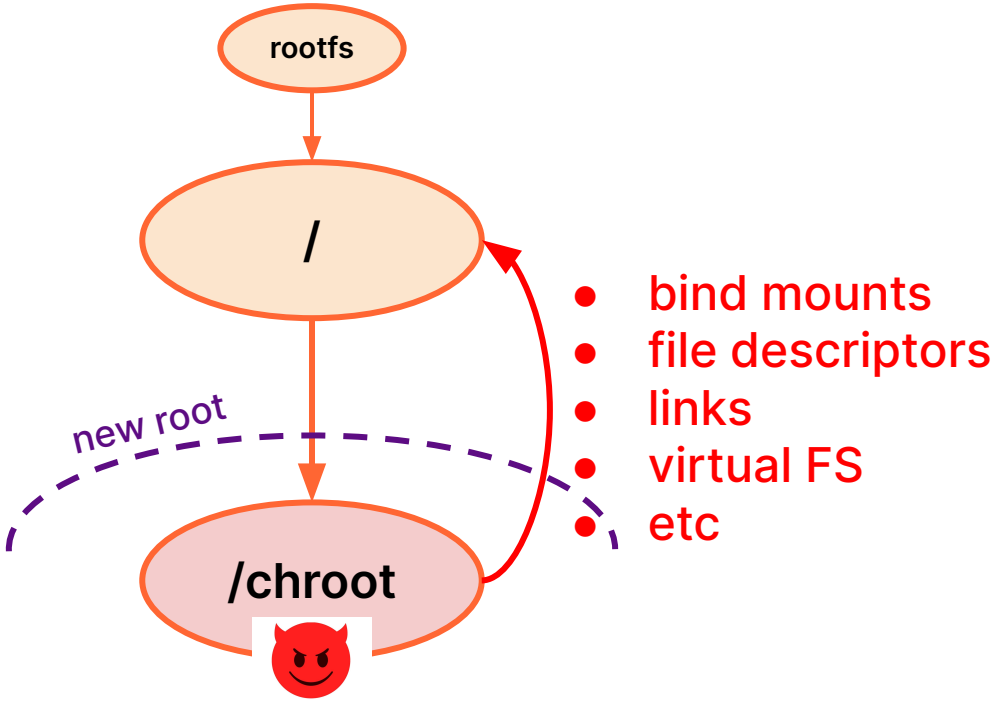
CLOUDFLARE

Traditional chroot



CLOUDFLARE

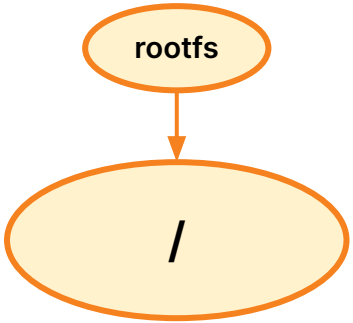
Traditional chroot



- bind mounts
- file descriptors
- links
- virtual FS
- etc

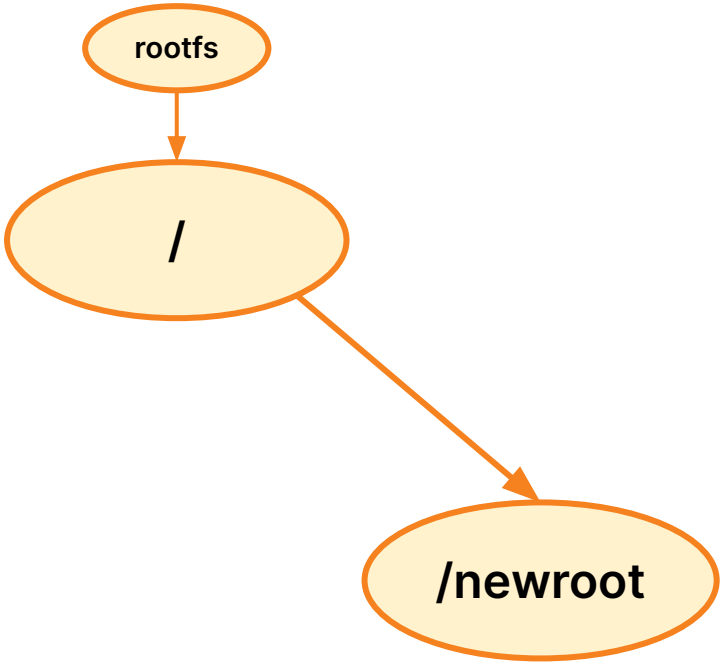
CLOUDFLARE

Mount namespaces and pivot_root()



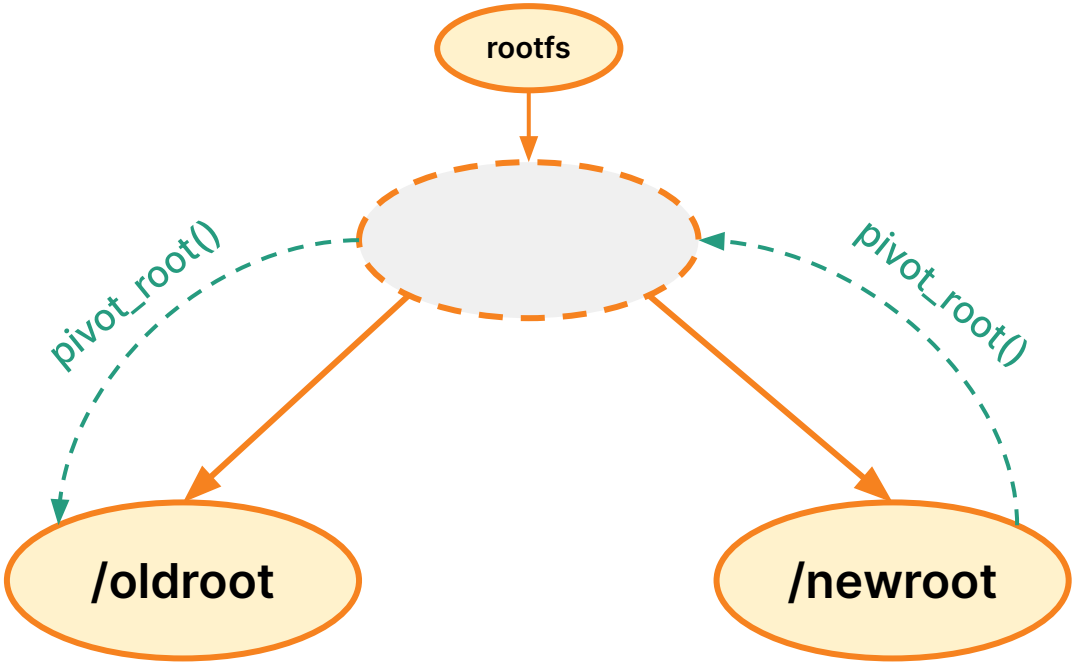
CLOUDFLARE

Mount namespaces and pivot_root()



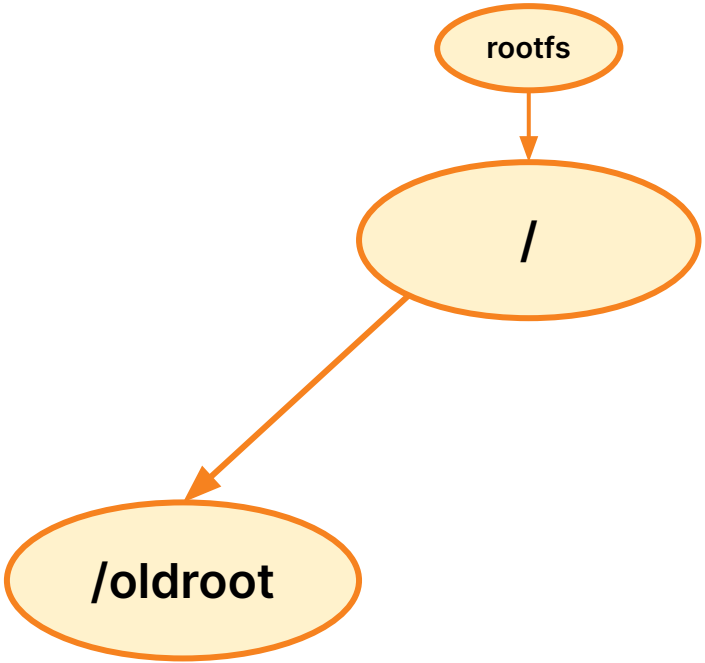
CLOUDFLARE

Mount namespaces and pivot_root()



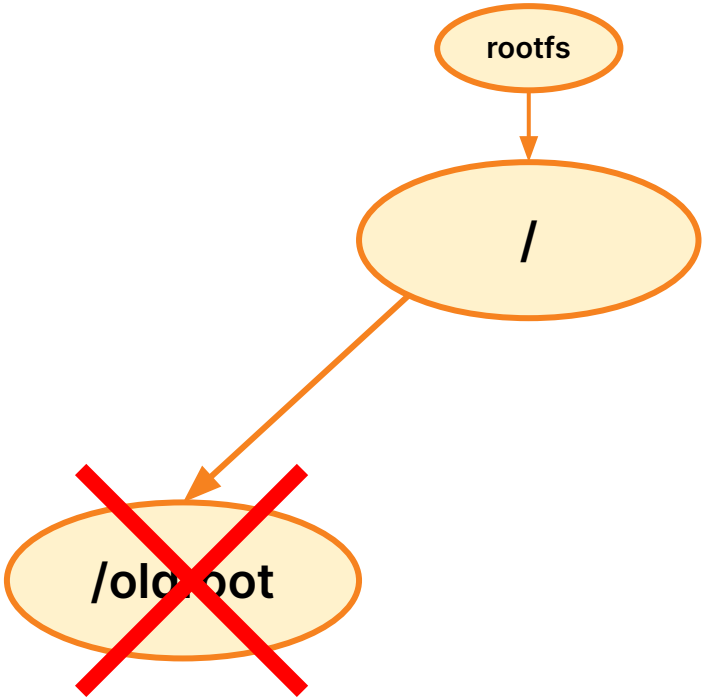
CLOUDFLARE

Mount namespaces and pivot_root()



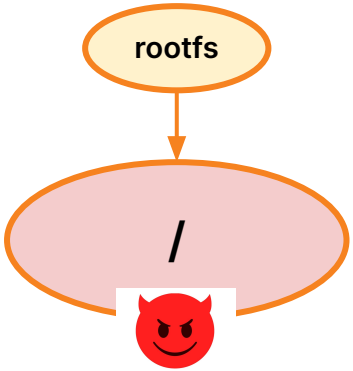
CLOUDFLARE

Mount namespaces and pivot_root()



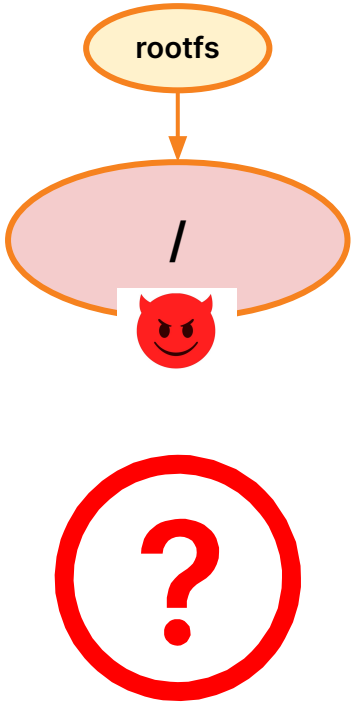
CLOUDFLARE

Mount namespaces and pivot_root()



CLOUDFLARE

Mount namespaces and pivot_root()



CLOUDFLARE

Mount namespaces example

```
ignat@dev:~$
```

CLOUDFLARE

Mount namespaces example

```
ignat@dev:~$ mount | grep ^tmpfs
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
tmpfs on /run type tmpfs
(rw,nosuid,nodev,size=13162688k,nr_inodes=819200,mode=755,inode64)
tmpfs on /run/lock type tmpfs
(rw,nosuid,nodev,noexec,relatime,size=5120k,inode64)
tmpfs on /run/user/1000 type tmpfs
(rw,nosuid,nodev,relatime,size=6581340k,nr_inodes=1645335,mode=700,uid=1000,gid
=1000,inode64)
ignat@dev:~$
```

CLOUDFLARE

Mount namespaces example

```
ignat@dev:~$ mount | grep ^tmpfs
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
tmpfs on /run type tmpfs
(rw,nosuid,nodev,size=13162688k,nr_inodes=819200,mode=755,inode64)
tmpfs on /run/lock type tmpfs
(rw,nosuid,nodev,noexec,relatime,size=5120k,inode64)
tmpfs on /run/user/1000 type tmpfs
(rw,nosuid,nodev,relatime,size=6581340k,nr_inodes=1645335,mode=700,uid=1000,gid
=1000,inode64)
ignat@dev:~$ sudo unshare --mount
root@dev:/home/ignat#
```

CLOUDFLARE

Mount namespaces example

```
ignat@dev:~$ mount | grep ^tmpfs
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
tmpfs on /run type tmpfs
(rw,nosuid,nodev,size=13162688k,nr_inodes=819200,mode=755,inode64)
tmpfs on /run/lock type tmpfs
(rw,nosuid,nodev,noexec,relatime,size=5120k,inode64)
tmpfs on /run/user/1000 type tmpfs
(rw,nosuid,nodev,relatime,size=6581340k,nr_inodes=1645335,mode=700,uid=1000,gid
=1000,inode64)
ignat@dev:~$ sudo unshare --mount
root@dev:/home/ignat# umount /run/user/1000
root@dev:/home/ignat#
```

CLOUDFLARE

Mount namespaces example

```
ignat@dev:~$ mount | grep ^tmpfs
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
tmpfs on /run type tmpfs
(rw,nosuid,nodev,size=13162688k,nr_inodes=819200,mode=755,inode64)
tmpfs on /run/lock type tmpfs
(rw,nosuid,nodev,noexec,relatime,size=5120k,inode64)
tmpfs on /run/user/1000 type tmpfs
(rw,nosuid,nodev,relatime,size=6581340k,nr_inodes=1645335,mode=700,uid=1000,gid
=1000,inode64)
ignat@dev:~$ sudo unshare --mount
root@dev:/home/ignat# umount /run/user/1000
root@dev:/home/ignat# mountpoint /run/user/1000
/run/user/1000 is not a mountpoint
root@dev:/home/ignat#
```

CLOUDFLARE

Mount namespaces example

```
ignat@dev:~$ mount | grep ^tmpfs
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
tmpfs on /run type tmpfs
(rw,nosuid,nodev,size=13162688k,nr_inodes=819200,mode=755,inode64)
tmpfs on /run/lock type tmpfs
(rw,nosuid,nodev,noexec,relatime,size=5120k,inode64)
tmpfs on /run/user/1000 type tmpfs
(rw,nosuid,nodev,relatime,size=6581340k,nr_inodes=1645335,mode=700,uid=1000,gid
=1000,inode64)
ignat@dev:~$ sudo unshare --mount
root@dev:/home/ignat# umount /run/user/1000
root@dev:/home/ignat# mountpoint /run/user/1000
/run/user/1000 is not a mountpoint
root@dev:/home/ignat# exit
logout
ignat@dev:~$
```

CLOUDFLARE

Mount namespaces example

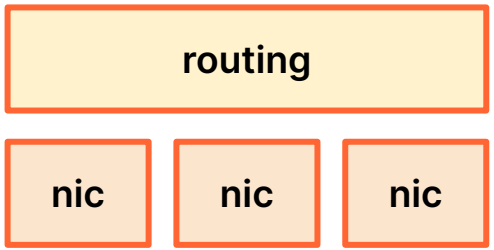
```
ignat@dev:~$ mount | grep ^tmpfs
tmpfs on /dev/shm type tmpfs (rw,nosuid,nodev,inode64)
tmpfs on /run type tmpfs
(rw,nosuid,nodev,size=13162688k,nr_inodes=819200,mode=755,inode64)
tmpfs on /run/lock type tmpfs
(rw,nosuid,nodev,noexec,relatime,size=5120k,inode64)
tmpfs on /run/user/1000 type tmpfs
(rw,nosuid,nodev,relatime,size=6581340k,nr_inodes=1645335,mode=700,uid=1000,gid
=1000,inode64)
ignat@dev:~$ sudo unshare --mount
root@dev:/home/ignat# umount /run/user/1000
root@dev:/home/ignat# mountpoint /run/user/1000
/run/user/1000 is not a mountpoint
root@dev:/home/ignat# exit
logout
ignat@dev:~$ mountpoint /run/user/1000
/run/user/1000 is a mountpoint
```

CLOUDFLARE

Network resources

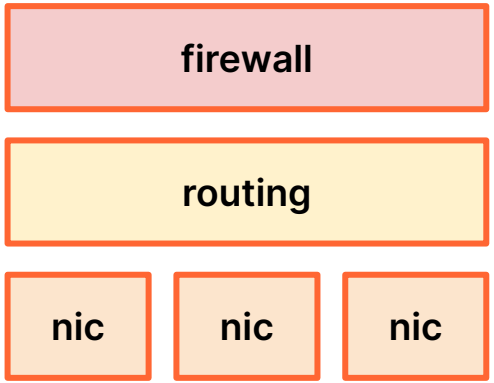
CLOUDFLARE

Network resources



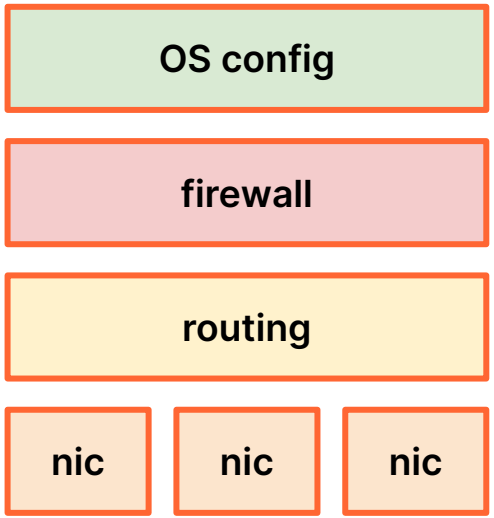
CLOUDFLARE

Network resources



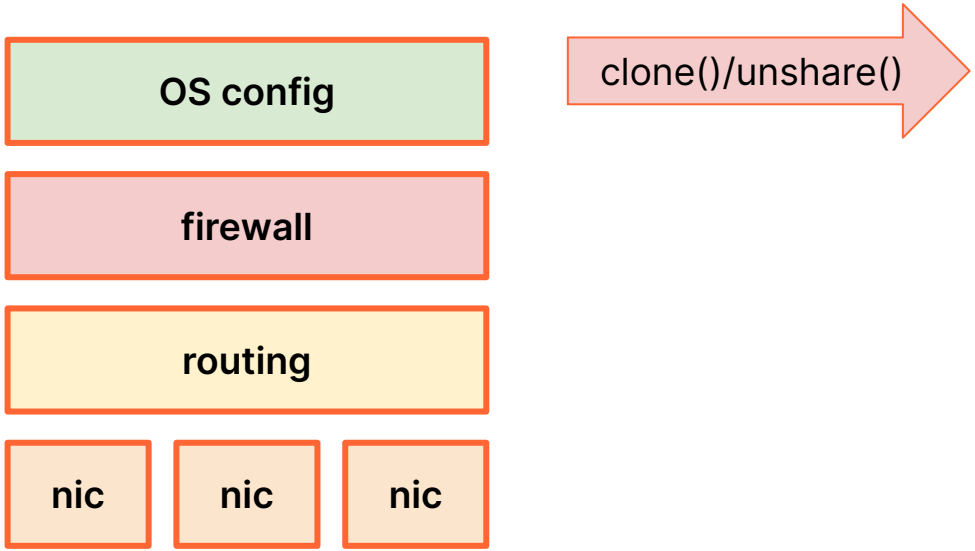
CLOUDFLARE

Network resources



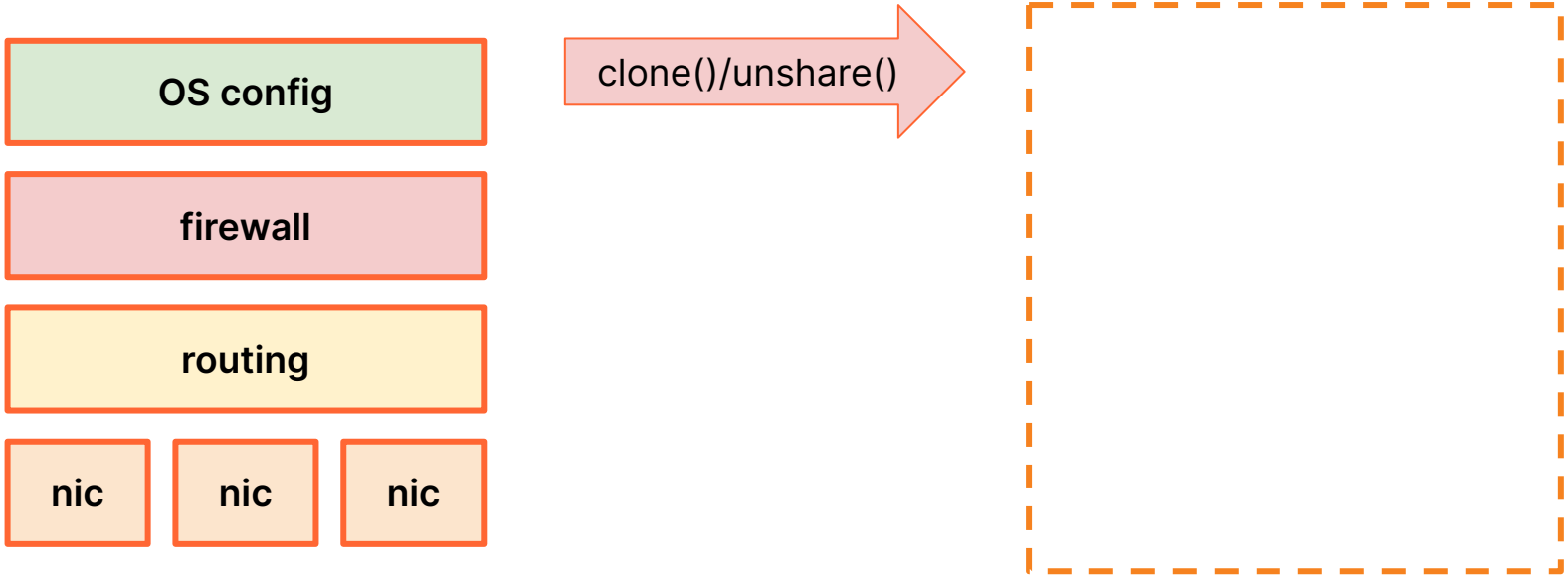
CLOUDFLARE

Network namespaces



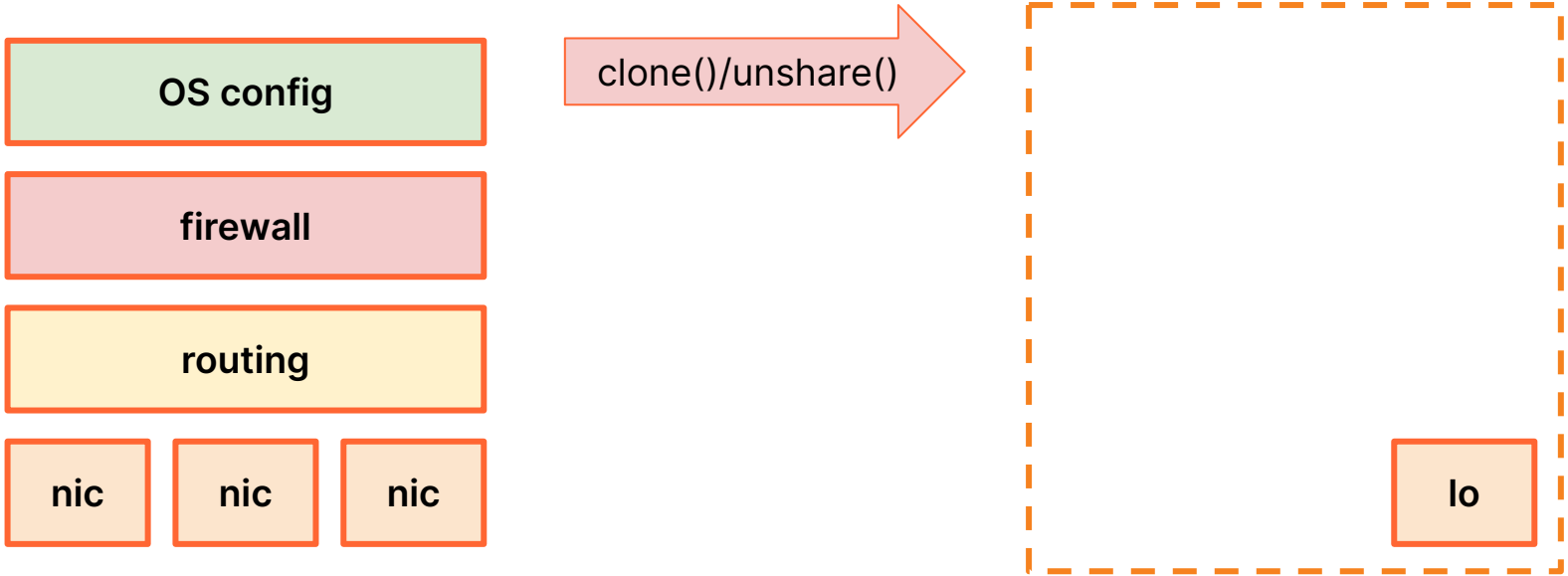
CLOUDFLARE

Network namespaces



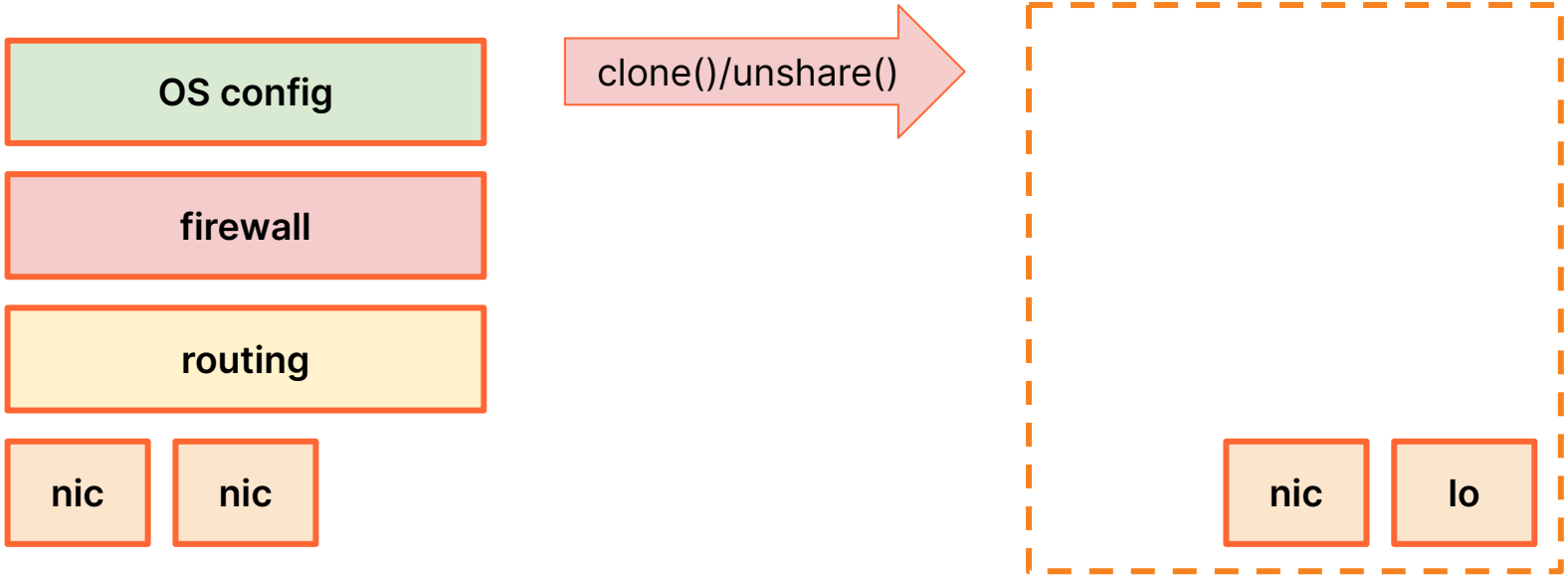
CLOUDFLARE

Network namespaces



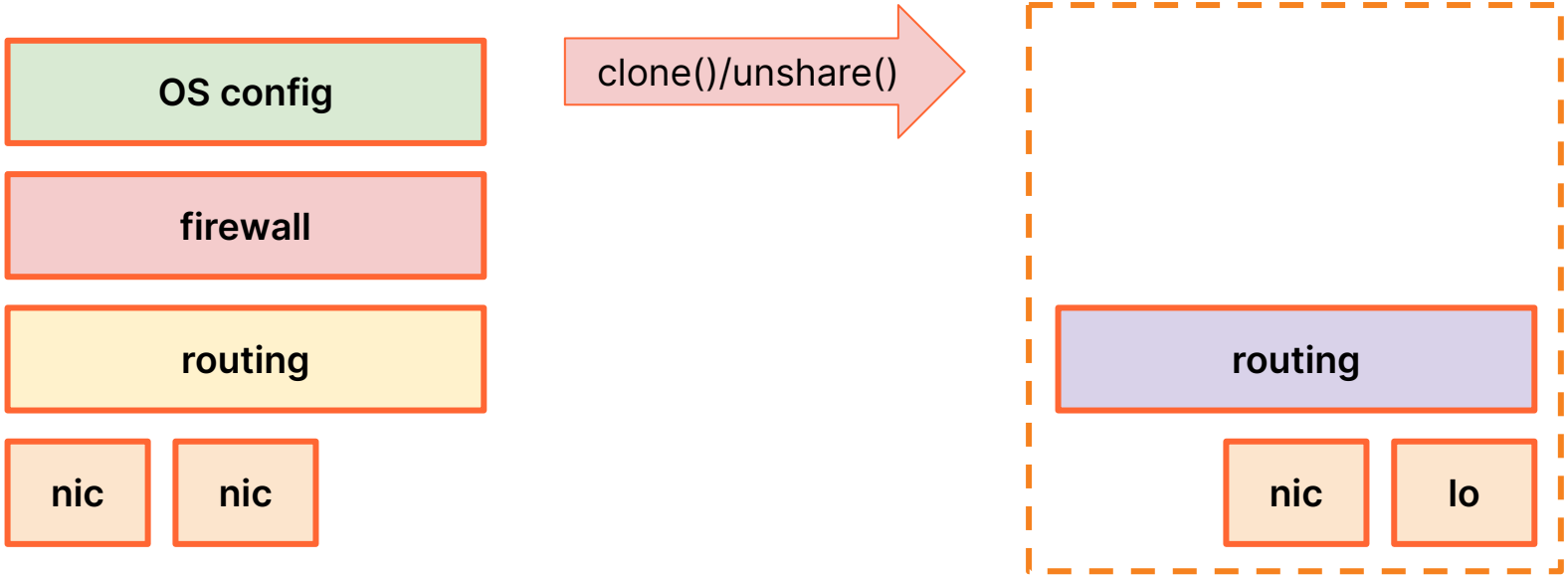
CLOUDFLARE

Network namespaces



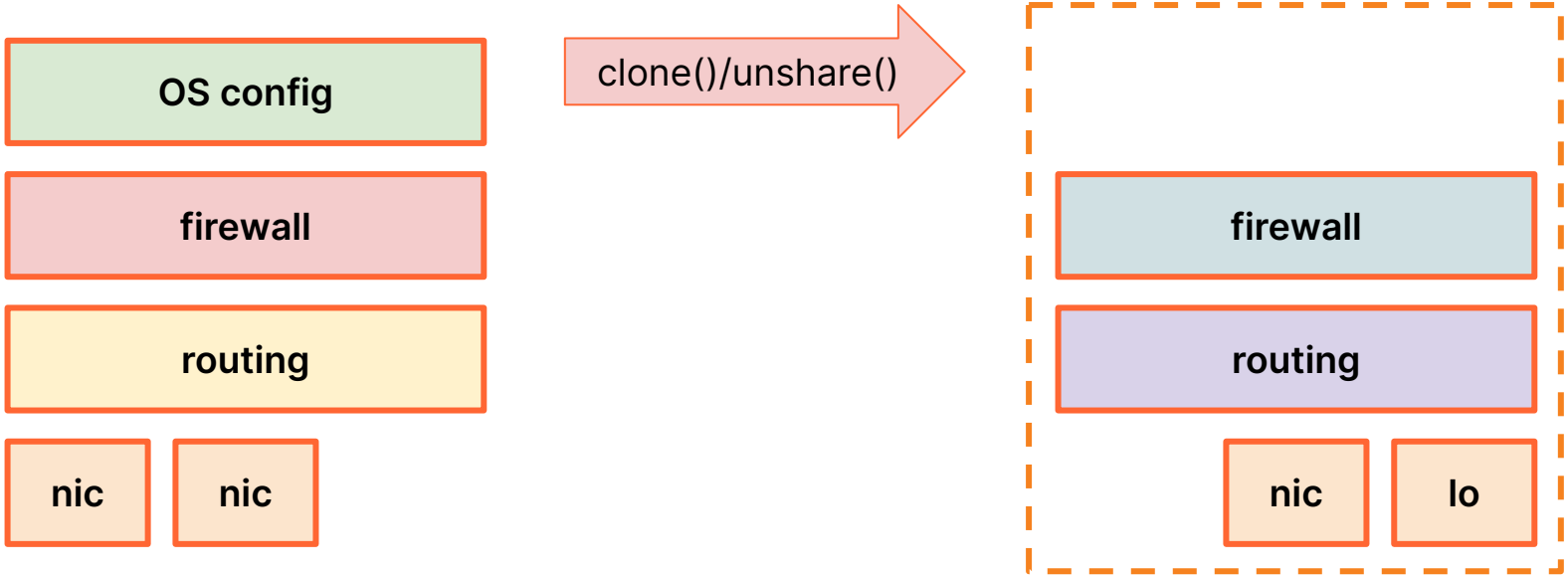
CLOUDFLARE

Network namespaces



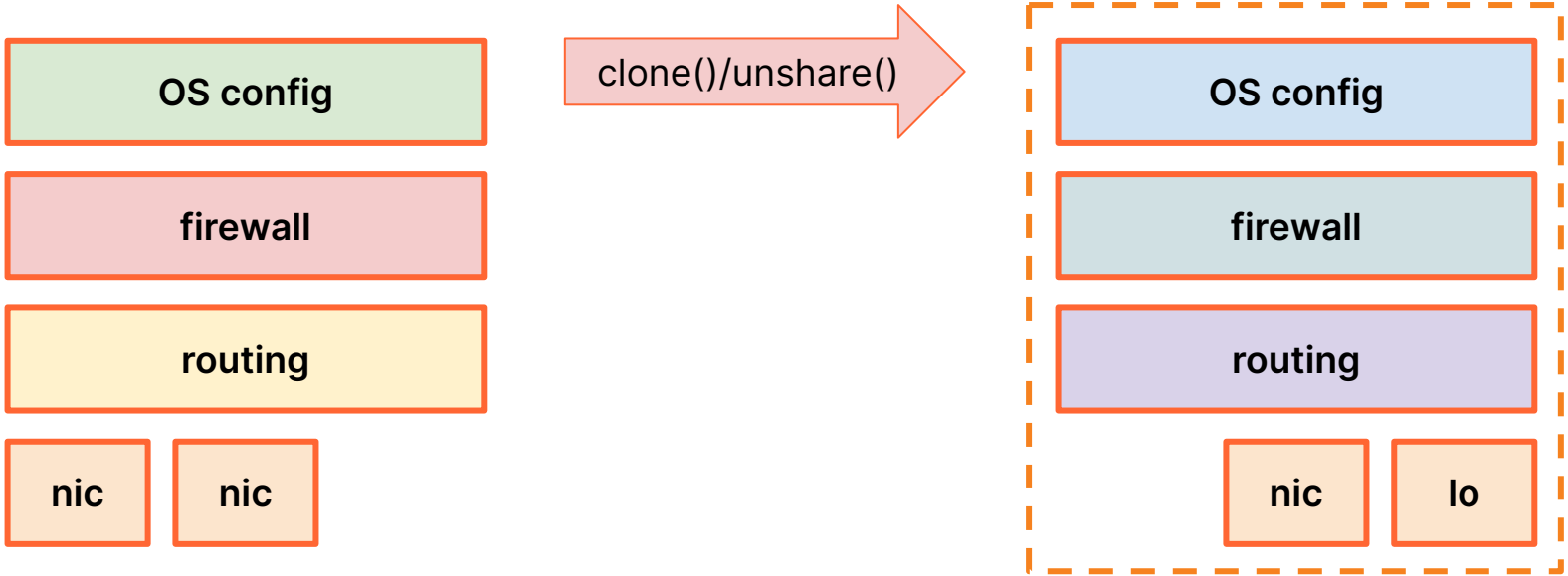
CLOUDFLARE

Network namespaces



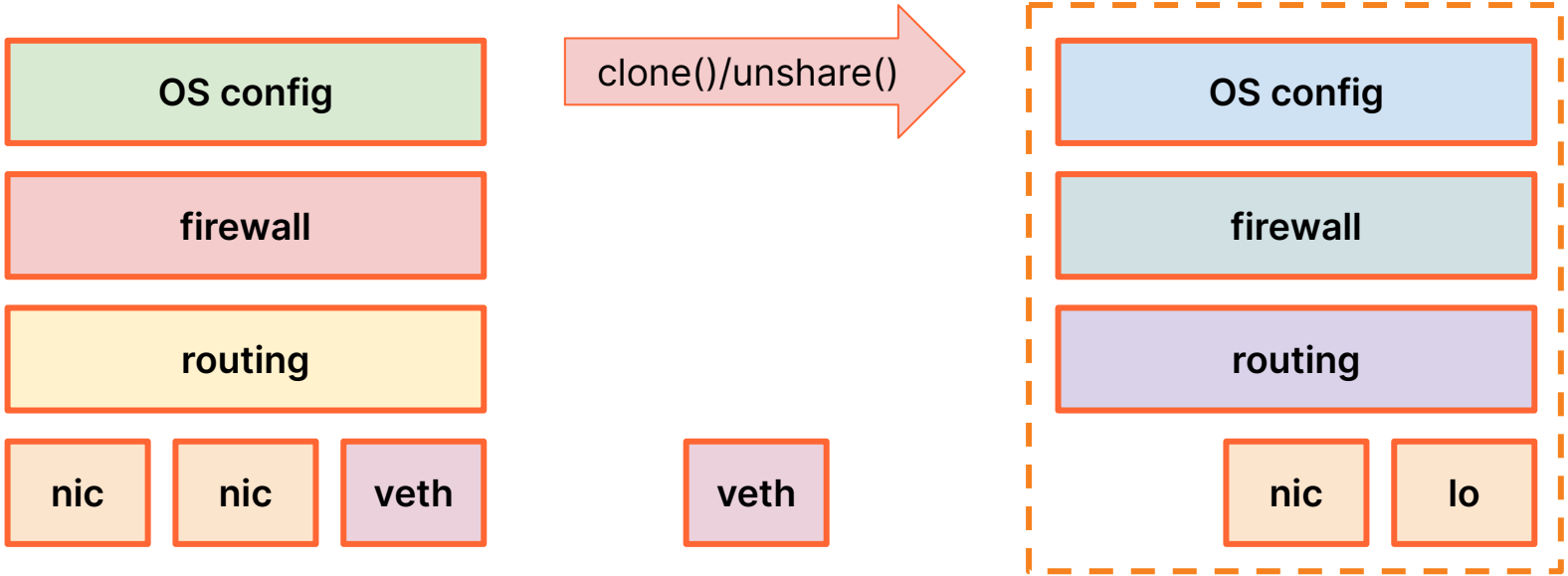
CLOUDFLARE

Network namespaces



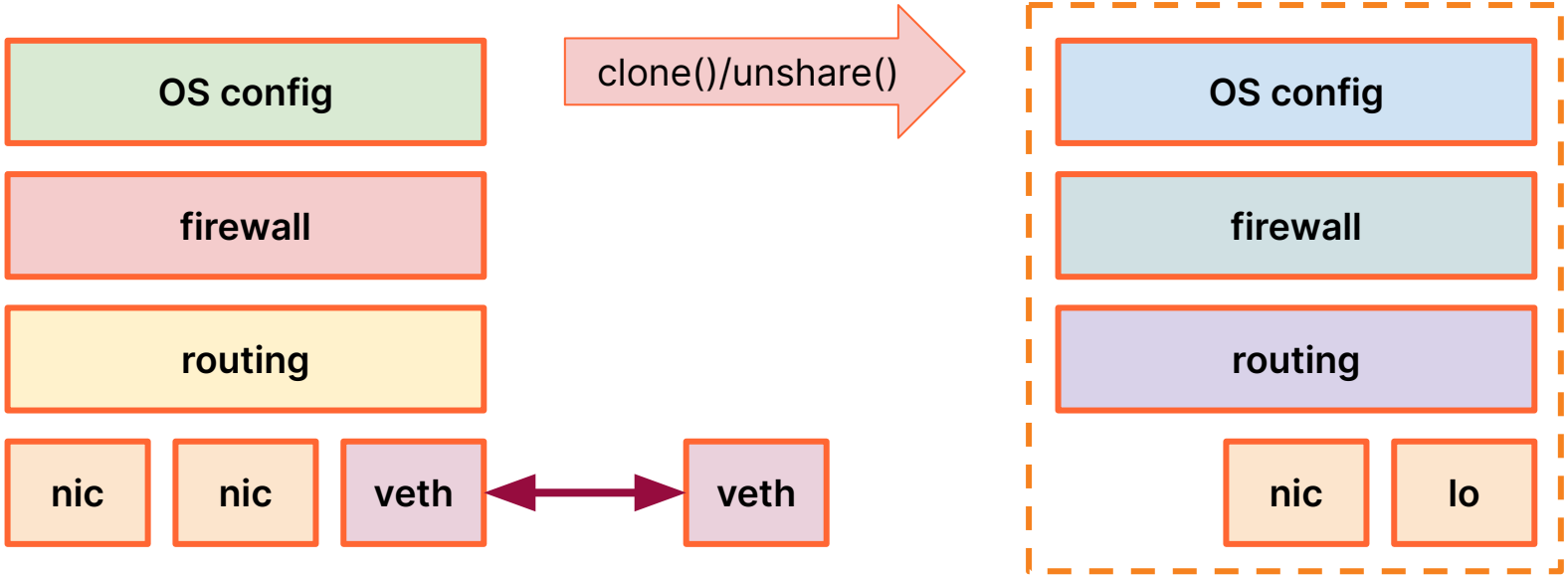
CLOUDFLARE

Network namespaces



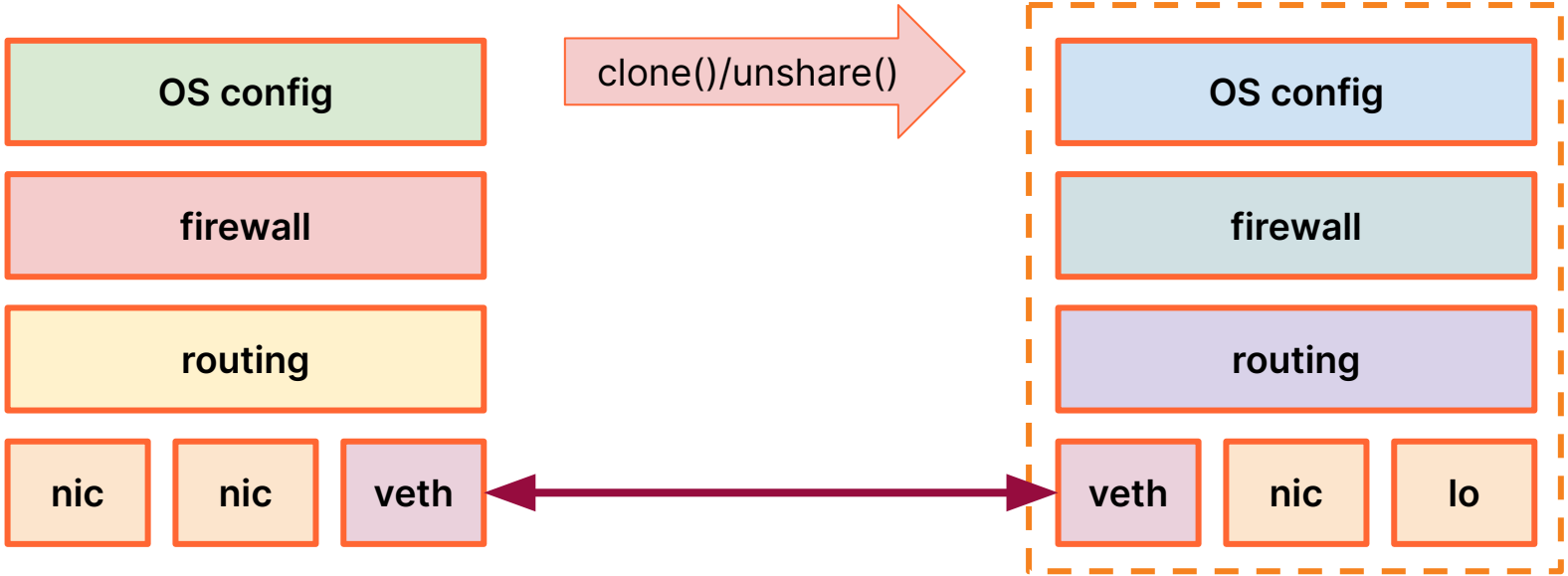
CLOUDFLARE

Network namespaces



CLOUDFLARE

Network namespaces



CLOUDFLARE

Network namespaces example

```
ignat@dev:~$
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
DEFAULT group default qlen 1000
    link/ether 52:54:00:e6:15:a5 brd ff:ff:ff:ff:ff:ff
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode
DEFAULT group default
    link/ether 02:42:ec:9d:a8:8f brd ff:ff:ff:ff:ff:ff
ignat@dev:~$
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
DEFAULT group default qlen 1000
    link/ether 52:54:00:e6:15:a5 brd ff:ff:ff:ff:ff:ff
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode
DEFAULT group default
    link/ether 02:42:ec:9d:a8:8f brd ff:ff:ff:ff:ff:ff
ignat@dev:~$ sudo unshare --net
root@dev:/home/ignat#
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
DEFAULT group default qlen 1000
    link/ether 52:54:00:e6:15:a5 brd ff:ff:ff:ff:ff:ff
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode
DEFAULT group default
    link/ether 02:42:ec:9d:a8:8f brd ff:ff:ff:ff:ff:ff
ignat@dev:~$ sudo unshare --net
root@dev:/home/ignat# ip link
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
root@dev:/home/ignat#
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
DEFAULT group default qlen 1000
    link/ether 52:54:00:e6:15:a5 brd ff:ff:ff:ff:ff:ff
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode
DEFAULT group default
    link/ether 02:42:ec:9d:a8:8f brd ff:ff:ff:ff:ff:ff
ignat@dev:~$ sudo unshare --net
root@dev:/home/ignat# ip link
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
root@dev:/home/ignat# ip link set lo up
root@dev:/home/ignat#
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$ ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp1s0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode
DEFAULT group default qlen 1000
    link/ether 52:54:00:e6:15:a5 brd ff:ff:ff:ff:ff:ff
3: docker0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc noqueue state DOWN mode
DEFAULT group default
    link/ether 02:42:ec:9d:a8:8f brd ff:ff:ff:ff:ff:ff
ignat@dev:~$ sudo unshare --net
root@dev:/home/ignat# ip link
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
root@dev:/home/ignat# ip link set lo up
root@dev:/home/ignat# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$
```

```
root@dev:/home/ignat#
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$
```

```
root@dev:/home/ignat# ip link add right
type veth peer left netns 1
root@dev:/home/ignat#
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$
```

```
root@dev:/home/ignat# ip link add right
type veth peer left netns 1
root@dev:/home/ignat# ip link show right
2: right@if5: <BROADCAST,MULTICAST> mtu
1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/ether 06:45:05:83:b7:e8 brd
ff:ff:ff:ff:ff:ff link-netnsid 0
root@dev:/home/ignat#
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$ sudo ip link show left
5: left@if2: <BROADCAST,MULTICAST> mtu 1500
qdisc noop state DOWN mode DEFAULT group
default qlen 1000
ignat@dev:~$
```

```
root@dev:/home/ignat# ip link add right
type veth peer left netns 1
root@dev:/home/ignat# ip link show right
2: right@if5: <BROADCAST,MULTICAST> mtu
1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/ether 06:45:05:83:b7:e8 brd
ff:ff:ff:ff:ff:ff link-netnsid 0
root@dev:/home/ignat#
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$ sudo ip link show left
5: left@if2: <BROADCAST,MULTICAST> mtu 1500
qdisc noop state DOWN mode DEFAULT group
default qlen 1000
ignat@dev:~$
```

```
root@dev:/home/ignat# ip link add right
type veth peer left netns 1
root@dev:/home/ignat# ip link show right
2: right@if5: <BROADCAST,MULTICAST> mtu
1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/ether 06:45:05:83:b7:e8 brd
ff:ff:ff:ff:ff:ff link-netnsid 0
root@dev:/home/ignat# ip address add dev
right 192.168.1.7/24
root@dev:/home/ignat#
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$ sudo ip link show left
5: left@if2: <BROADCAST,MULTICAST> mtu 1500
qdisc noop state DOWN mode DEFAULT group
default qlen 1000
ignat@dev:~$
```

```
root@dev:/home/ignat# ip link add right
type veth peer left netns 1
root@dev:/home/ignat# ip link show right
2: right@if5: <BROADCAST,MULTICAST> mtu
1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/ether 06:45:05:83:b7:e8 brd
ff:ff:ff:ff:ff:ff link-netnsid 0
root@dev:/home/ignat# ip address add dev
right 192.168.1.7/24
root@dev:/home/ignat# ip link set right up
root@dev:/home/ignat#
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$ sudo ip link show left
5: left@if2: <BROADCAST,MULTICAST> mtu 1500
qdisc noop state DOWN mode DEFAULT group
default qlen 1000
ignat@dev:~$ sudo ip address add dev left
192.168.1.3/24
ignat@dev:~$
```

```
root@dev:/home/ignat# ip link add right
type veth peer left netns 1
root@dev:/home/ignat# ip link show right
2: right@if5: <BROADCAST,MULTICAST> mtu
1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/ether 06:45:05:83:b7:e8 brd
ff:ff:ff:ff:ff:ff link-netnsid 0
root@dev:/home/ignat# ip address add dev
right 192.168.1.7/24
root@dev:/home/ignat# ip link set right up
root@dev:/home/ignat#
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$ sudo ip link show left
5: left@if2: <BROADCAST,MULTICAST> mtu 1500
qdisc noop state DOWN mode DEFAULT group
default qlen 1000
ignat@dev:~$ sudo ip address add dev left
192.168.1.3/24
ignat@dev:~$ sudo ip link set left up
ignat@dev:~$
```

```
root@dev:/home/ignat# ip link add right
type veth peer left netns 1
root@dev:/home/ignat# ip link show right
2: right@if5: <BROADCAST,MULTICAST> mtu
1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/ether 06:45:05:83:b7:e8 brd
ff:ff:ff:ff:ff:ff link-netnsid 0
root@dev:/home/ignat# ip address add dev
right 192.168.1.7/24
root@dev:/home/ignat# ip link set right up
root@dev:/home/ignat#
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$ sudo ip link show left
5: left@if2: <BROADCAST,MULTICAST> mtu 1500
qdisc noop state DOWN mode DEFAULT group
default qlen 1000
ignat@dev:~$ sudo ip address add dev left
192.168.1.3/24
ignat@dev:~$ sudo ip link set left up
ignat@dev:~$ ping -c 1 192.168.1.7
PING 192.168.1.7 (192.168.1.7) 56(84) bytes of data.
64 bytes from 192.168.1.7: icmp_seq=1 ttl=64 time=0.118
ms

--- 192.168.1.7 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time
0ms
rtt min/avg/max/mdev = 0.118/0.118/0.118/0.000 ms
ignat@dev:~$
```

```
root@dev:/home/ignat# ip link add right
type veth peer left netns 1
root@dev:/home/ignat# ip link show right
2: right@if5: <BROADCAST,MULTICAST> mtu
1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/ether 06:45:05:83:b7:e8 brd
ff:ff:ff:ff:ff:ff link-netnsid 0
root@dev:/home/ignat# ip address add dev
right 192.168.1.7/24
root@dev:/home/ignat# ip link set right up
root@dev:/home/ignat#
```

CLOUDFLARE

Network namespaces example

```
ignat@dev:~$ sudo ip link show left
5: left@if2: <BROADCAST,MULTICAST> mtu 1500
qdisc noop state DOWN mode DEFAULT group
default qlen 1000
ignat@dev:~$ sudo ip address add dev left
192.168.1.3/24
ignat@dev:~$ sudo ip link set left up
ignat@dev:~$ ping -c 1 192.168.1.7
PING 192.168.1.7 (192.168.1.7) 56(84) bytes of data.
64 bytes from 192.168.1.7: icmp_seq=1 ttl=64 time=0.118
ms

--- 192.168.1.7 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time
0ms
rtt min/avg/max/mdev = 0.118/0.118/0.118/0.000 ms
ignat@dev:~$
```

```
root@dev:/home/ignat# ip link add right
type veth peer left netns 1
root@dev:/home/ignat# ip link show right
2: right@if5: <BROADCAST,MULTICAST> mtu
1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/ether 06:45:05:83:b7:e8 brd
ff:ff:ff:ff:ff:ff link-netnsid 0
root@dev:/home/ignat# ip address add dev
right 192.168.1.7/24
root@dev:/home/ignat# ip link set right up
root@dev:/home/ignat# exit
logout
ignat@dev:~$
```

CLOUDFLARE

Network namespaces example

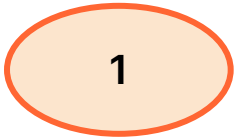
```
ignat@dev:~$ sudo ip link show left
5: left@if2: <BROADCAST,MULTICAST> mtu 1500
qdisc noop state DOWN mode DEFAULT group
default qlen 1000
ignat@dev:~$ sudo ip address add dev left
192.168.1.3/24
ignat@dev:~$ sudo ip link set left up
ignat@dev:~$ ping -c 1 192.168.1.7
PING 192.168.1.7 (192.168.1.7) 56(84) bytes of data.
64 bytes from 192.168.1.7: icmp_seq=1 ttl=64 time=0.118
ms

--- 192.168.1.7 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time
0ms
rtt min/avg/max/mdev = 0.118/0.118/0.118/0.000 ms
ignat@dev:~$ sudo ip link show left
Device "left" does not exist.
```

```
root@dev:/home/ignat# ip link add right
type veth peer left netns 1
root@dev:/home/ignat# ip link show right
2: right@if5: <BROADCAST,MULTICAST> mtu
1500 qdisc noop state DOWN mode DEFAULT
group default qlen 1000
    link/ether 06:45:05:83:b7:e8 brd
ff:ff:ff:ff:ff:ff link-netnsid 0
root@dev:/home/ignat# ip address add dev
right 192.168.1.7/24
root@dev:/home/ignat# ip link set right up
root@dev:/home/ignat# exit
logout
ignat@dev:~$
```

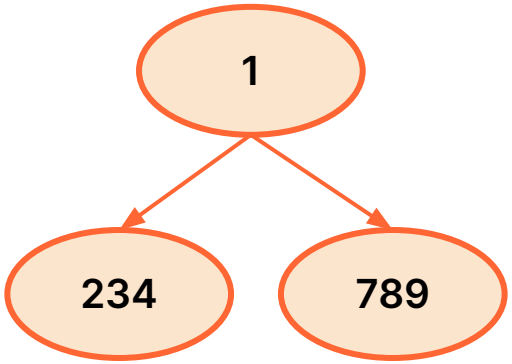
CLOUDFLARE

Process tree



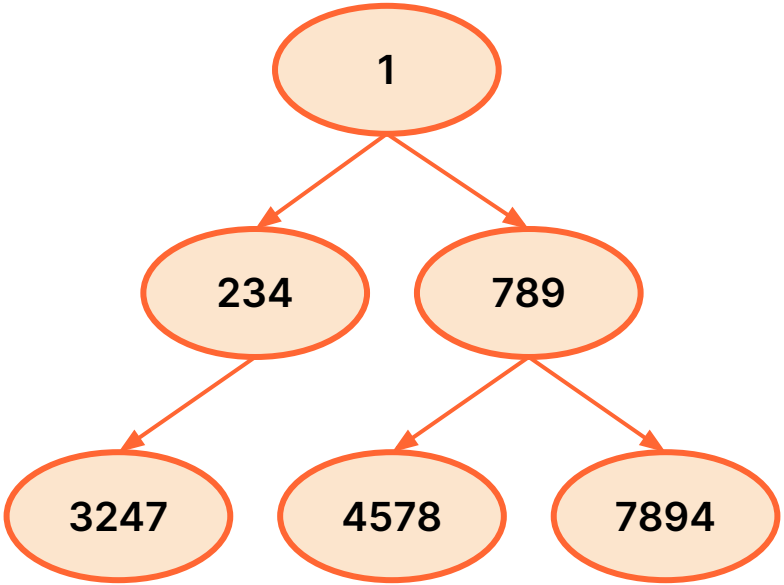
CLOUDFLARE

Process tree



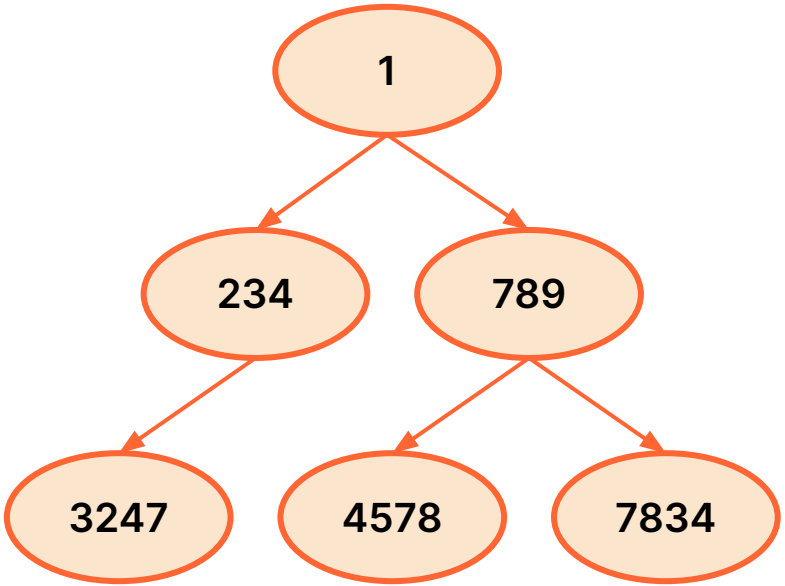
CLOUDFLARE

Process tree



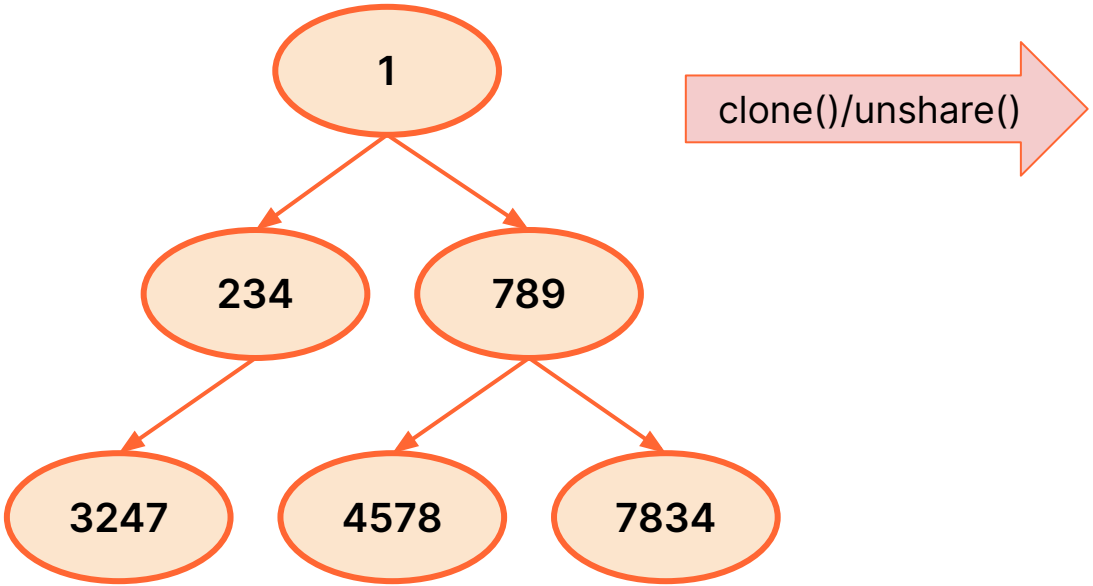
CLOUDFLARE

PID namespaces



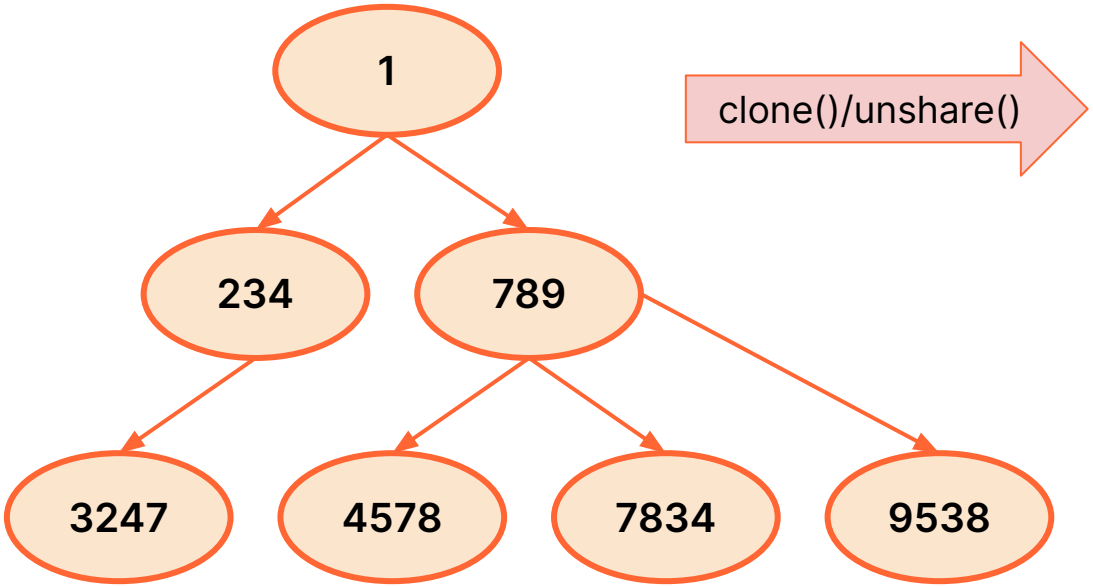
CLOUDFLARE

PID namespaces



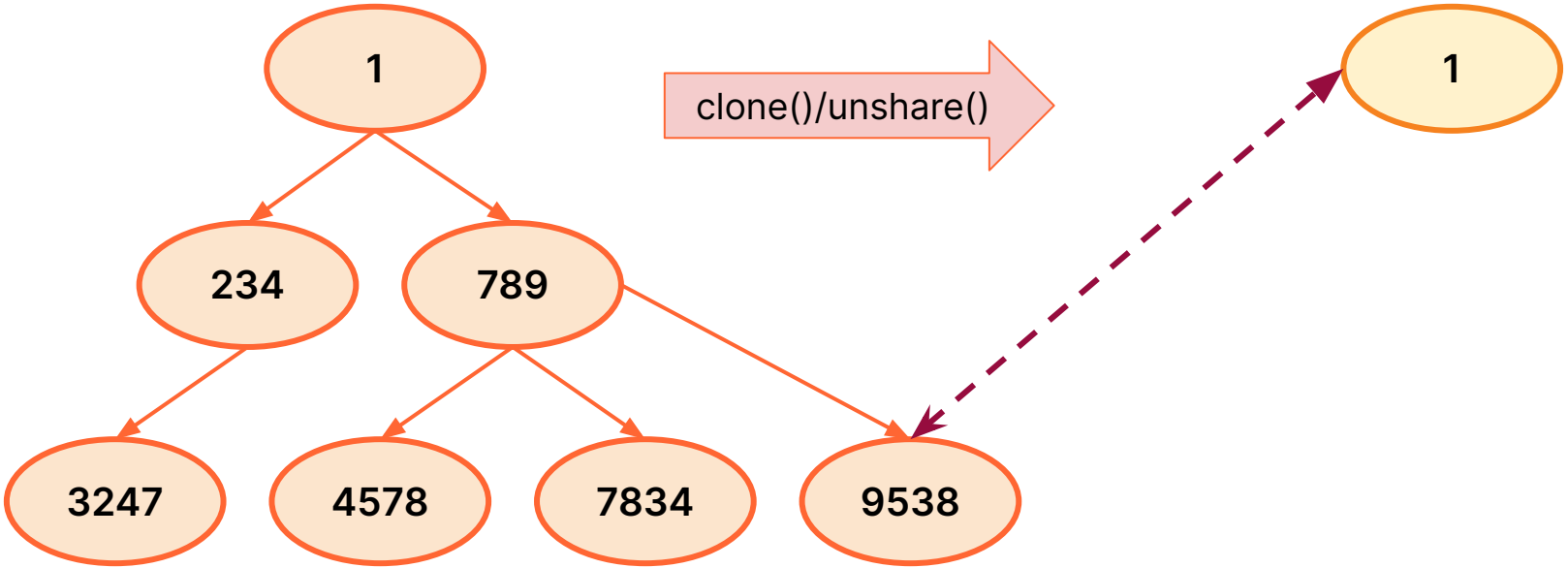
CLOUDFLARE

PID namespaces



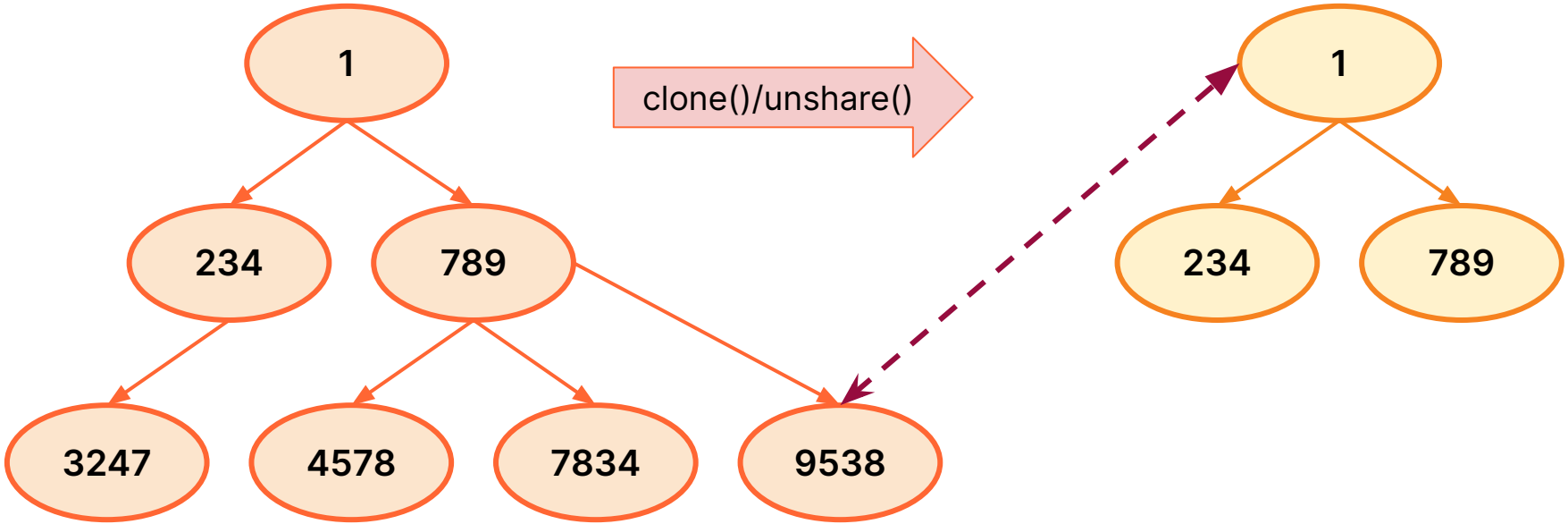
CLOUDFLARE

PID namespaces



CLOUDFLARE

PID namespaces



CLOUDFLARE

PID namespaces example

```
ignat@dev:~$
```

CLOUDFLARE

PID namespaces example

```
ignat@dev:~$ echo $$
7551
ignat@dev:~$
```

CLOUDFLARE

PID namespaces example

```
ignat@dev:~$ echo $$
7551
ignat@dev:~$ sudo unshare --pid --fork
root@dev:/home/ignat#
```

CLOUDFLARE

PID namespaces example

```
ignat@dev:~$ echo $$
7551
ignat@dev:~$ sudo unshare --pid --fork
root@dev:/home/ignat# echo $$
1
root@dev:/home/ignat#
```

CLOUDFLARE

PID namespaces example

```
ignat@dev:~$ echo $$
7551
ignat@dev:~$ sudo unshare --pid --fork
root@dev:/home/ignat# echo $$
1
root@dev:/home/ignat# cd /proc/self
root@dev:/proc/self#
```

CLOUDFLARE

PID namespaces example

```
ignat@dev:~$ echo $$
7551
ignat@dev:~$ sudo unshare --pid --fork
root@dev:/home/ignat# echo $$
1
root@dev:/home/ignat# cd /proc/self
root@dev:/proc/self# grep ^Pid: status
Pid:    7574
root@dev:/proc/self#
```

CLOUDFLARE

PID namespaces example

```
ignat@dev:~$ echo $$
7551
ignat@dev:~$ sudo unshare --pid --fork
root@dev:/home/ignat# echo $$
1
root@dev:/home/ignat# cd /proc/self
root@dev:/proc/self# grep ^Pid: status
Pid:    7574
root@dev:/proc/self# grep ^NSpid: status
NSpid:  7574    1
root@dev:/proc/7571#
```

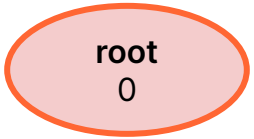
CLOUDFLARE

PID namespaces example

```
ignat@dev:~$ echo $$
7551
ignat@dev:~$ sudo unshare --pid --fork
root@dev:/home/ignat# echo $$
1
root@dev:/home/ignat# cd /proc/self
root@dev:/proc/self# grep ^Pid: status
Pid:    7574
root@dev:/proc/self# grep ^NSpid: status
NSpid:  7574    1
root@dev:/proc/7571# pstree --long --ns-changes --show-pids 7551
bash(7551)---sudo(7571)---sudo(7572)---unshare(7573)---bash(7574,pid)---pstree(7657)
```

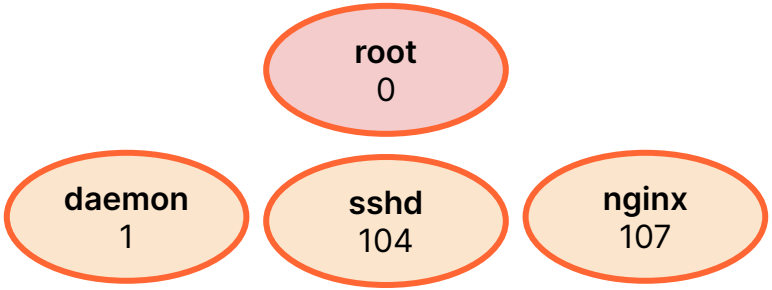
CLOUDFLARE

Linux users



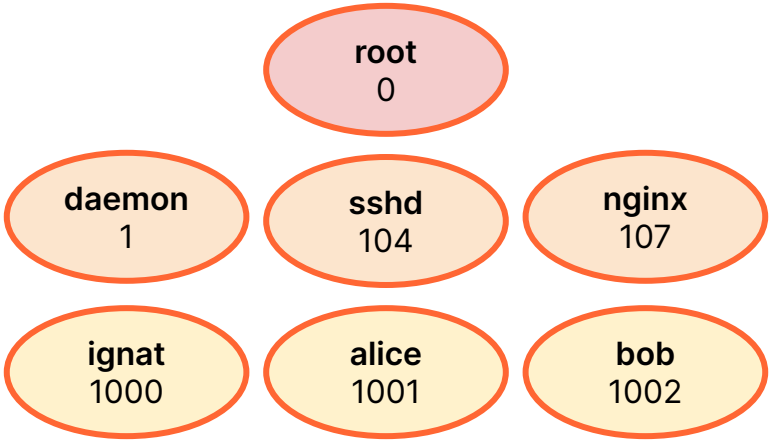
CLOUDFLARE

Linux users



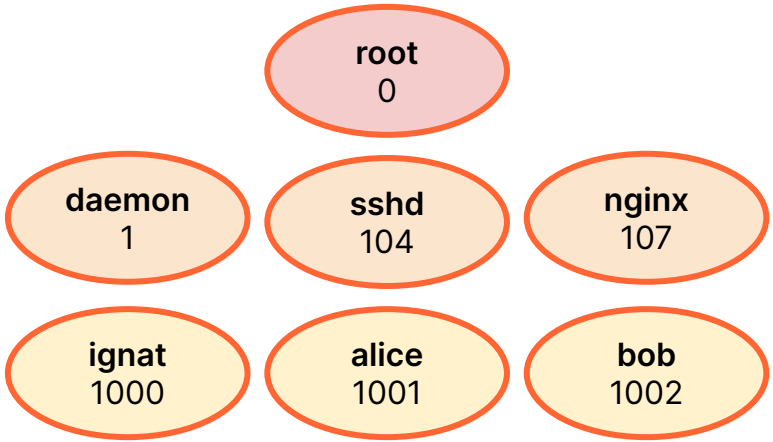
CLOUDFLARE

Linux users



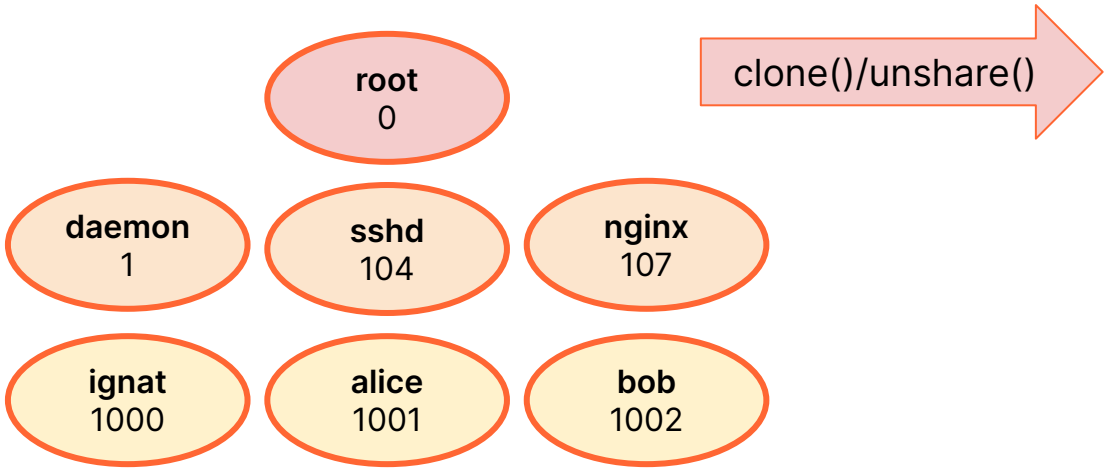
CLOUDFLARE

User namespaces



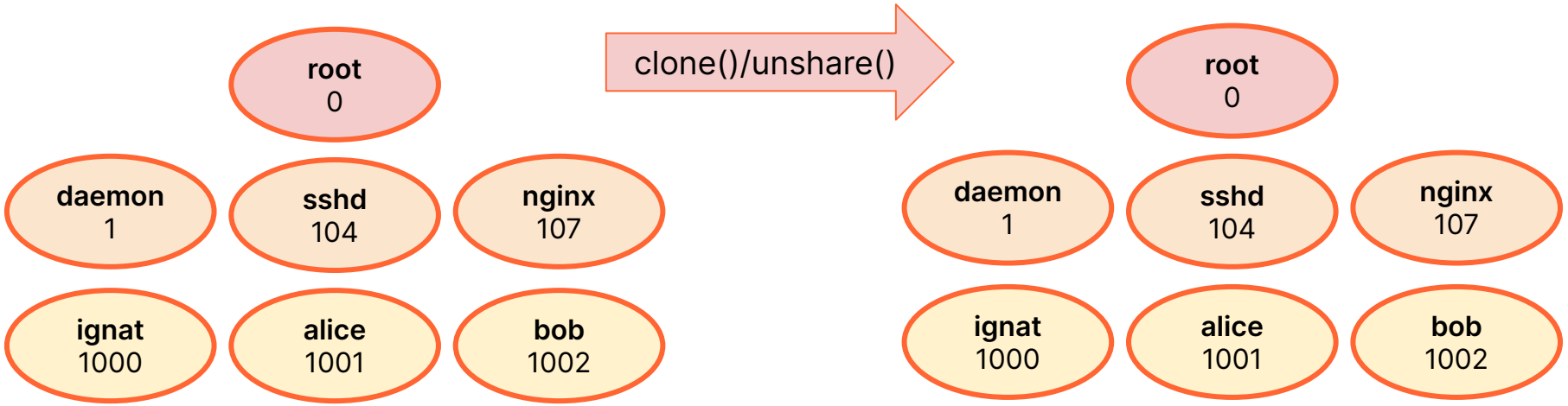
CLOUDFLARE

User namespaces



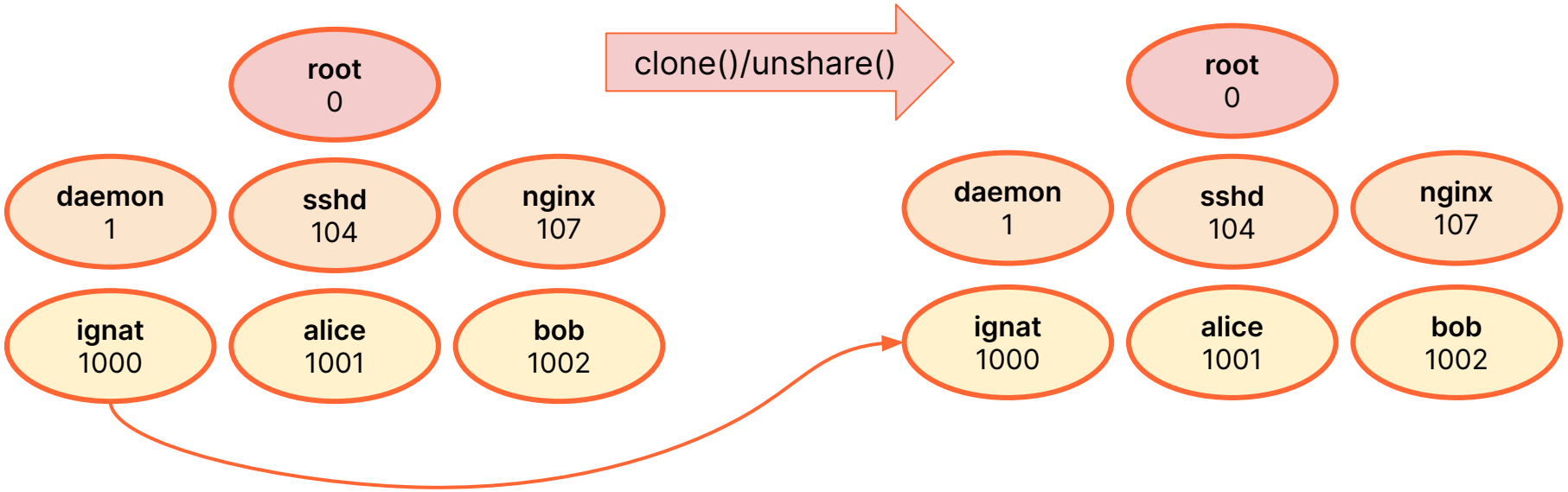
CLOUDFLARE

User namespaces



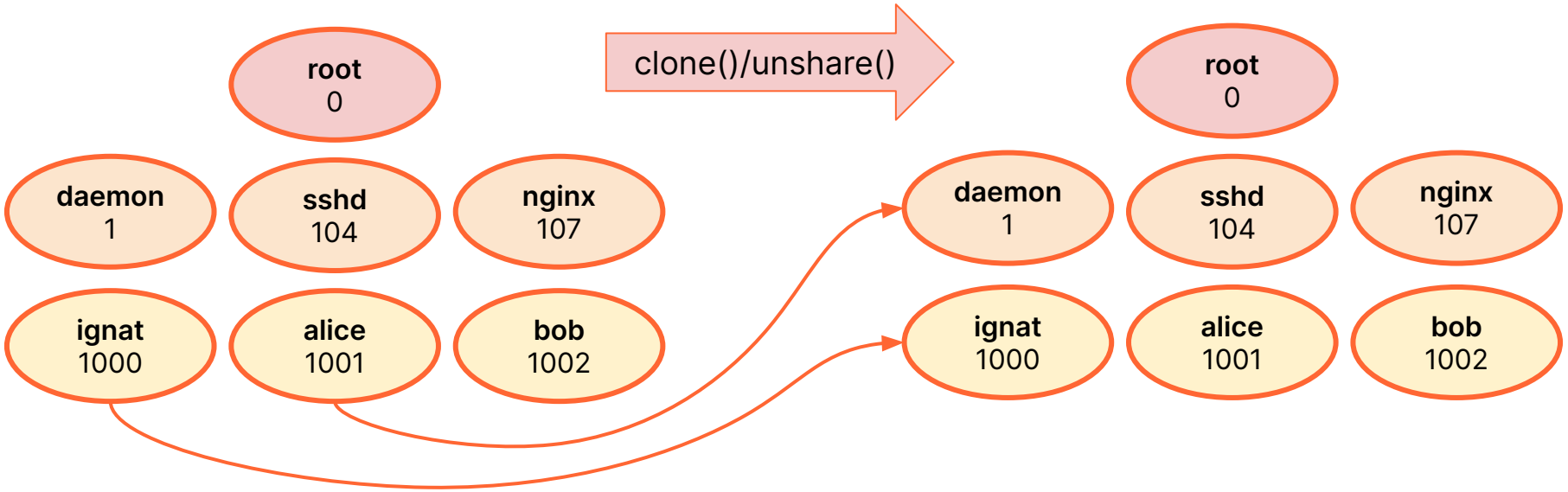
CLOUDFLARE

User namespaces



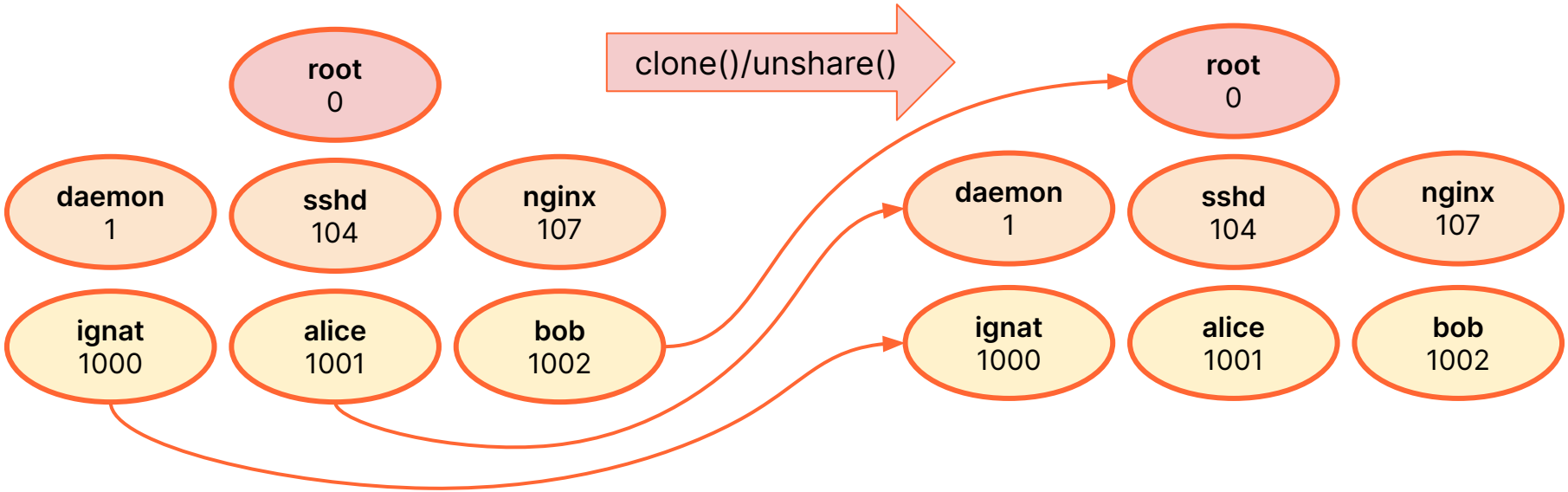
CLOUDFLARE

User namespaces



CLOUDFLARE

User namespaces



CLOUDFLARE

User namespaces example

```
ignat@dev:~$
```

CLOUDFLARE

User namespaces example

```
ignat@dev:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker)
ignat@dev:~$
```

CLOUDFLARE

User namespaces example

```
ignat@dev:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker)
ignat@dev:~$ unshare --user
nobody@dev:~$
```

CLOUDFLARE

User namespaces example

```
ignat@dev:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker)
ignat@dev:~$ unshare --user
nobody@dev:~$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
nobody@dev:~$
```

CLOUDFLARE

User namespaces example

```
ignat@dev:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker)
ignat@dev:~$ unshare --user
nobody@dev:~$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
nobody@dev:~$ exit
logout
ignat@dev:~$
```

CLOUDFLARE

User namespaces example

```
ignat@dev:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker)
ignat@dev:~$ unshare --user
nobody@dev:~$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
nobody@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-user=1 --map-group=1
daemon@dev:~$
```

CLOUDFLARE

User namespaces example

```
ignat@dev:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker)
ignat@dev:~$ unshare --user
nobody@dev:~$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
nobody@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-user=1 --map-group=1
daemon@dev:~$ id
uid=1(daemon) gid=1(daemon) groups=1(daemon),65534(nogroup)
daemon@dev:~$
```

CLOUDFLARE

User namespaces example

```
ignat@dev:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker)
ignat@dev:~$ unshare --user
nobody@dev:~$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
nobody@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-user=1 --map-group=1
daemon@dev:~$ id
uid=1(daemon) gid=1(daemon) groups=1(daemon),65534(nogroup)
daemon@dev:~$ exit
logout
ignat@dev:~$
```

CLOUDFLARE

User namespaces example

```
ignat@dev:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker)
ignat@dev:~$ unshare --user
nobody@dev:~$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
nobody@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-user=1 --map-group=1
daemon@dev:~$ id
uid=1(daemon) gid=1(daemon) groups=1(daemon),65534(nogroup)
daemon@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-user=1337
I have no name!@dev:~$
```

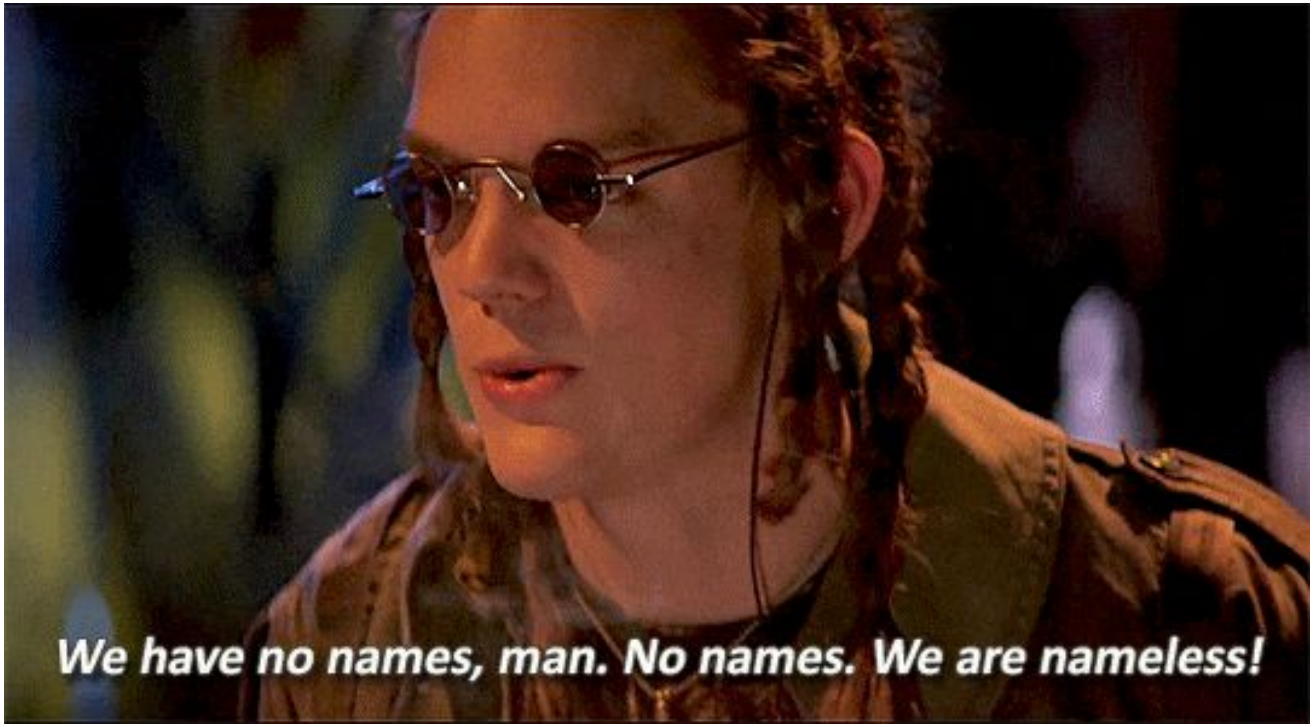
CLOUDFLARE

User namespaces example

```
ignat@dev:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker)
ignat@dev:~$ unshare --user
nobody@dev:~$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
nobody@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-user=1 --map-group=1
daemon@dev:~$ id
uid=1(daemon) gid=1(daemon) groups=1(daemon),65534(nogroup)
daemon@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-user=1337
I have no name!@dev:~$ id
uid=1337 gid=65534(nogroup) groups=65534(nogroup)
I have no name!@dev:~$
```

CLOUDFLARE

User namespaces example



CLOUDFLARE

User namespaces example

```
ignat@dev:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker)
ignat@dev:~$ unshare --user
nobody@dev:~$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
nobody@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-user=1 --map-group=1
daemon@dev:~$ id
uid=1(daemon) gid=1(daemon) groups=1(daemon),65534(nogroup)
daemon@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-user=1337
I have no name!@dev:~$ id
uid=1337 gid=65534(nogroup) groups=65534(nogroup)
I have no name!@dev:~$ exit
logout
ignat@dev:~$
```

CLOUDFLARE

User namespaces example

```
ignat@dev:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker)
ignat@dev:~$ unshare --user
nobody@dev:~$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
nobody@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-user=1 --map-group=1
daemon@dev:~$ id
uid=1(daemon) gid=1(daemon) groups=1(daemon),65534(nogroup)
daemon@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-user=1337
I have no name!@dev:~$ id
uid=1337 gid=65534(nogroup) groups=65534(nogroup)
I have no name!@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-root-user
root@dev:~#
```

CLOUDFLARE

User namespaces example

```
ignat@dev:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker)
ignat@dev:~$ unshare --user
nobody@dev:~$ id
uid=65534(nobody) gid=65534(nogroup) groups=65534(nogroup)
nobody@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-user=1 --map-group=1
daemon@dev:~$ id
uid=1(daemon) gid=1(daemon) groups=1(daemon),65534(nogroup)
daemon@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-user=1337
I have no name!@dev:~$ id
uid=1337 gid=65534(nogroup) groups=65534(nogroup)
I have no name!@dev:~$ exit
logout
ignat@dev:~$ unshare --user --map-root-user
root@dev:~# id
uid=0(root) gid=0(root) groups=0(root),65534(nogroup)
```

CLOUDFLARE

Unprivileged user namespaces

@ignatkn

CLOUDFLARE

Spot the difference

Mount vs User

CLOUDFLARE

Mount vs User

Spot the difference

```
ignat@dev:~$ sudo unshare --mount
...
```

@ignatkn

CLOUDFLARE

Mount vs User

Spot the difference

```
ignat@dev:~$ sudo unshare --mount
...
```

```
ignat@dev:~$ unshare --user
...
```

@ignatkn

CLOUDFLARE

Mount vs User

Spot the difference

```
ignat@dev:~$ sudo unshare --mount
...
```

```
ignat@dev:~$ unshare --user
...
```

@ignatkn

CLOUDFLARE

Mount vs User

Spot the difference

```
ignat@dev:~$ sudo unshare --mount
...
```

```
ignat@dev:~$ unshare --user --map-root-user
...
```

@ignatkn

CLOUDFLARE

Mount vs User

Spot the difference

```
ignat@dev:~$ sudo unshare --mount
...
```

```
ignat@dev:~$ unshare --user --map-root-user
...
```

@ignatkn

CLOUDFLARE

Mount vs User

Spot the difference

```
ignat@dev:~$ sudo unshare --mount
...
```

```
ignat@dev:~$ unshare --user --map-root-user
root@dev:~#
```


UNLIMITED

@ignatkn

CLOUDFLARE

Better together?

```
ignat@dev:~$
```

Mount

+

User

CLOUDFLARE

Mount + User

Better together?

```
ignat@dev:~$ unshare --user
--map-root-user
root@dev:~#
```

CLOUDFLARE

Mount + User

Better together?

```
ignat@dev:~$ unshare --user
--map-root-user
root@dev:~# mountpoint /tmp
/tmp is not a mountpoint
root@dev:~#
```

CLOUDFLARE

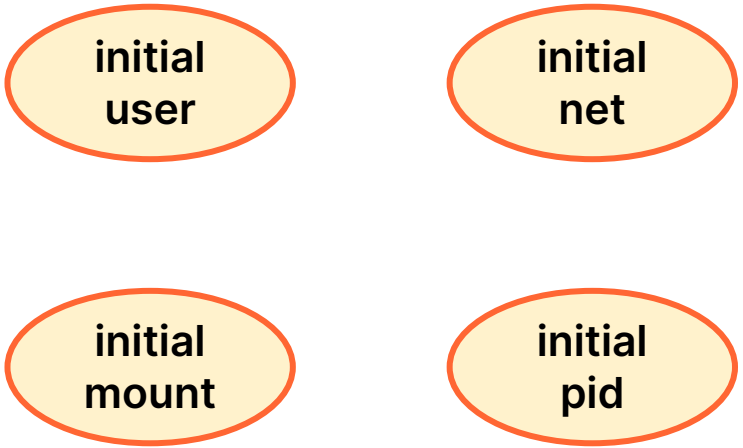
Mount + User

Better together?

```
ignat@dev:~$ unshare --user
--map-root-user
root@dev:~# mountpoint /tmp
/tmp is not a mountpoint
root@dev:~# mount -t tmpfs none /tmp
mount: /tmp: permission denied.
...
root@dev:~#
```

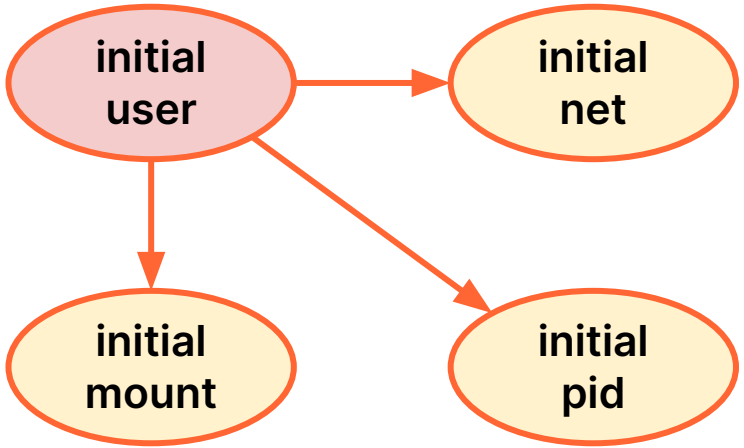
CLOUDFLARE

Namespace ownership



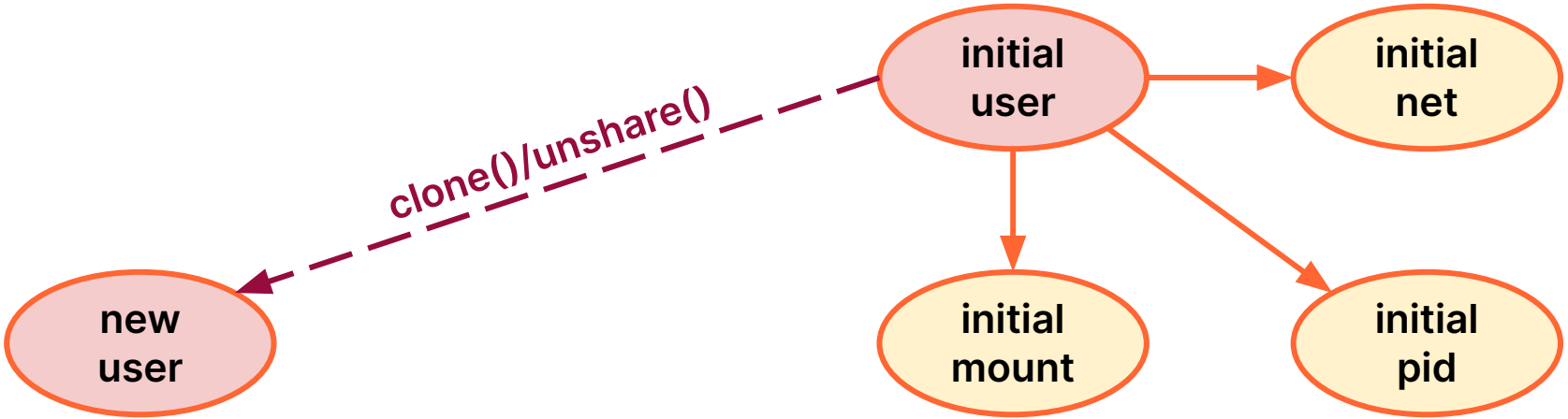
CLOUDFLARE

Namespace ownership



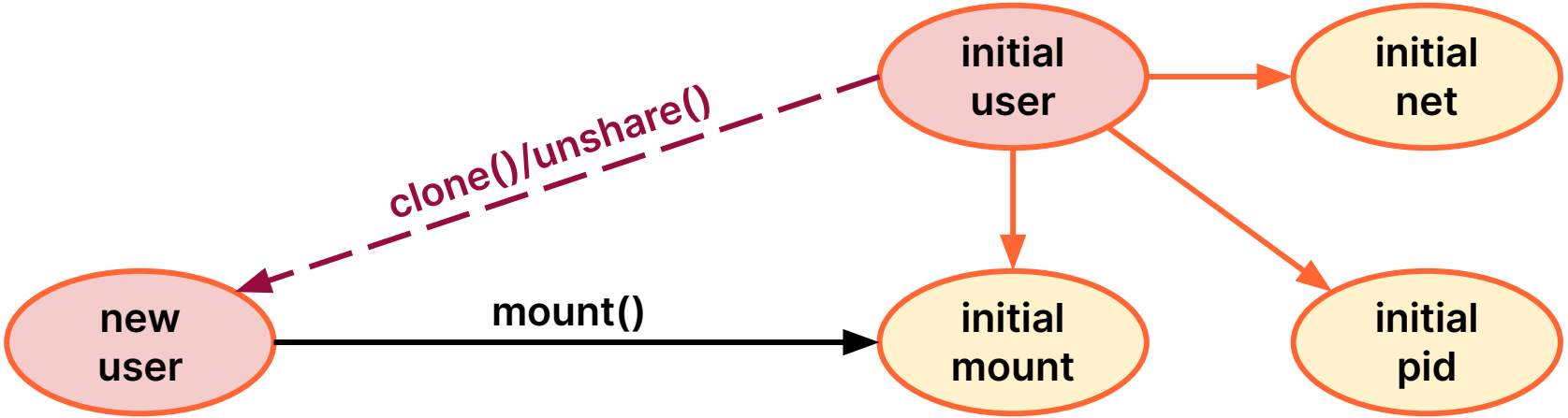
CLOUDFLARE

Namespace ownership



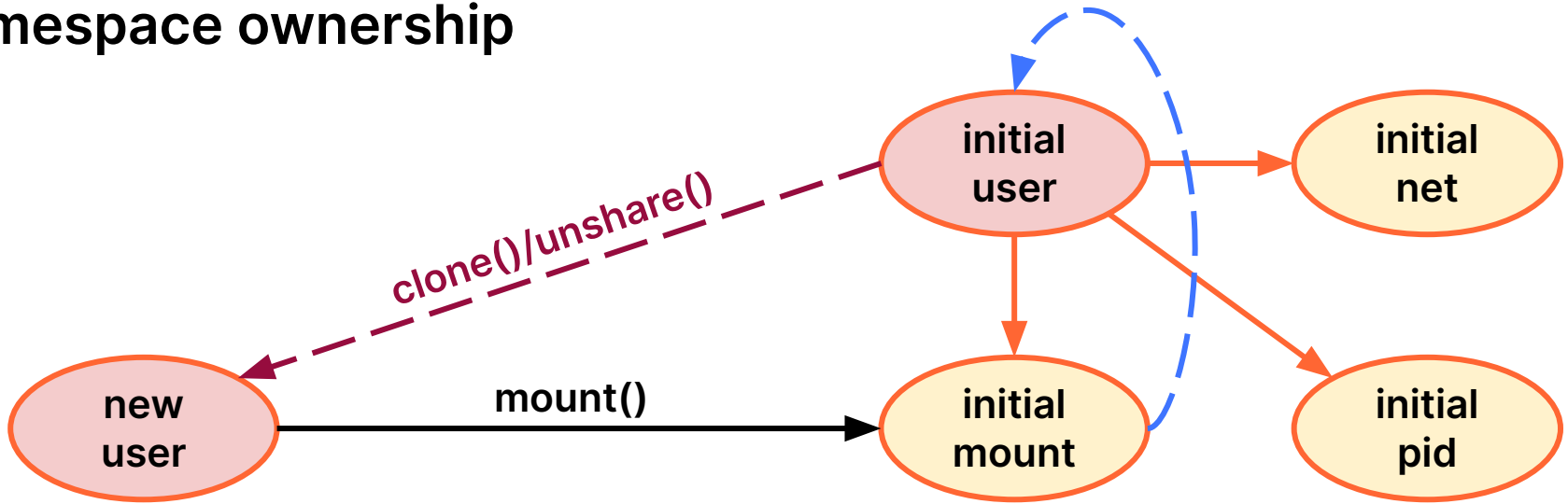
CLOUDFLARE

Namespace ownership



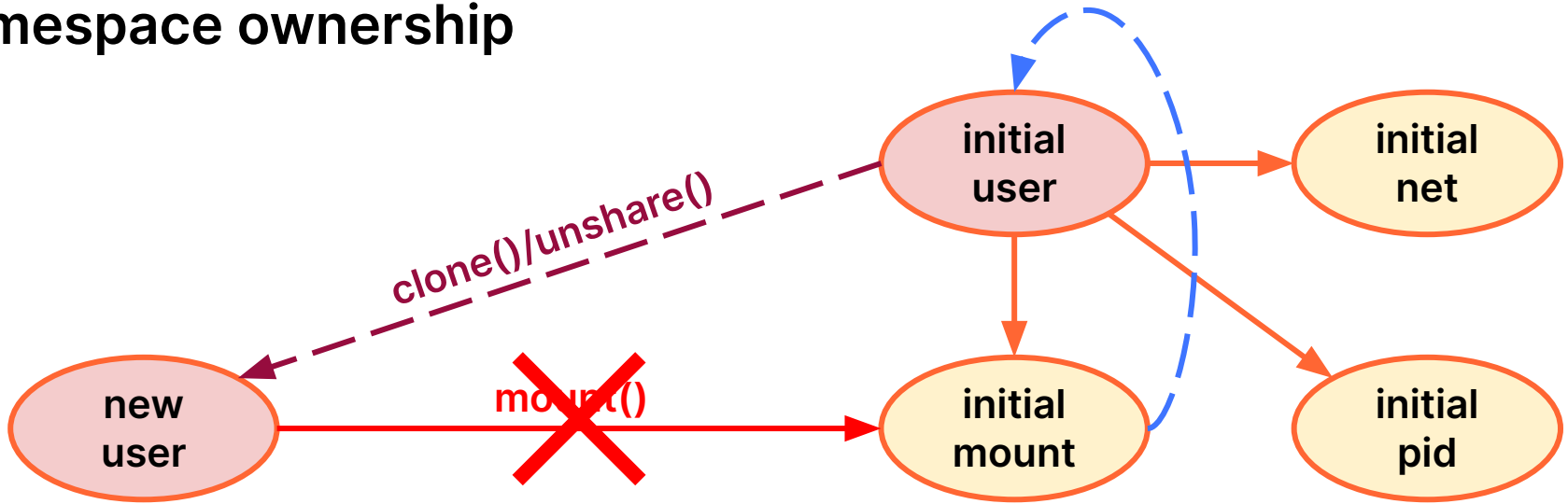
CLOUDFLARE

Namespace ownership



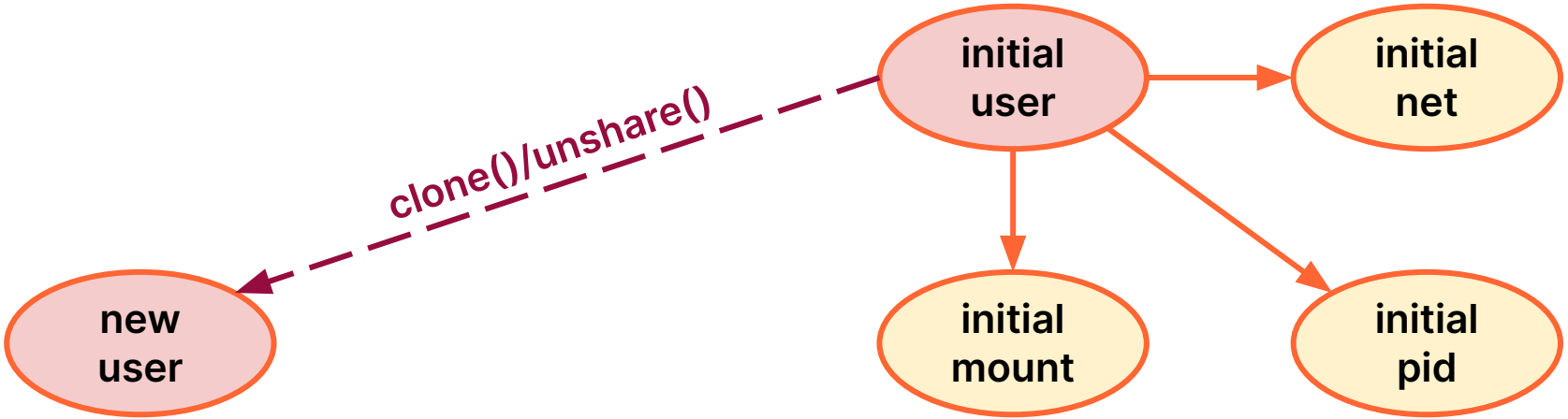
CLOUDFLARE

Namespace ownership



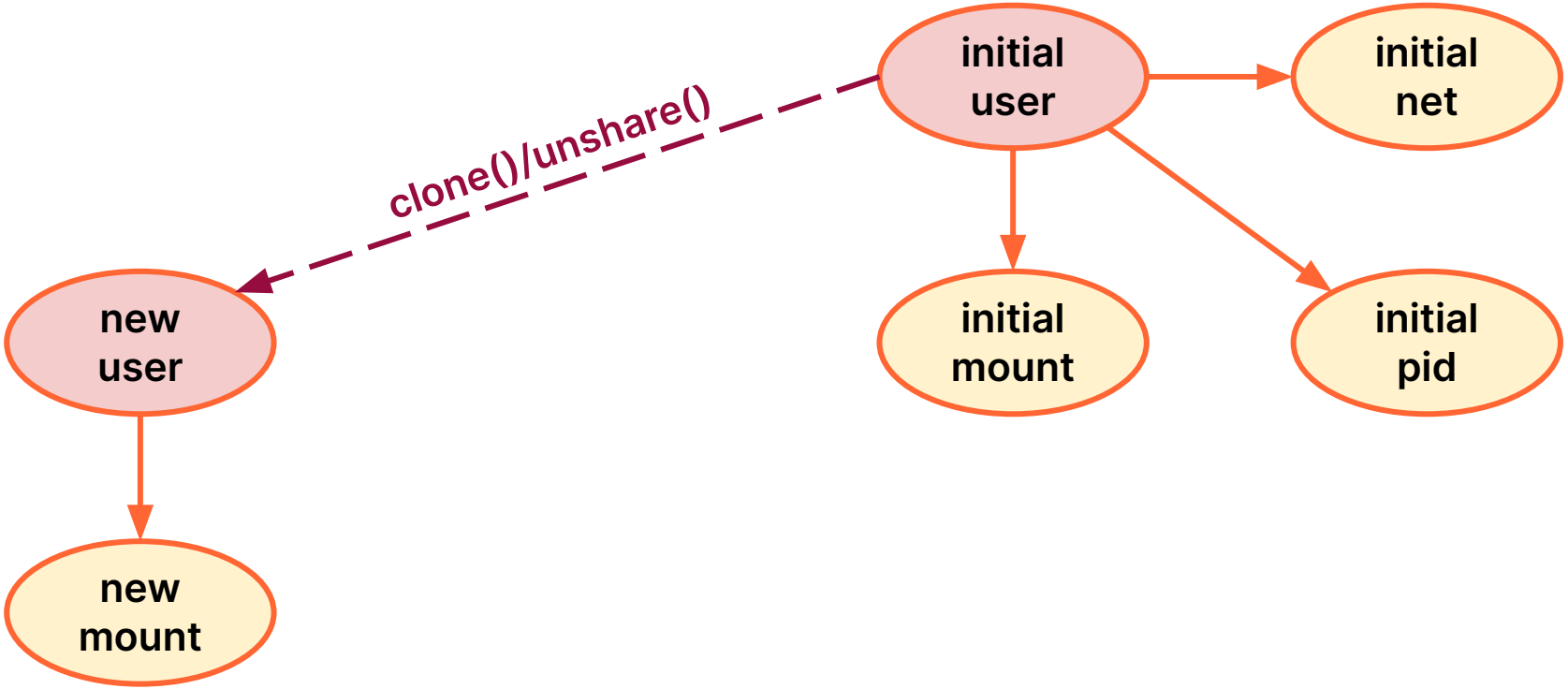
CLOUDFLARE

Namespace ownership



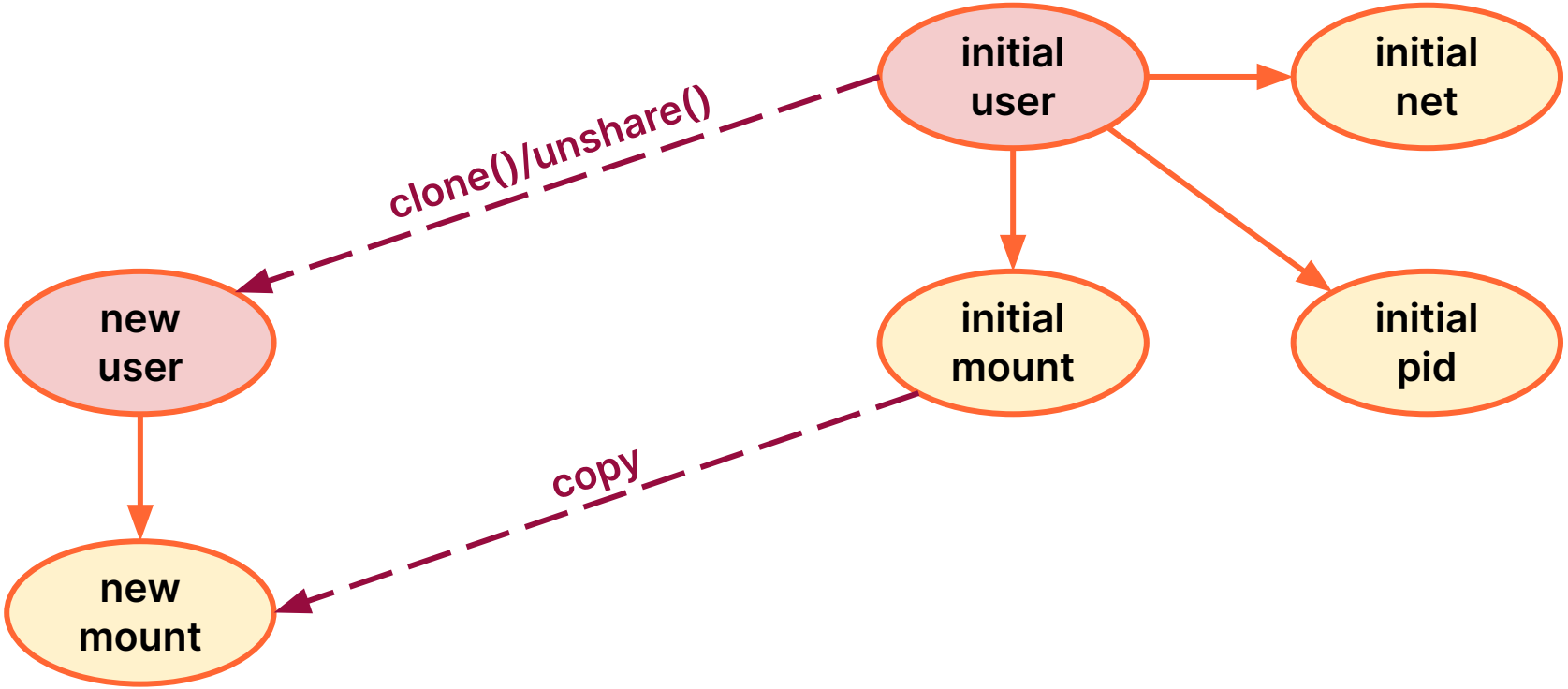
CLOUDFLARE

Namespace ownership



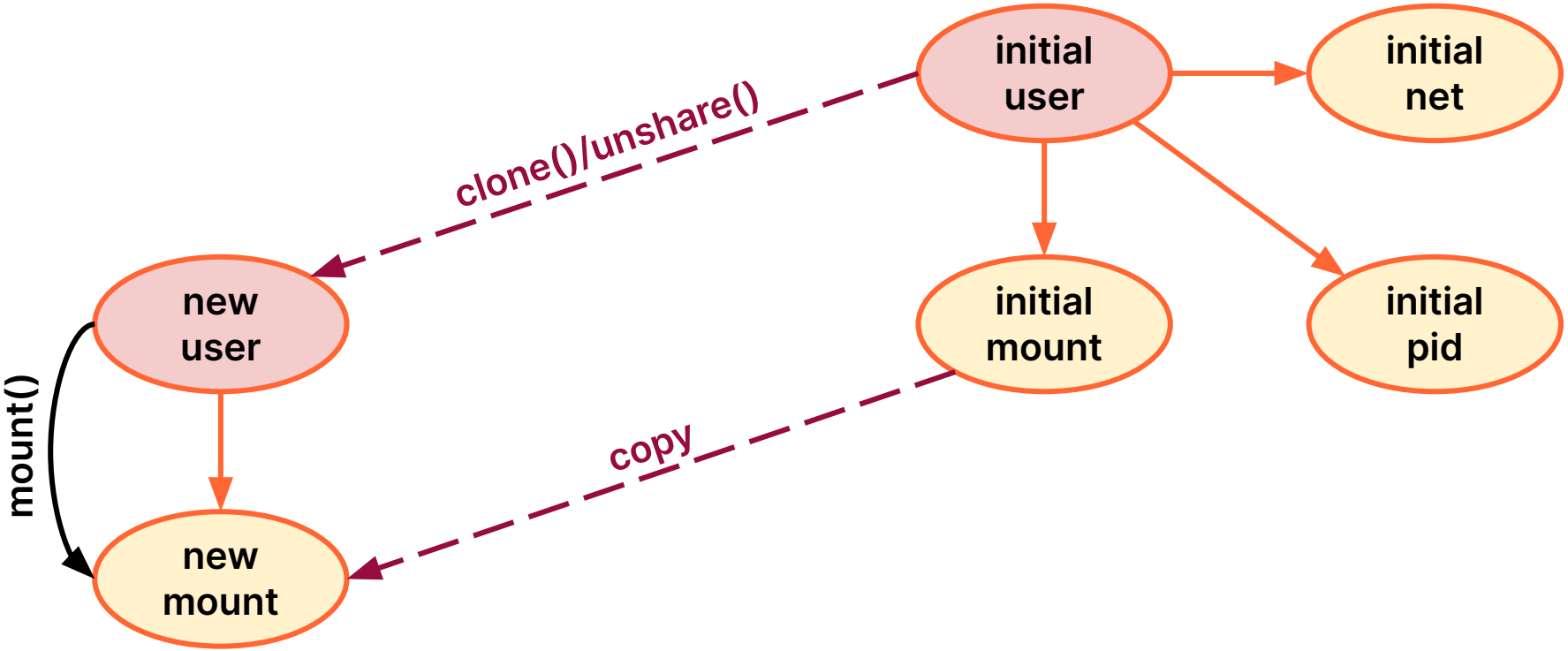
CLOUDFLARE

Namespace ownership



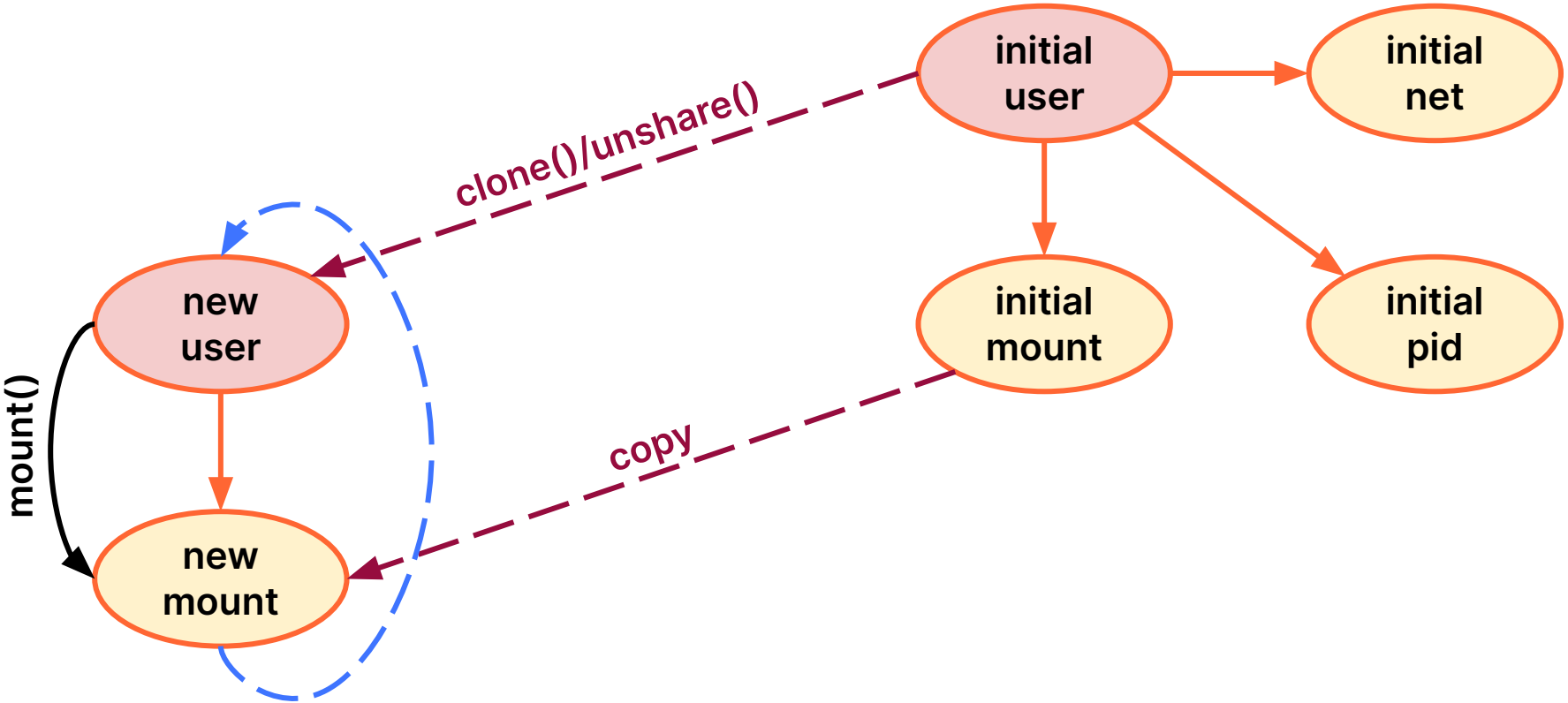
CLOUDFLARE

Namespace ownership



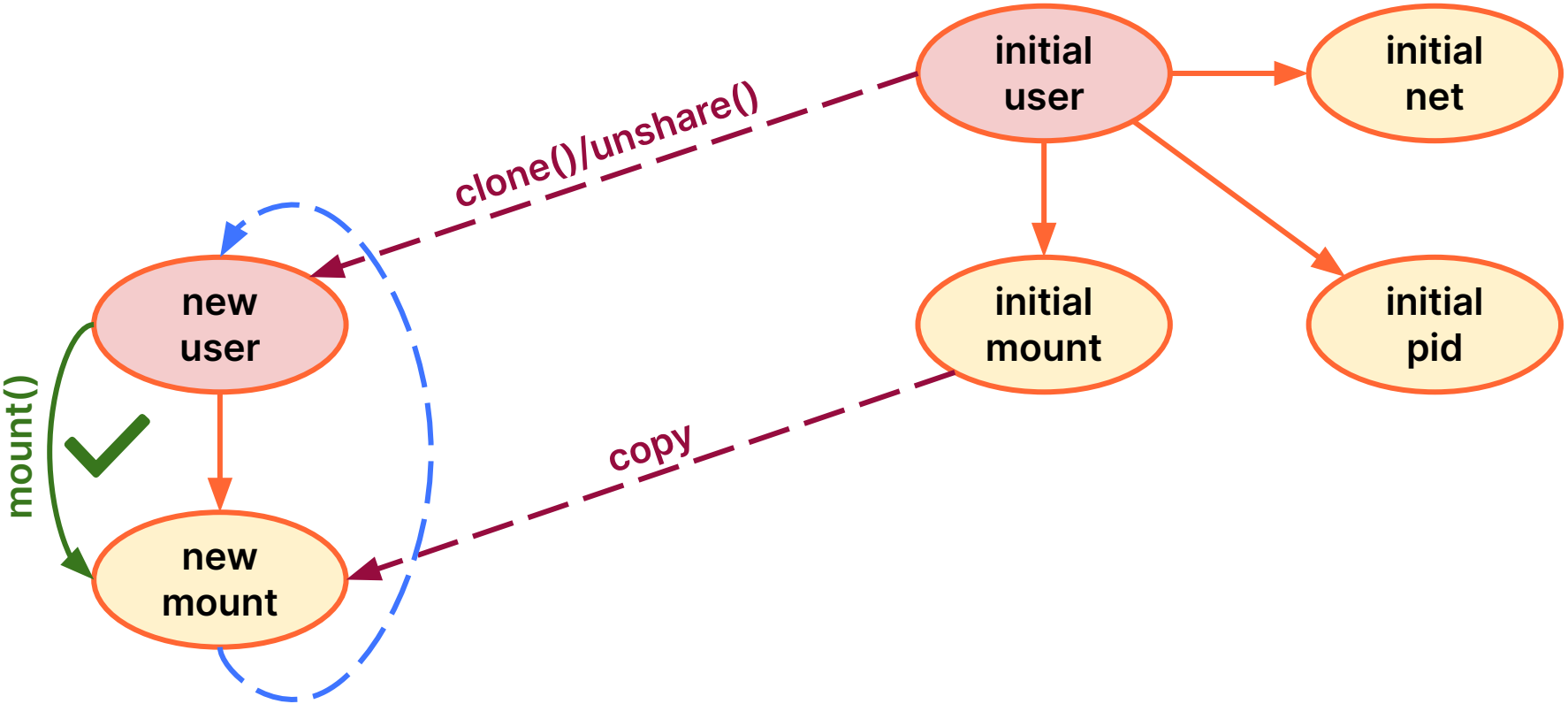
CLOUDFLARE

Namespace ownership



CLOUDFLARE

Namespace ownership



CLOUDFLARE

Mount + User

Better together!

```
ignat@dev:~$ unshare --user
--map-root-user
root@dev:~# mountpoint /tmp
/tmp is not a mountpoint
root@dev:~# mount -t tmpfs none /tmp
mount: /tmp: permission denied.
...
root@dev:~#
```

CLOUDFLARE

Mount + User

Better together!

```
ignat@dev:~$ unshare --user
--map-root-user
root@dev:~# mountpoint /tmp
/tmp is not a mountpoint
root@dev:~# mount -t tmpfs none /tmp
mount: /tmp: permission denied.
...
root@dev:~# unshare --mount
root@dev:~#
```

CLOUDFLARE

Mount + User

Better together!

```
ignat@dev:~$ unshare --user
--map-root-user
root@dev:~# mountpoint /tmp
/tmp is not a mountpoint
root@dev:~# mount -t tmpfs none /tmp
mount: /tmp: permission denied.
...
root@dev:~# unshare --mount
root@dev:~# mount -t tmpfs none /tmp
root@dev:~#
```

@ignatkn

CLOUDFLARE

Mount + User

Better together!

```
ignat@dev:~$ unshare --user
--map-root-user
root@dev:~# mountpoint /tmp
/tmp is not a mountpoint
root@dev:~# mount -t tmpfs none /tmp
mount: /tmp: permission denied.
...
root@dev:~# unshare --mount
root@dev:~# mount -t tmpfs none /tmp
root@dev:~# mountpoint /tmp
/tmp is a mountpoint
```

@ignatkn

CLOUDFLARE

Unprivileged containers

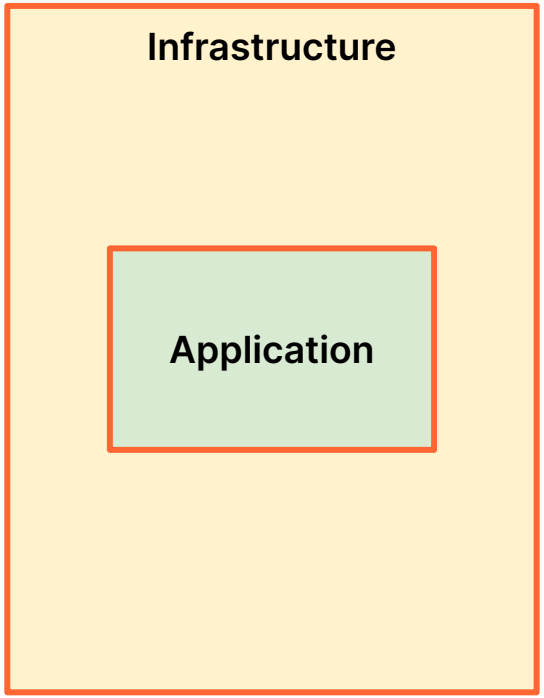
Code sandboxing

CLOUDFLARE

Untrusted (network) input

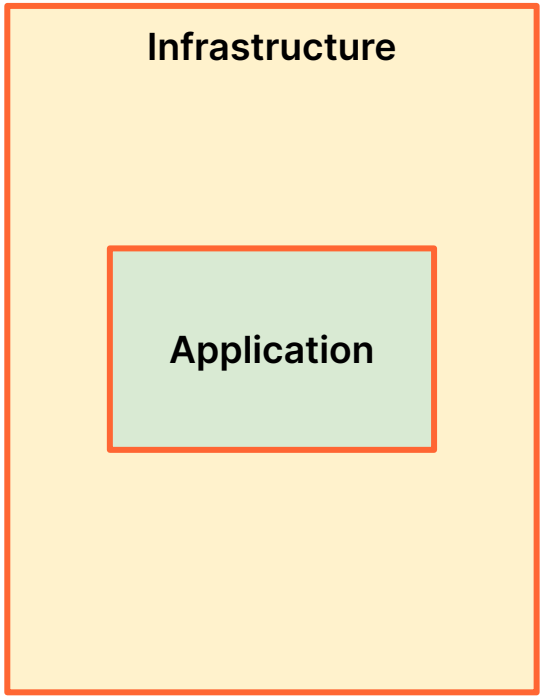
CLOUDFLARE

Untrusted (network) input



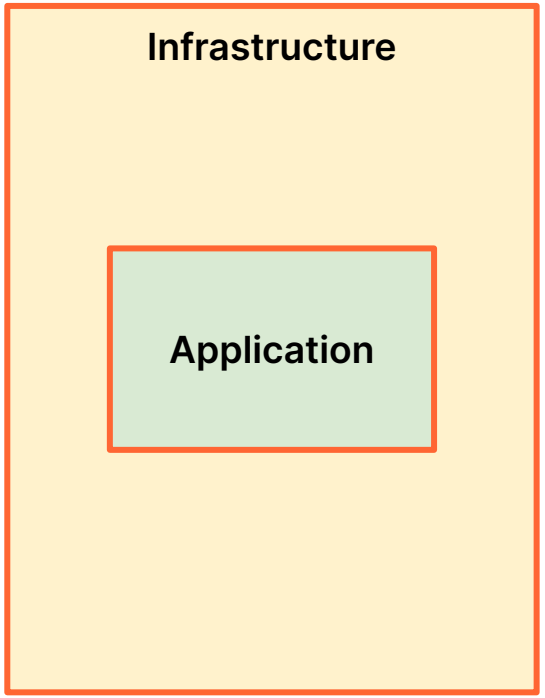
CLOUDFLARE

Untrusted (network) input



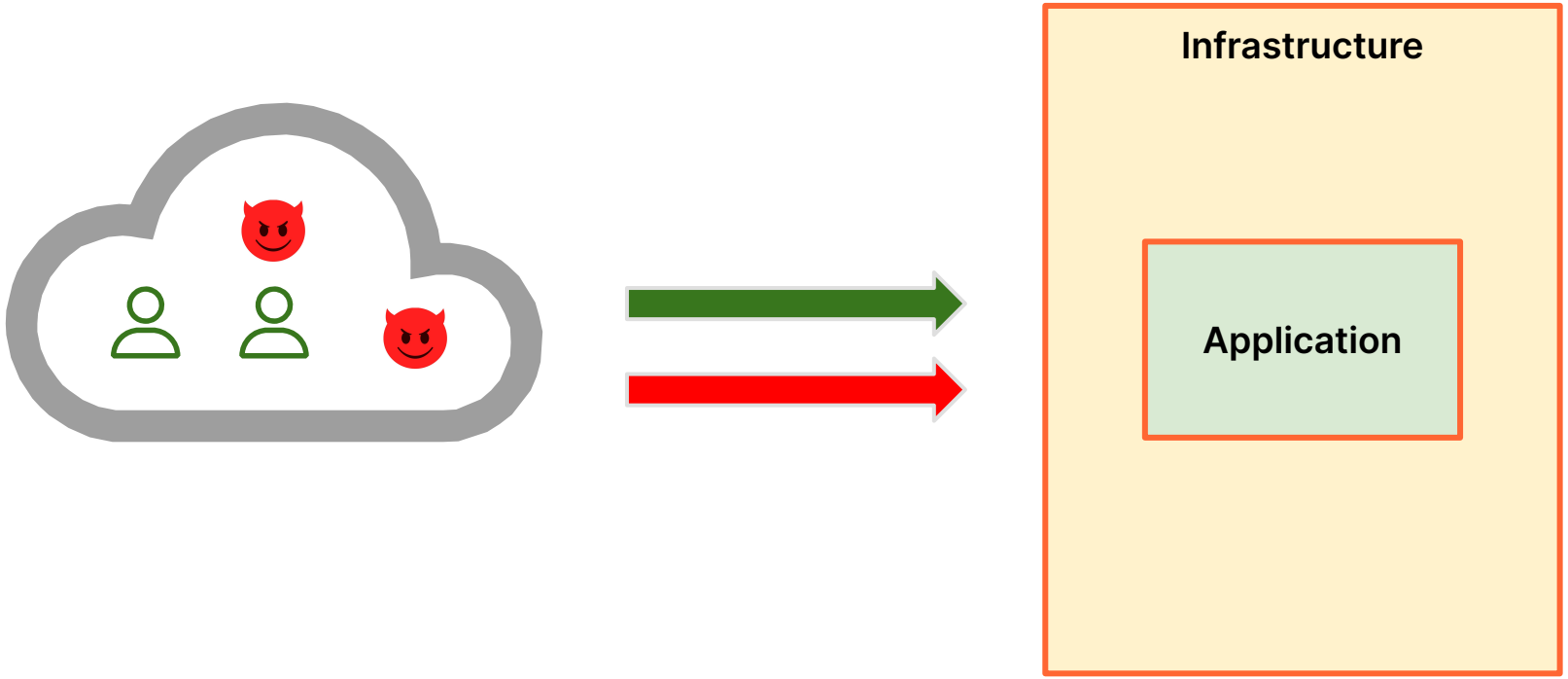
CLOUDFLARE

Untrusted (network) input



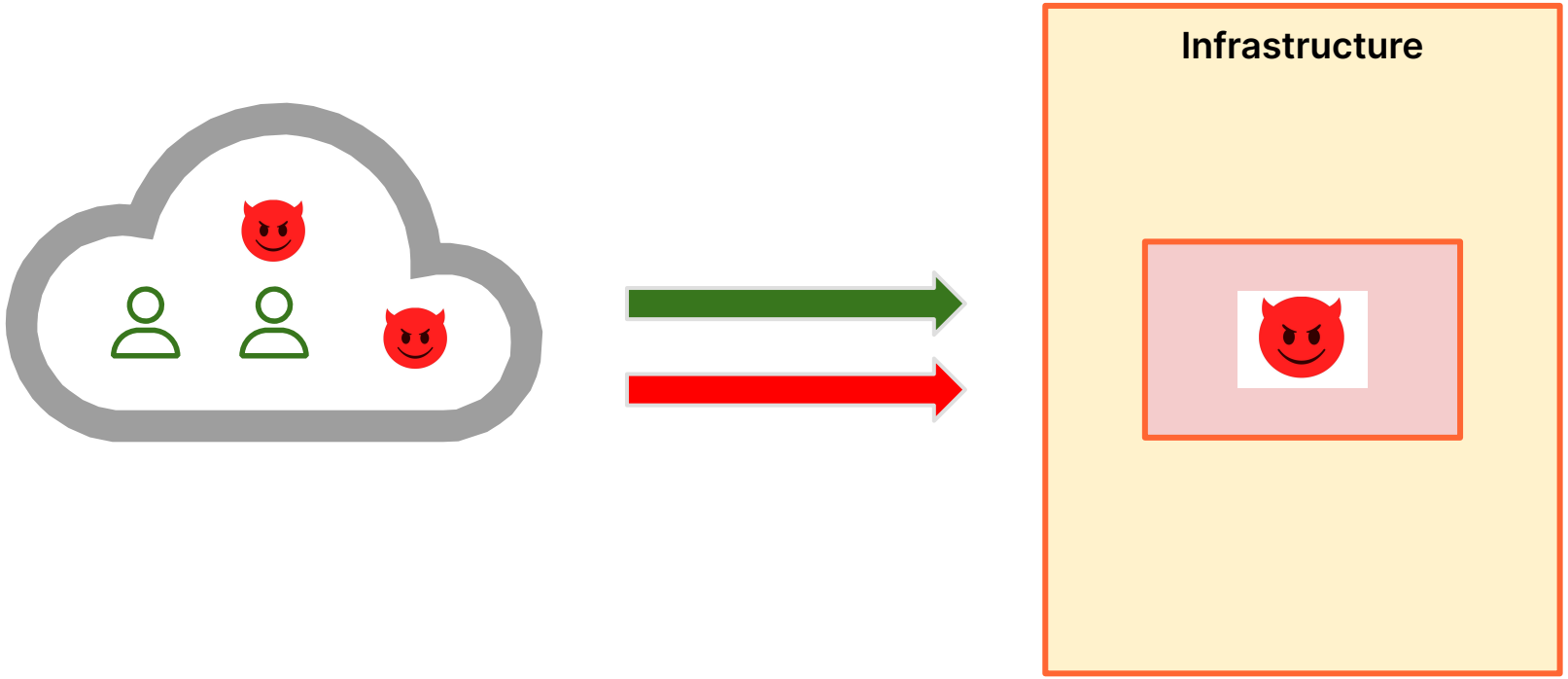
CLOUDFLARE

Untrusted (network) input



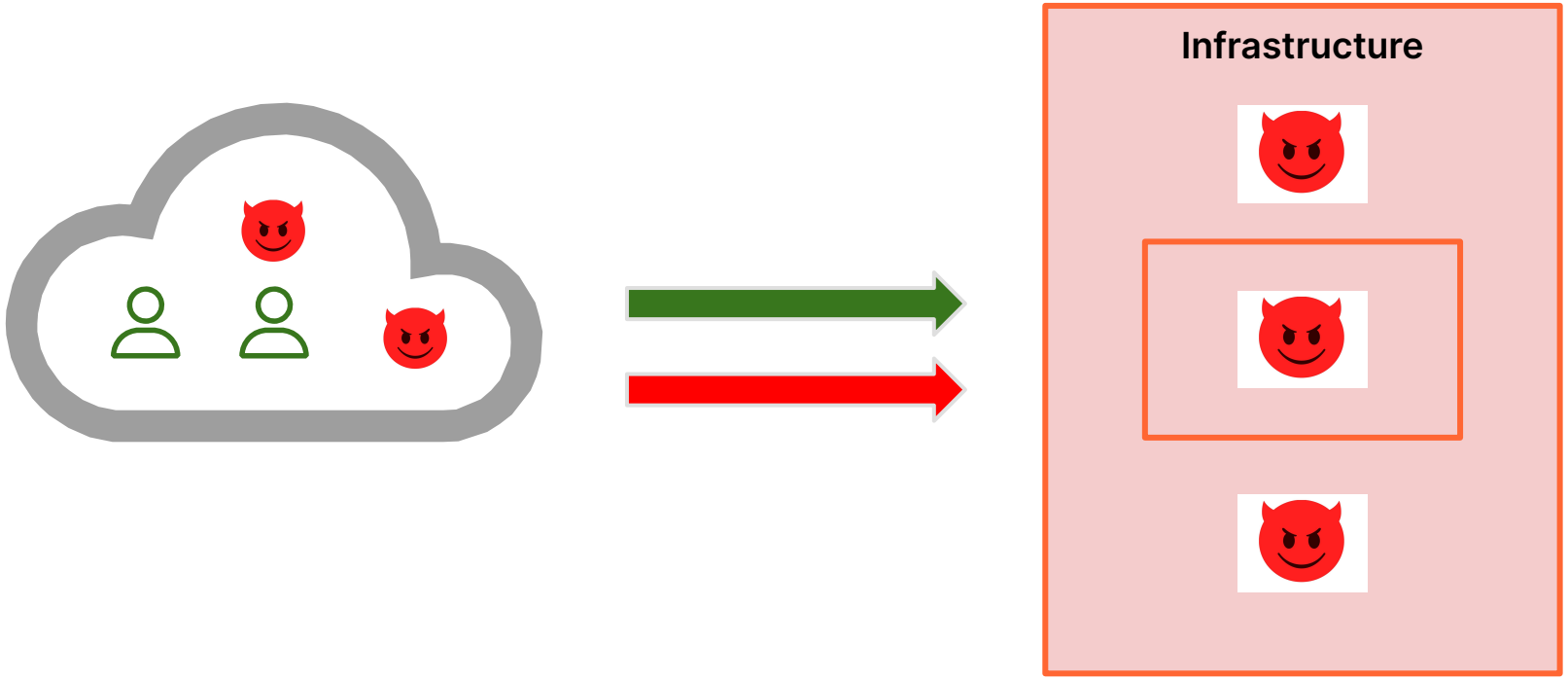
CLOUDFLARE

Untrusted (network) input



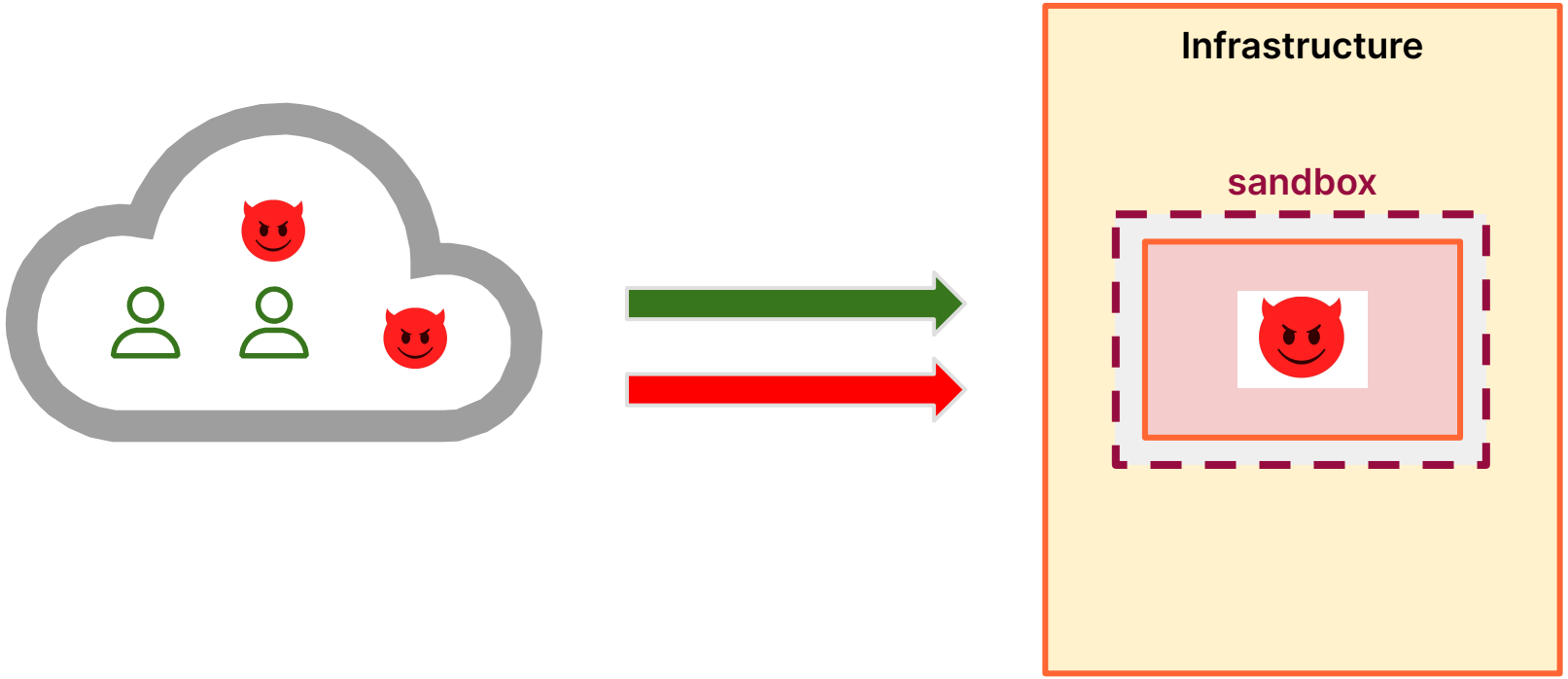
CLOUDFLARE

Untrusted (network) input



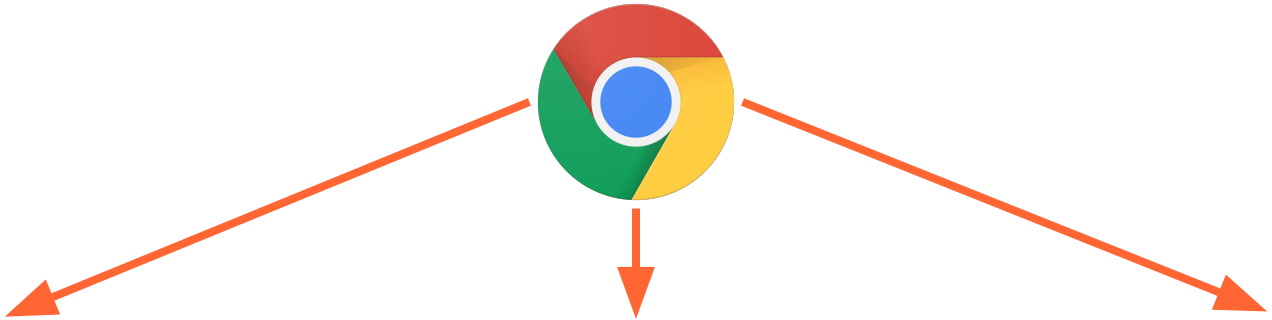
CLOUDFLARE

Untrusted (network) input



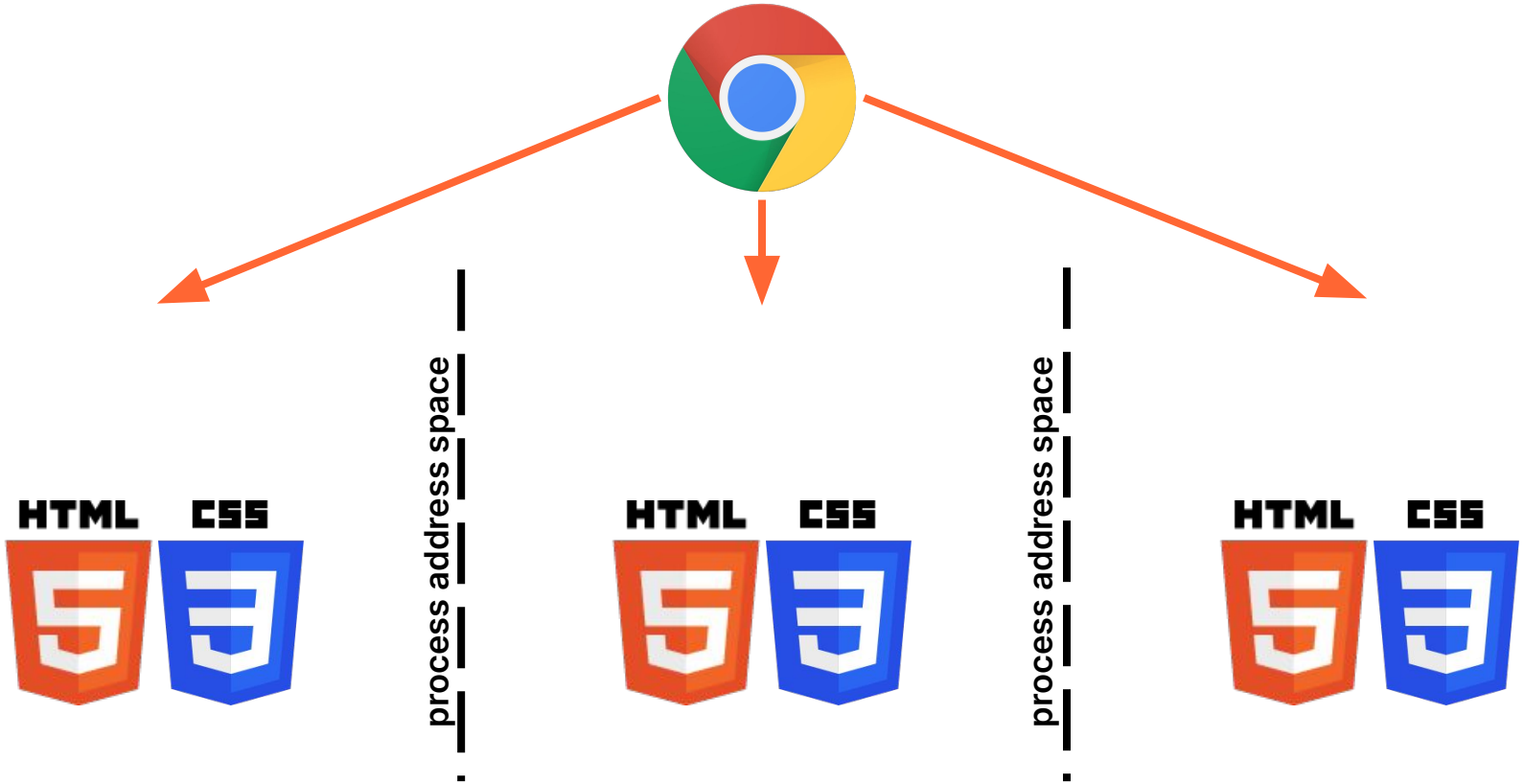
Chrome multi process sandboxing model

Chrome multi process sandboxing model



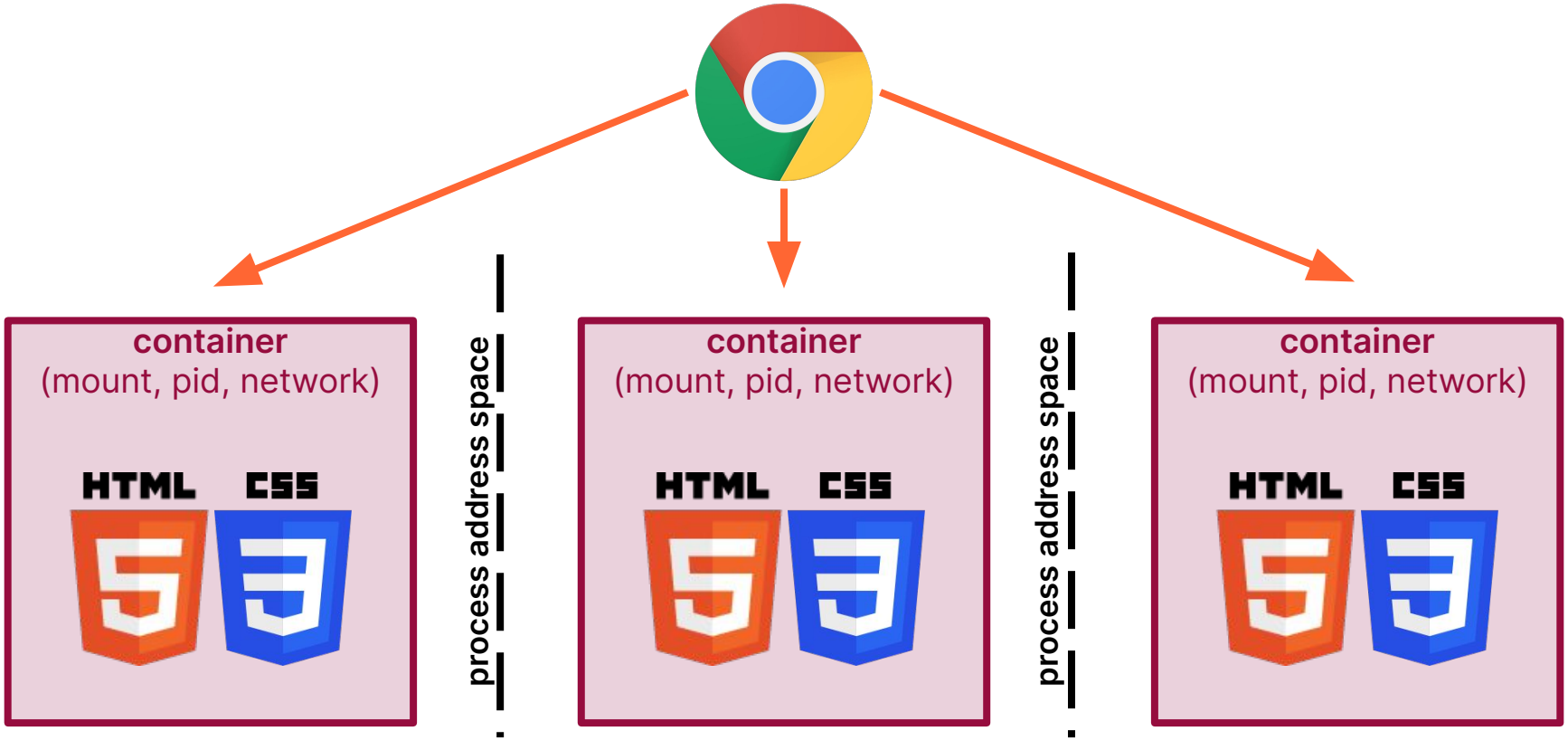
CLOUDFLARE

Chrome multi process sandboxing model



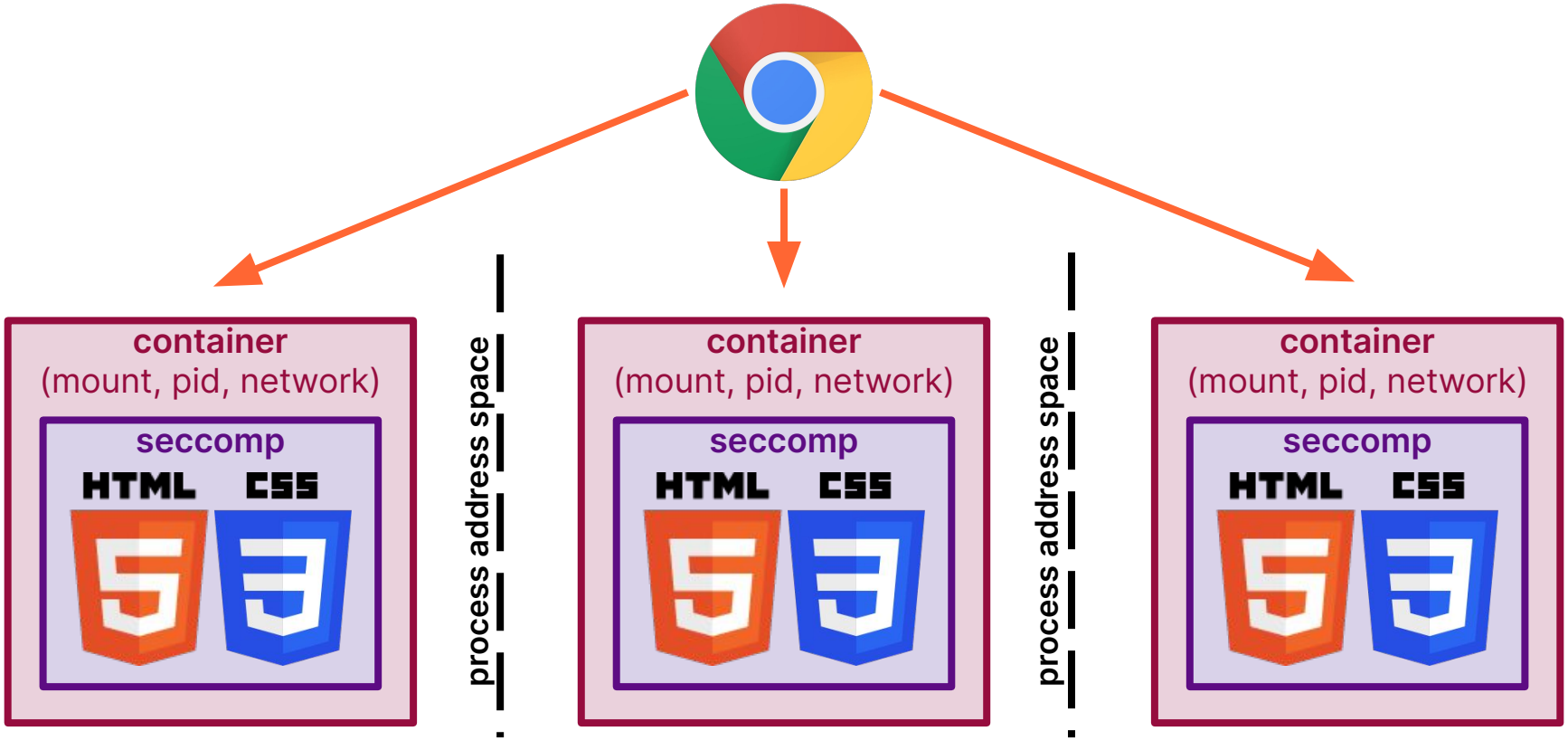
CLOUDFLARE

Chrome multi process sandboxing model



CLOUDFLARE

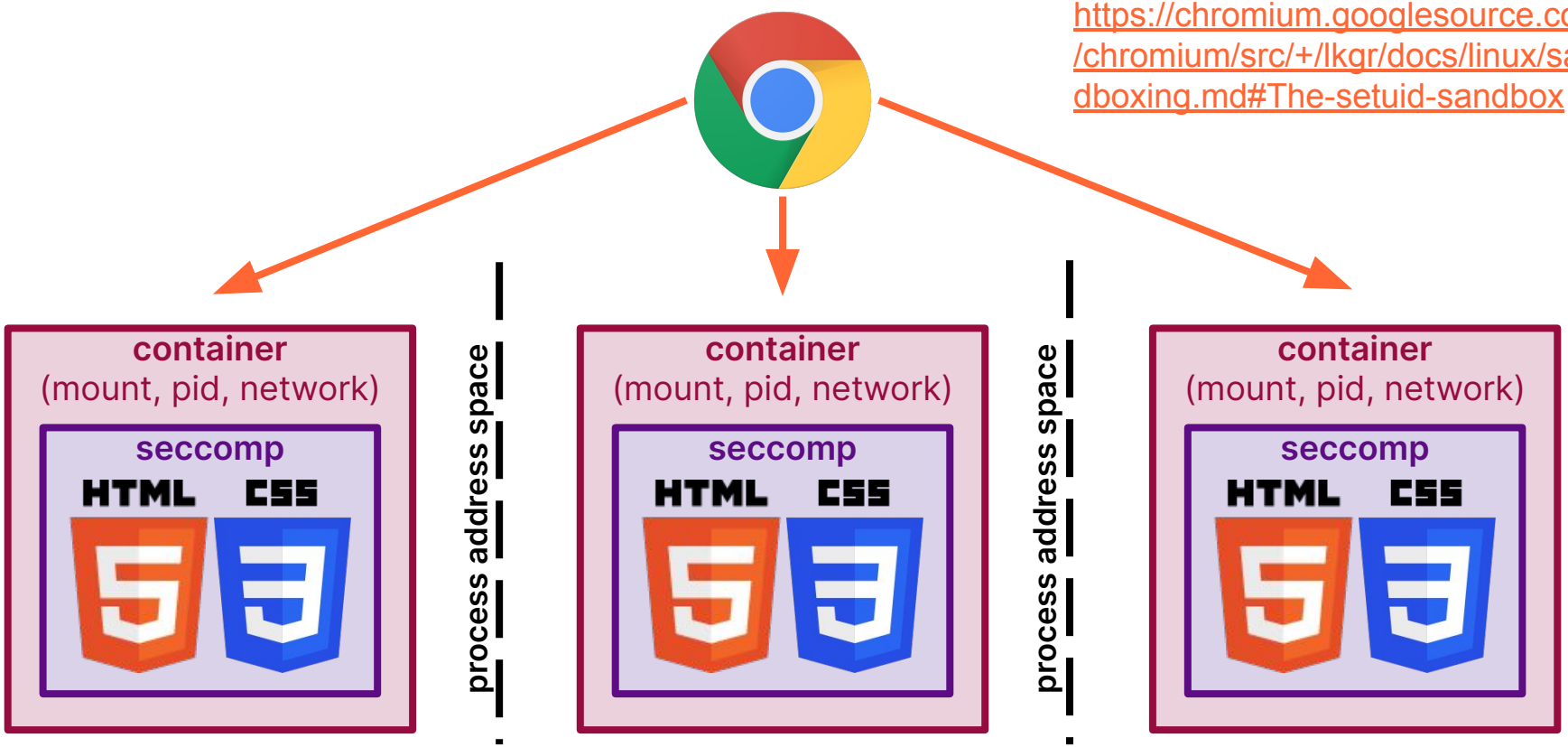
Chrome multi process sandboxing model



CLOUDFLARE

Chrome multi process sandboxing model

https://chromium.googlesource.com/chromium/src/+/lkgr/docs/linux/sandboxing.md#The-setuid-sandbox



CLOUDFLARE

Some other users

- https://rootlesscontaine.rs/
  - allows unprivileged users to fully manage a container runtime

Some other users

- https://rootlesscontaine.rs/
  - allows unprivileged users to fully manage a container runtime
- systemd-coredump
  - forks itself into a unprivileged container to process coredumps

Some other users

- https://rootlesscontaine.rs/
  - allows unprivileged users to fully manage a container runtime
- systemd-coredump
  - forks itself into a unprivileged container to process coredumps
- various testing frameworks, fake root applications etc

CLOUDFLARE

Unprivileged user namespaces are a great tool for applications to create effective sandboxes and harden their security

Unprivileged user namespaces are a great tool for applications to create effective sandboxes and harden their security

Or are they?

CLOUDFLARE

Security hardening guides

10 Sandboxing applications

See also **Wikipedia:Sandbox (computer security)**.

Note: The user namespace configuration item `CONFIG_USER_NS` is currently enabled in **`linux`**, **`linux-lts`**, **`linux-zen`** and **`linux-hardened`**. Lack of it may prevent certain sandboxing features from being made available to applications.

Warning: Unprivileged user namespace usage (`CONFIG_USER_NS_UNPRIVILEGED`) is enabled by default in **`linux`**, **`linux-lts`** and **`linux-zen`**, which greatly increases the attack surface for local privilege escalation (see **FS#36969**).

To mitigate this, either :

- use the **`linux-hardened`** kernel which has the safe default, or
- set the `kernel.unprivileged_userns_clone` **sysctl** to `0` .

Note that this can break applications such as **Zoom**. You will also need to replace **`bubblewrap`** with **`bubblewrap-suid`** if it is installed on your system. See **Bubblewrap#Installation**.

https://wiki.archlinux.org/title/security#Sandboxing_applications

CLOUDFLARE

Security hardening guides

@ignatkn

CLOUDFLARE

A tale of a kernel bug

CVE-2022-47929

A long time ago in a galaxy far, far away....

CLOUDFLARE

Once upon a time there was a bug...

Kernel / KRN-[redacted]

tc crashed with htb

CLOUDFLARE

Once upon a time there was a bug...

Kernel / KRN-[redacted]

tc crashed with htb

Dates

Created: 2019-02-28 13:44 GMT

CLOUDFLARE

Once upon a time there was a bug...

```
From: Ignat Korchagin <ignat@cloudflare.com>
To: netdev@vger.kernel.org
Cc: Daniel Dao <dqminh@cloudflare.com>,
        Ivan Babrou <ivan@cloudflare.com>,
        Frank Hofmann <fhofmann@cloudflare.com>,
        Nicolae Vartolomei <nv@cloudflare.com>
Subject: noqueue qdisc and non-virtual interfaces
Date: Fri, 1 Mar 2019 12:19:07 +0000    [thread overview]
Message-ID: <CALrw=nEdA0asN4n7B3P2TyHKJ+UBPvoAiMrwkT42=fqp2-CPiw@mail.gmail.com> (raw)

Good day,

According to this link:
http://linux-tc-notes.sourceforge.net/tc/doc/sch_noqueue.txt it should
not be possible to assign noqueue qdisc to physical devices or
classes. Yet, I can clearly do it even on the latest 5.0-rc series
kernels:

$ sudo tc qdisc add dev eth0 root noqueue
$ sudo tc qdisc show dev eth0
qdisc noqueue 8005: root refcnt 385

If I understand the source correctly, noqueue qdisc was designed only
for network interfaces, which have IFF_NO_QUEUE in priv_flags in
net_device structure. However, I see no checks in the code, which
enforce it when attaching this qdisc to an interface.

I believe, it was not possible before, but commit
d66d6c3152e8d5a6db42a56bf7ae1c6cae87ba48 changed that.

Regardless, if it is intentional or not, the change is incomplete as
it introduces a potential NULL ptr dereference bug, which can be
triggered by doing something like:

dev="eth0"
sudo tc qdisc replace dev $dev root noqueue
sudo tc qdisc delete dev $dev root
sudo tc qdisc replace dev $dev root handle 1: htb default 1
sudo tc class add dev $dev parent 1: classid 1:1 htb rate 1000Gbps
sudo tc qdisc add dev $dev parent 1:1 handle 10: noqueue

I believe it should not be possible to have noqueue qdisc on physical
nics, but, please, let me know if I'm wrong.

Regards,
Ignat
```

https://lore.kernel.org/all/CALrw=nEdA0asN4n7B3P2TyHKJ+UBPvoAiMrwkT42=fqp2-CPiw@mail.gmail.com/

CLOUDFLARE

Once upon a time there was a bug...

CLOUDFLARE

Once upon a time there was a bug...

```
dev="eth0"
sudo tc qdisc replace dev $dev root noqueue
sudo tc qdisc delete dev $dev root
sudo tc qdisc replace dev $dev root handle 1: htb default 1
sudo tc class add dev $dev parent 1: classid 1:1 htb rate 1000Gbps
sudo tc qdisc add dev $dev parent 1:1 handle 10: noqueue
```

CLOUDFLARE

Once upon a time there was a bug...

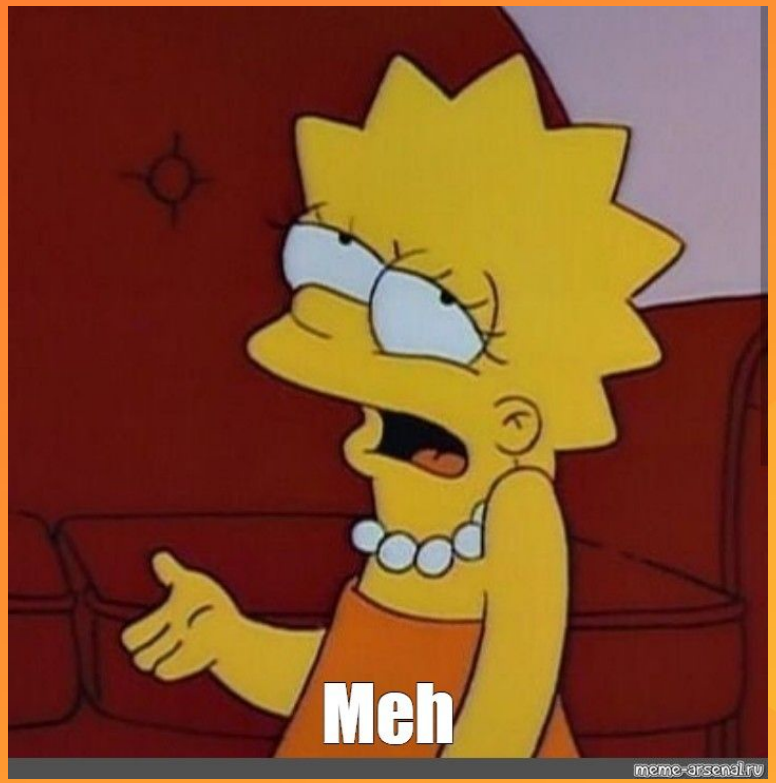
```
dev="eth0"
sudo tc qdisc replace dev $dev root noqueue
sudo tc qdisc delete dev $dev root
sudo tc qdisc replace dev $dev root handle 1: htb default 1
sudo tc class add dev $dev parent 1: classid 1:1 htb rate 1000Gbps
sudo tc qdisc add dev $dev parent 1:1 handle 10: noqueue
```

So the system admin can crash the system?

CLOUDFLARE

“

So the system admin can crash the system?

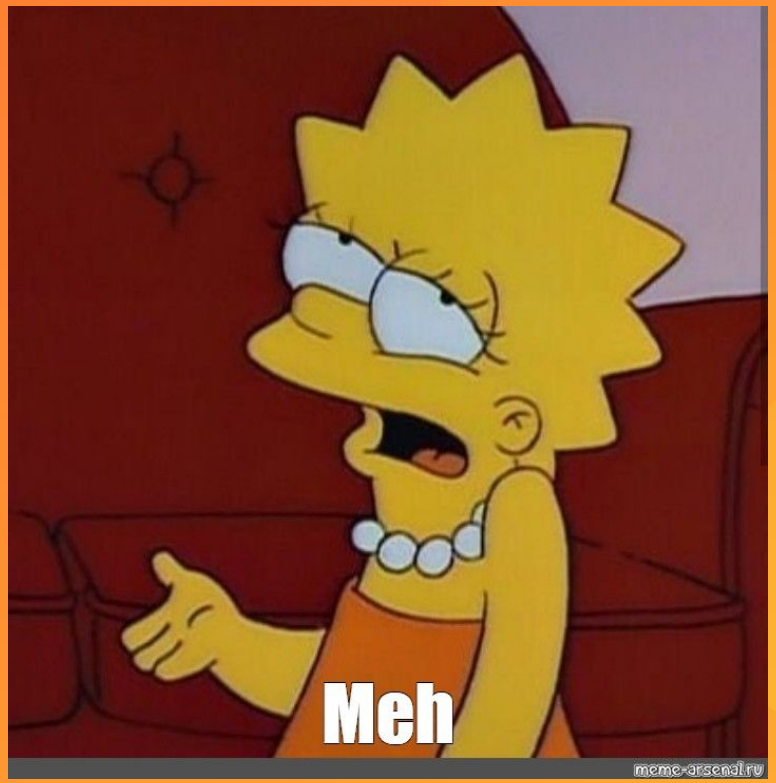


CLOUDFLARE

“

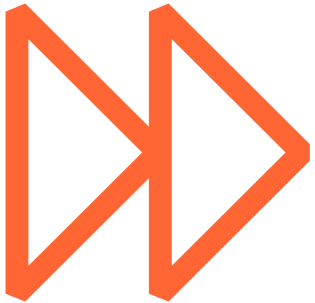
So the system admin can crash the system?

```
# echo t > /proc/sysrq-trigger
```



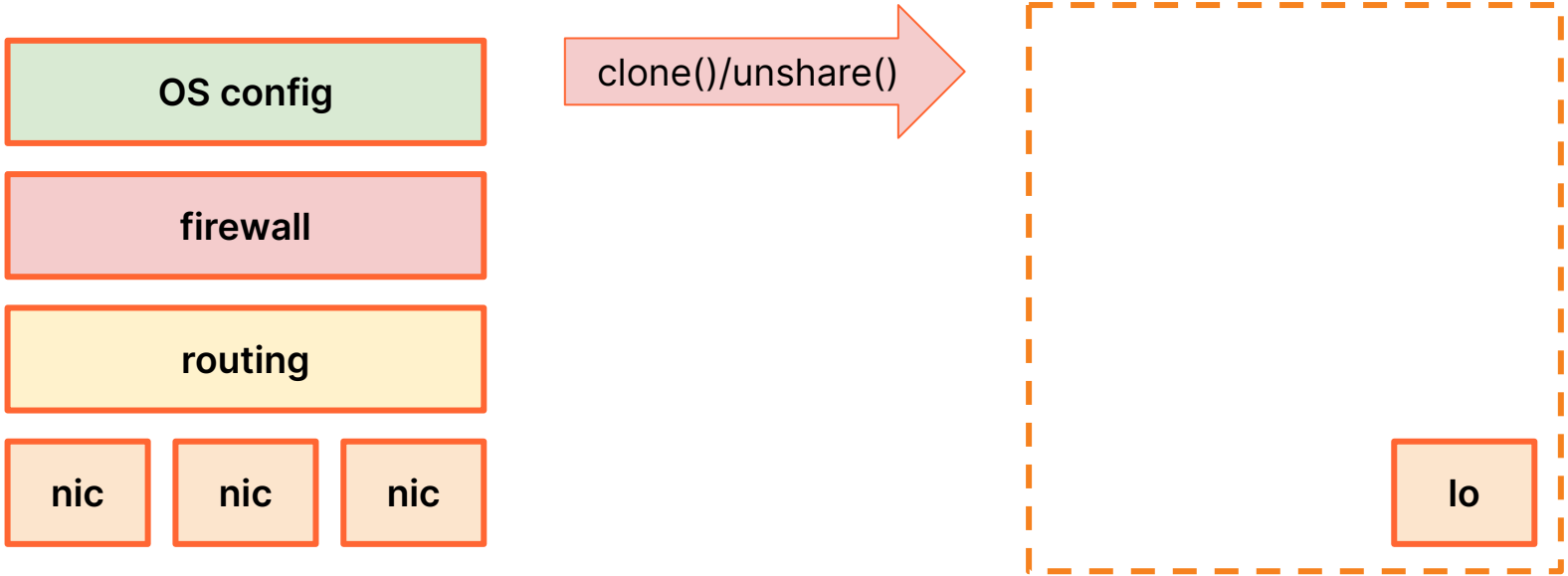
CLOUDFLARE

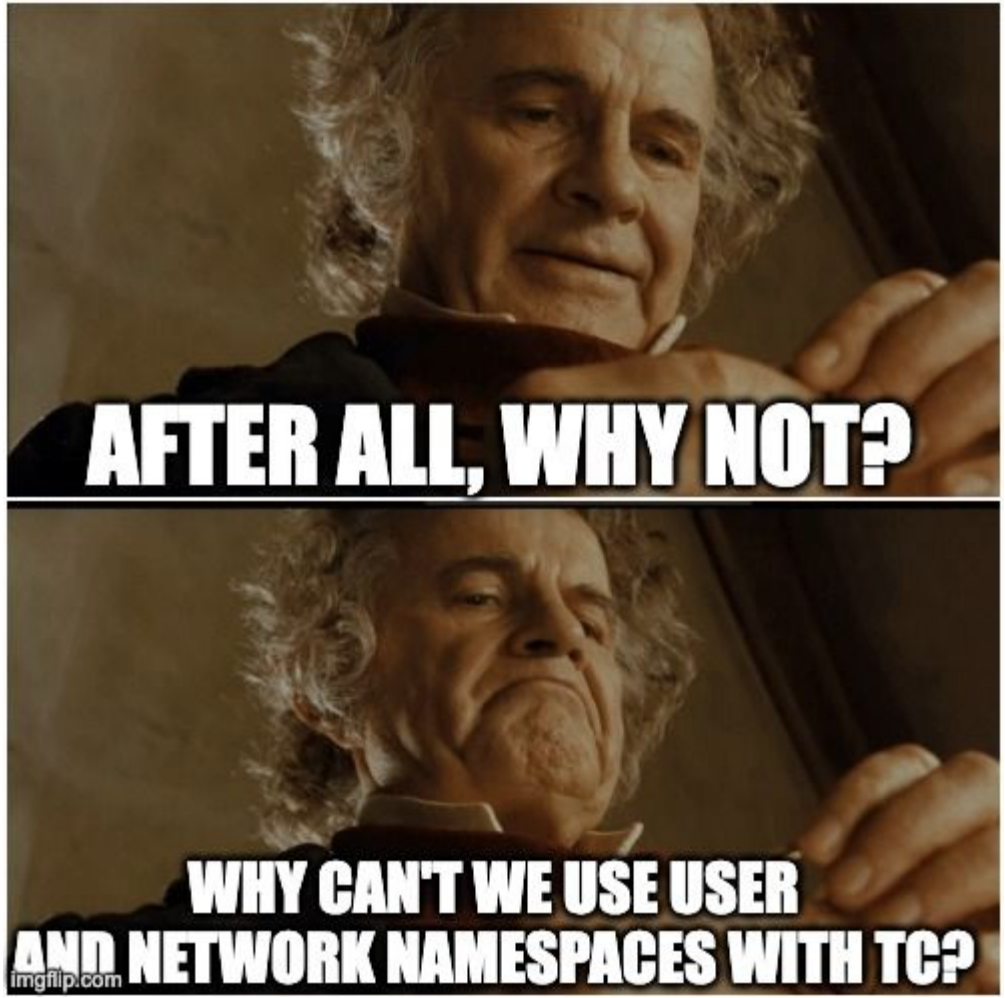
Fast forward to 2022



CLOUDFLARE

Remember network namespaces?




AFTER ALL, WHY NOT?
WHY CAN'T WE USE USER
AND NETWORK NAMESPACES WITH TC?
imgflip.com

CLOUDFLARE

Promoting the bug to a security vulnerability

```
ignat@buggy:~$
```

CLOUDFLARE

Promoting the bug to a security vulnerability

```
ignat@buggy:~$ uname -r
6.1.5-cloudflare-2023.1.4
ignat@buggy:~$
```

CLOUDFLARE

Promoting the bug to a security vulnerability

```
ignat@buggy:~$ uname -r
6.1.5-cloudflare-2023.1.4
ignat@buggy:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker),994(kvm)
ignat@buggy:~$
```

CLOUDFLARE

Promoting the bug to a security vulnerability

```
ignat@buggy:~$ uname -r
6.1.5-cloudflare-2023.1.4
ignat@buggy:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker),994(kvm)
ignat@buggy:~$ unshare --user --map-root-user --net
root@buggy:~#
```

CLOUDFLARE

Promoting the bug to a security vulnerability

```
ignat@buggy:~$ uname -r
6.1.5-cloudflare-2023.1.4
ignat@buggy:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker),994(kvm)
ignat@buggy:~$ unshare --user --map-root-user --net
root@buggy:~# ip link set lo up
root@buggy:~#
```

CLOUDFLARE

Promoting the bug to a security vulnerability

```
ignat@buggy:~$ uname -r
6.1.5-cloudflare-2023.1.4
ignat@buggy:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker),994(kvm)
ignat@buggy:~$ unshare --user --map-root-user --net
root@buggy:~# ip link set lo up
root@buggy:~# tc qdisc replace dev lo root handle 1: htb default 1
root@buggy:~#
```

CLOUDFLARE

Promoting the bug to a security vulnerability

```
ignat@buggy:~$ uname -r
6.1.5-cloudflare-2023.1.4
ignat@buggy:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker),994(kvm)
ignat@buggy:~$ unshare --user --map-root-user --net
root@buggy:~# ip link set lo up
root@buggy:~# tc qdisc replace dev lo root handle 1: htb default 1
root@buggy:~# tc class add dev lo parent 1: classid 1:1 htb rate 10mbit
root@buggy:~#
```

CLOUDFLARE

Promoting the bug to a security vulnerability

```
ignat@buggy:~$ uname -r
6.1.5-cloudflare-2023.1.4
ignat@buggy:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker),994(kvm)
ignat@buggy:~$ unshare --user --map-root-user --net
root@buggy:~# ip link set lo up
root@buggy:~# tc qdisc replace dev lo root handle 1: htb default 1
root@buggy:~# tc class add dev lo parent 1: classid 1:1 htb rate 10mbit
root@buggy:~# tc qdisc add dev lo parent 1:1 handle 10: noqueue
root@buggy:~#
```


GIVE ME A PING. ONE. PING. ONLY,
PLEASE.
ydrn.co

CLOUDFLARE

Promoting the bug to a security vulnerability

```
ignat@buggy:~$ uname -r
6.1.5-cloudflare-2023.1.4
ignat@buggy:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker),994(kvm)
ignat@buggy:~$ unshare --user --map-root-user --net
root@buggy:~# ip link set lo up
root@buggy:~# tc qdisc replace dev lo root handle 1: htb default 1
root@buggy:~# tc class add dev lo parent 1: classid 1:1 htb rate 10mbit
root@buggy:~# tc qdisc add dev lo parent 1:1 handle 10: noqueue
root@buggy:~#
```

CLOUDFLARE

Promoting the bug to a security vulnerability

```
ignat@buggy:~$ uname -r
6.1.5-cloudflare-2023.1.4
ignat@buggy:~$ id
uid=1000(ignat) gid=1000(ignat) groups=1000(ignat),990(docker),994(kvm)
ignat@buggy:~$ unshare --user --map-root-user --net
root@buggy:~# ip link set lo up
root@buggy:~# tc qdisc replace dev lo root handle 1: htb default 1
root@buggy:~# tc class add dev lo parent 1: classid 1:1 htb rate 10mbit
root@buggy:~# tc qdisc add dev lo parent 1:1 handle 10: noqueue
root@buggy:~# ping -I lo -w 1 -c 1 1.1.1.1
ping: Warning: source address might be selected on device other than: lo
PING 1.1.1.1 (1.1.1.1) from 0.0.0.0 lo: 56(84) bytes of data.
[  342.000820][  T412] BUG: kernel NULL pointer dereference, address: 0000000000000000
[  342.005797][  T412] #PF: supervisor instruction fetch in kernel mode
[  342.011184][  T412] #PF: error_code(0x0010) - not-present page
[  342.017450][  T412] PGD 0 P4D 0
[  342.021725][  T412] Oops: 0010 [#1] PREEMPT SMP NOPTI
[  342.026103][  T412] CPU: 0 PID: 412 Comm: ping Not tainted 6.1.5-cloudflare-2023.1.4 #1
[  342.030754][  T412] Hardware name: QEMU Standard PC (i440FX + PIIX, 1996), BIOS
1.16.2-debian-1.16.2-1 04/01/2014
...
```


WE ARE NOT ALONE

CLOUDFLARE

There is more...

- CVE-2023-30549
- CVE-2022-34918
- CVE-2022-25636
- CVE-2022-24122
- CVE-2022-1055
- CVE-2022-0185
- CVE-2022-0492
- CVE-2021-3493
- CVE-2020-5291
- CVE-2020-16120
- CVE-2019-20794
- CVE-2018-18955
- CVE-2017-1000111
- CVE-2014-5207
- CVE-2014-4014
- CVE-2013-1958

https://cve.mitre.org/cgi-bin/cvekey.cgi?keyword=%22user+namespaces%22

CLOUDFLARE

Why does it happen?

```
int some_kernel_func(void)
{
  if (!capable(CAP_SYS_ADMIN))
    return -EPERM;

  /* do some privileged stuff */
}
```

CLOUDFLARE

Why does it happen?

```
int some_kernel_func(void)
{
  if (!capable(CAP_SYS_ADMIN))
    return -EPERM;


  /* do some privileged stuff */
}
```

CLOUDFLARE

Why does it happen?

```
int some_kernel_func(void)
{
  if (!capable(CAP_SYS_ADMIN))
    return -EPERM;


  /* do some privileged stuff */
}
```

CLOUDFLARE

Why does it happen?

```
int some_kernel_func(void)
{
  if (!capable(CAP_SYS_ADMIN))
    return -EPERM;

  /* do some privileged stuff */
}
```

may contain bugs

@ignatkn

CLOUDFLARE

Taming unprivileged user namespaces

CLOUDFLARE

Disable user namespaces

CONFIG_USER_NS=n

CLOUDFLARE

Disable user namespaces

CONFIG_USER_NS=n

```
ignat@dev:~$
```

CLOUDFLARE

Disable user namespaces

CONFIG_USER_NS=n

```
ignat@dev:~$ sudo sysctl user.max_user_namespaces
user.max_user_namespaces = 256436
ignat@dev:~$
```

CLOUDFLARE

Disable user namespaces

CONFIG_USER_NS=n

```
ignat@dev:~$ sudo sysctl user.max_user_namespaces
user.max_user_namespaces = 256436
ignat@dev:~$ unshare --user
nobody@dev:~$
```

CLOUDFLARE

Disable user namespaces

CONFIG_USER_NS=n

```
ignat@dev:~$ sudo sysctl user.max_user_namespaces
user.max_user_namespaces = 256436
ignat@dev:~$ unshare --user
nobody@dev:~$ exit
logout
ignat@dev:~$
```

CLOUDFLARE

Disable user namespaces

CONFIG_USER_NS=n

```
ignat@dev:~$ sudo sysctl user.max_user_namespaces
user.max_user_namespaces = 256436
ignat@dev:~$ unshare --user
nobody@dev:~$ exit
logout
ignat@dev:~$ sudo sysctl -w user.max_user_namespaces=0
user.max_user_namespaces = 0
ignat@dev:~$
```

CLOUDFLARE

Disable user namespaces

CONFIG_USER_NS=n

```
ignat@dev:~$ sudo sysctl user.max_user_namespaces
user.max_user_namespaces = 256436
ignat@dev:~$ unshare --user
nobody@dev:~$ exit
logout
ignat@dev:~$ sudo sysctl -w user.max_user_namespaces=0
user.max_user_namespaces = 0
ignat@dev:~$ unshare --user
unshare: unshare failed: No space left on device
```

CLOUDFLARE

Disable user namespaces

CLOUDFLARE

Making unprivileged user namespaces privileged again

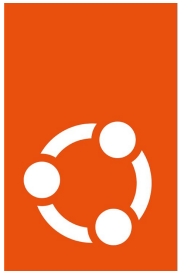
```
$ sudo sysctl -w kernel.unprivileged_userns_clone=0
```

CLOUDFLARE

Making unprivileged user namespaces privileged again

```
$ sudo sysctl -w kernel.unprivileged_userns_clone=0
```

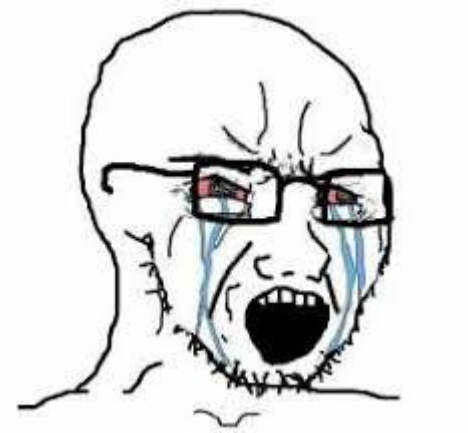
debian



Ubuntu

CLOUDFLARE

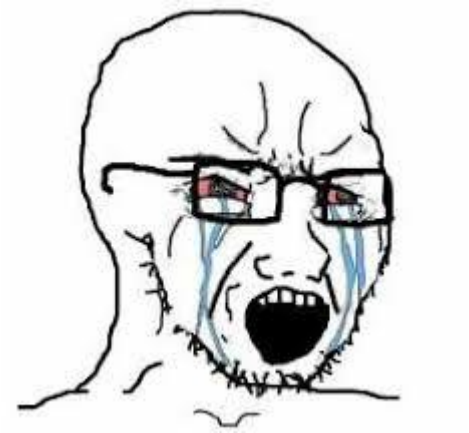
Making unprivileged user namespaces privileged again



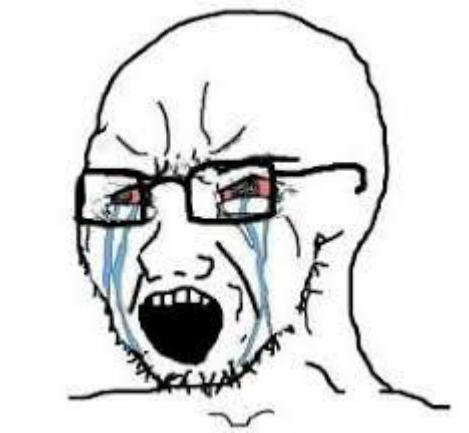
We need to disable unprivileged user namespaces, because they create more vulnerabilities.

CLOUDFLARE

Making unprivileged user namespaces privileged again



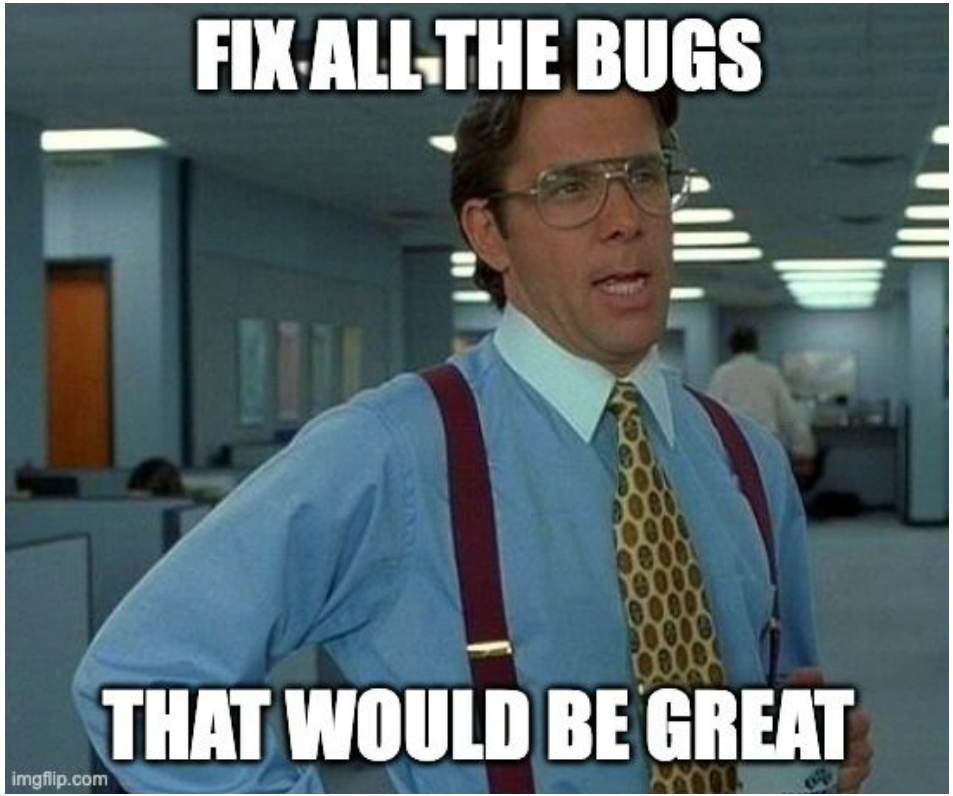
We need to disable unprivileged user namespaces, because they create more vulnerabilities.



We need to keep unprivileged user namespaces, because they allow for more secure designs.

CLOUDFLARE

Just fix all the bugs!



CLOUDFLARE

Making unprivileged user namespaces privileged again

Disable?

CLOUDFLARE

Making unprivileged user namespaces privileged again

Disable?

Keep?

CLOUDFLARE

Making unprivileged user namespaces privileged again

Disable?

Keep?

Let’s do both!

CLOUDFLARE

How do we allow some applications to create user namespaces, while prohibiting the others?

CLOUDFLARE

How do we allow some applications to create user namespaces, while prohibiting the others?

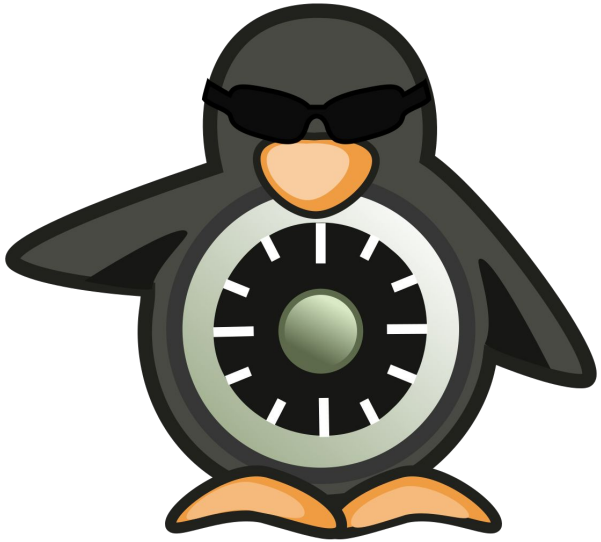
Policy

CLOUDFLARE

How do we allow some applications to create user namespaces, while prohibiting the others?

Policy = LSM

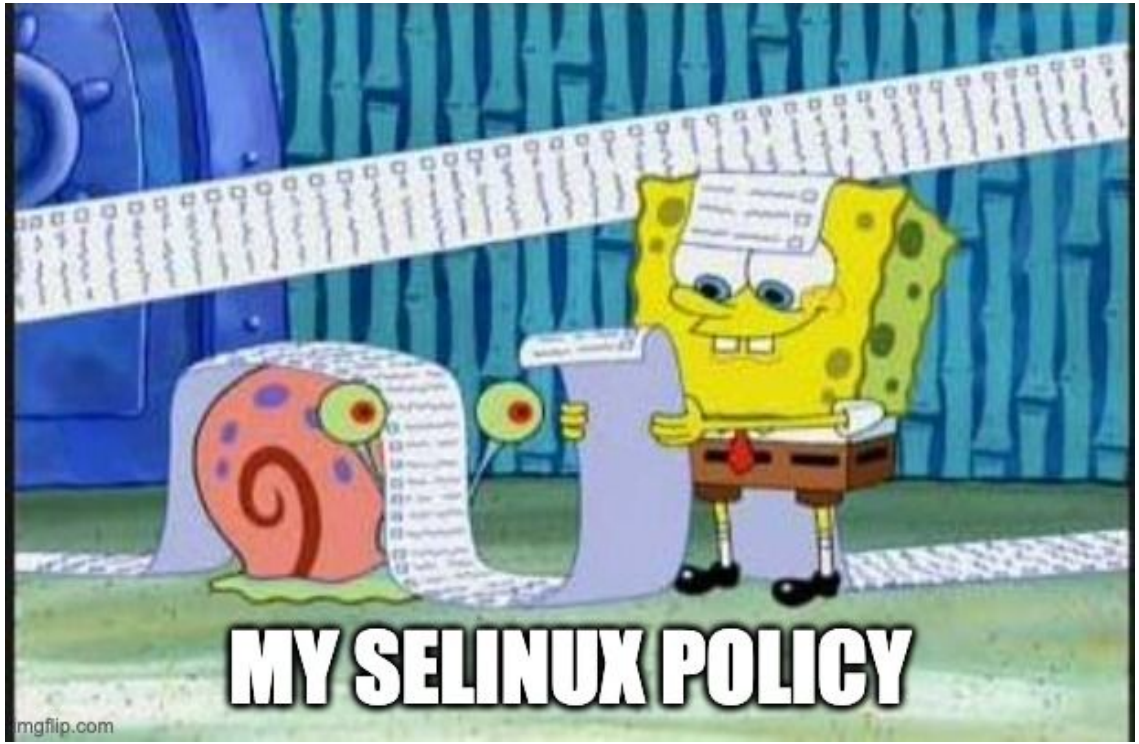
SELinux?



SELinux

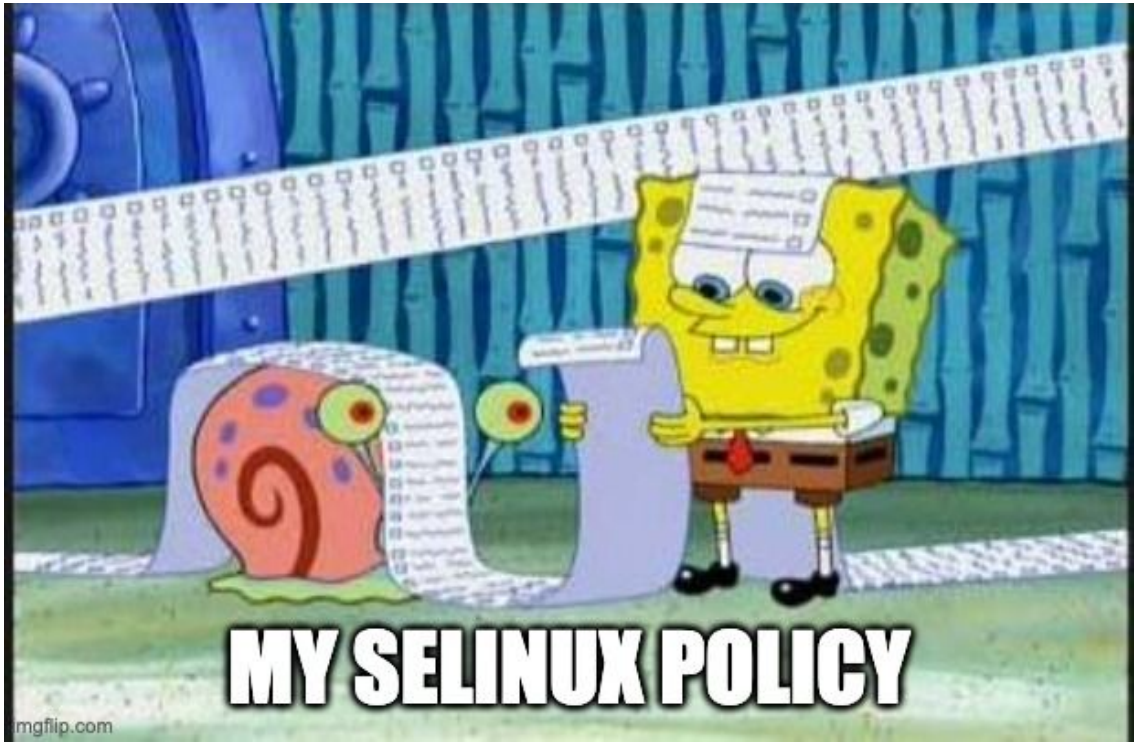
CLOUDFLARE

SELinux?



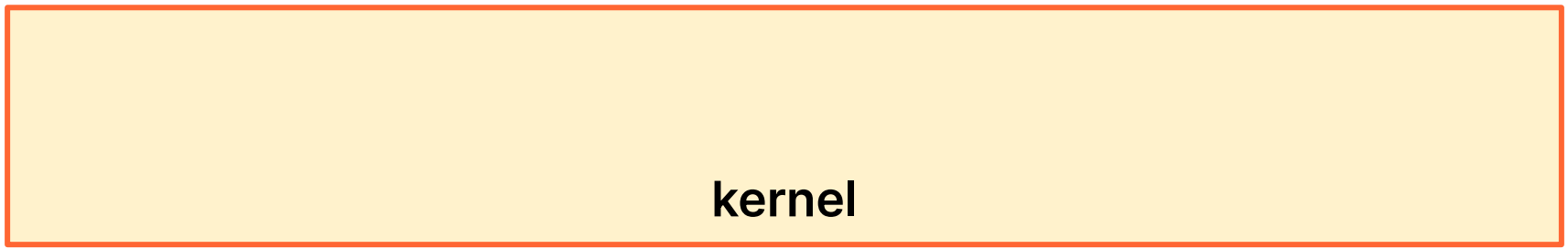
CLOUDFLARE

SELinux?



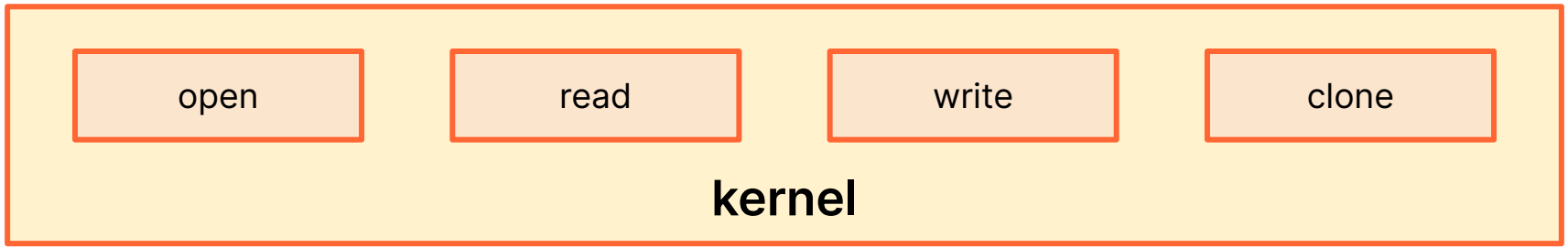
CLOUDFLARE

BPF-LSM



CLOUDFLARE

BPF-LSM

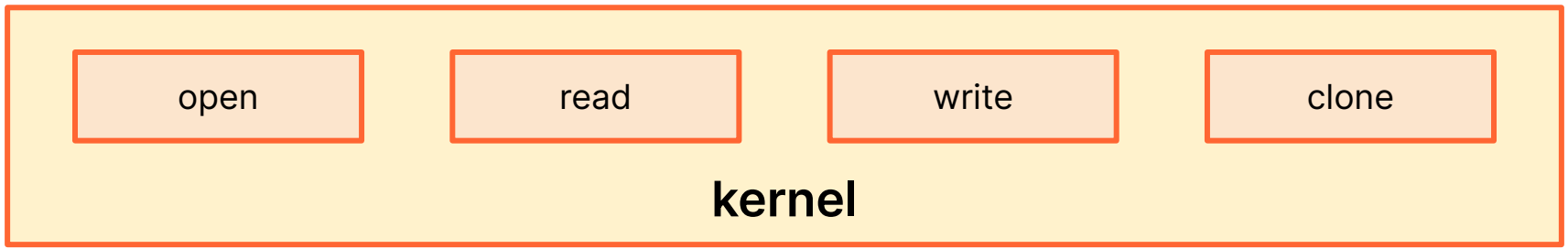


CLOUDFLARE

BPF-LSM

```
int check_xxx(void)
{
    if !allowed(ctx) {
        return -EPERM;
    }

    return 0;
}
```

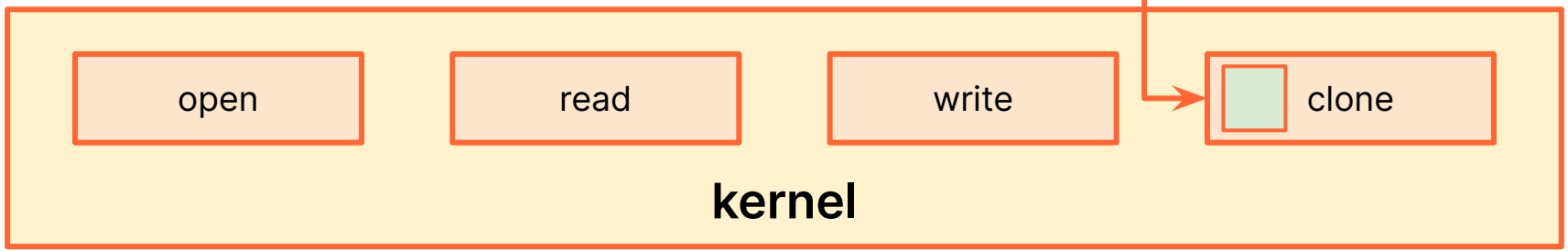


CLOUDFLARE

BPF-LSM

```
int check_xxx(void)
{
    if !allowed(ctx) {
        return -EPERM;
    }

    return 0;
}
```

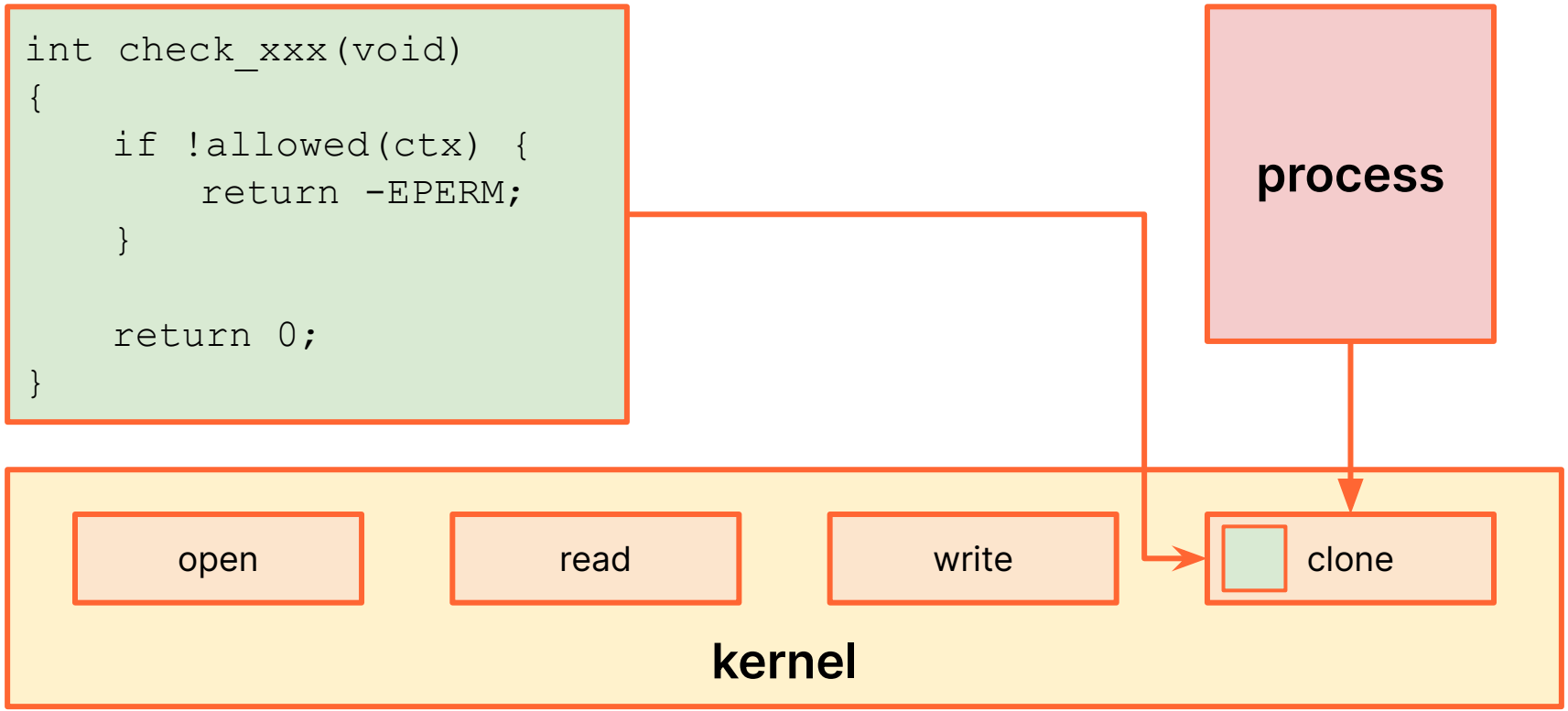


CLOUDFLARE

BPF-LSM

```
int check_xxx(void)
{
    if !allowed(ctx) {
        return -EPERM;
    }

    return 0;
}
```



CLOUDFLARE

Selecting the LSM hook: cred_prepare

https://elixir.bootlin.com/linux/v5.15.137/source/security/security.c#L1712

CLOUDFLARE

Selecting the LSM hook: cred_prepare

https://elixir.bootlin.com/linux/v5.15.137/source/security/security.c#L1712

```
ignat@dev:~$ unshare --user
unshare: unshare failed: Cannot allocate memory
```

CLOUDFLARE

Selecting the LSM hook: cred_prepare

https://elixir.bootlin.com/linux/v5.15.137/source/security/security.c#L1712

```
ignat@dev:~$ unshare --user
unshare: unshare failed: Cannot allocate memory
```

```
int unshare_userns(unsigned long unshare_flags, struct cred **new_cred)
{
        struct cred *cred;
        int err = -ENOMEM;

        if (!(unshare_flags & CLONE_NEWUSER))
                return 0;

        cred = prepare_creds();
        if (cred) {
                err = create_user_ns(cred);
                if (err)
                        put_cred(cred);
                else
                        *new_cred = cred;
        }

        return err;
}
```

CLOUDFLARE

Selecting the LSM hook: cred_prepare

https://elixir.bootlin.com/linux/v5.15.137/source/security/security.c#L1712

```
ignat@dev:~$ unshare --user
unshare: unshare failed: Cannot allocate memory
```

```
int unshare_userns(unsigned long unshare_flags, struct cred **new_cred)
{
        struct cred *cred;
        int err = -ENOMEM;

        if (!(unshare_flags & CLONE_NEWUSER))
                return 0;

        cred = prepare_creds();
        if (cred) {
                err = create_user_ns(cred);
                if (err)
                        put_cred(cred);
                else
                        *new_cred = cred;
        }

        return err;
}
```

CLOUDFLARE

We need a new hook

https://lwn.net/Articles/903580/

CLOUDFLARE

We need a new hook

We need to keep unprivileged user namespaces, because they allow for more secure designs.

https://lwn.net/Articles/903580/

CLOUDFLARE

We need a new hook

```
You aren't fixing what your problem you are papering over it by deny
access to the user namespace.

Nack Nack Nack.

Stop.

Go back to the drawing board.

Do not pass go.

Do not collect $200.
```

We need to keep unprivileged user namespaces, because they allow for more secure designs.

https://lwn.net/Articles/903580/

CLOUDFLARE

Linux kernel: biggest free open source project

CLOUDFLARE

Linux kernel: biggest free open source project

CLOUDFLARE

Linux kernel: biggest free open source project

CLOUDFLARE

Linux kernel: biggest free open source project

> And I think you are in denial about how many problems the
> user-namespace stuff has caused.
>
> Distros are literally turning it off entirely because the whole "let
> users create their own namespace" has *NOT* been a great success.
>
> I personally think it was a mistake. We're stuck with it, but we most
> definitely need knobs to manage it that isn't just "enable/disable
> USER_NS" in the kernel config.
>
> So this whole "don't do this" approach you have is not acceptable.
>
> 99% of all code does NOT WANT the user namespace thing, and it's been
> a big new attack surface for the kernel getting things subtly wrong.
>
> I do not understand your "people need to be able to do this with no
> controls", when the alternative is to literally turn it off ENTIRELY.
>
> I'm not saying that an LSM is the only place to do it, but I don't
> think there have been any better suggestions either.
>
> Put another way: your "no limits are acceptable" is simply not
> realistic, and you haven't given any sane alternatives that I am aware
> of. No way to say "sure, let trusted system apps create their
> namespaces, but not random things".
>
> Linus

https://www.spinics.net/lists/linux-security-module/msg48941.html

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ cat userns.c
#include <linux/bpf.h>
#include <bpf/bpf_tracing.h>
#include <bpf/bpf_helpers.h>
#include <errno.h>

char LICENSE[] SEC("license") = "GPL";

struct cred {
};

SEC("lsm/userns_create")
int BPF_PROG(check_userns_create, struct cred *cred, int ret)
{
    char comm[16];
    bpf_get_current_comm(comm, sizeof(comm));
    if (comm[0] == 't')
        return 0;

    return -EPERM;
}
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ cat userns.c
#include <linux/bpf.h>
#include <bpf/bpf_tracing.h>
#include <bpf/bpf_helpers.h>
#include <errno.h>

char LICENSE[] SEC("license") = "GPL";

struct cred {
};

SEC("lsm/userns_create")
int BPF_PROG(check_userns_create, struct cred *cred, int ret)
{
    char comm[16];
    bpf_get_current_comm(comm, sizeof(comm));
    if (comm[0] == 't')
        return 0;

    return -EPERM;
}
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ cat userns.c
#include <linux/bpf.h>
#include <bpf/bpf_tracing.h>
#include <bpf/bpf_helpers.h>
#include <errno.h>

char LICENSE[] SEC("license") = "GPL";

struct cred {
};

SEC("lsm/userns_create")
int BPF_PROG(check_userns_create, struct cred *cred, int ret)
{
    char comm[16];
    bpf_get_current_comm(comm, sizeof(comm));
    if (comm[0] == 't')
        return 0;

    return -EPERM;
}
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ cat userns.c
#include <linux/bpf.h>
#include <bpf/bpf_tracing.h>
#include <bpf/bpf_helpers.h>
#include <errno.h>

char LICENSE[] SEC("license") = "GPL";

struct cred {
};

SEC("lsm/userns_create")
int BPF_PROG(check_userns_create, struct cred *cred, int ret)
{
    char comm[16];
    bpf_get_current_comm(comm, sizeof(comm));
    if (comm[0] == 't')
        return 0;

    return -EPERM;
}
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ clang -O2 -target bpf -I/usr/include
-I/usr/include/x86_64-linux-gnu -c userns.c -o userns.o
ignat@dev:~$
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ clang -O2 -target bpf -I/usr/include
-I/usr/include/x86_64-linux-gnu -c userns.c -o userns.o
ignat@dev:~$ /sbin/bpftool gen skeleton userns.o >
userns.skel.h
ignat@dev:~$
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ cat load.c
#include "userns.skel.h"

int main(void)
{
    struct userns *skel = userns__open_and_load();
    if (!skel)
        return 1;

    userns__attach(skel);
    getchar();

    return 0;
}
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ cat load.c
#include "userns.skel.h"

int main(void)
{
    struct userns *skel = userns__open_and_load();
    if (!skel)
        return 1;

    userns__attach(skel);
    getchar();

    return 0;
}
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ cat load.c
#include "userns.skel.h"

int main(void)
{
    struct userns *skel = userns__open_and_load();
    if (!skel)
        return 1;

    userns__attach(skel);
    getchar();

    return 0;
}
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ clang -O2 -target bpf -I/usr/include
-I/usr/include/x86_64-linux-gnu -c userns.c -o userns.o
ignat@dev:~$ /sbin/bpftool gen skeleton userns.o >
userns.skel.h
ignat@dev:~$
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ clang -O2 -target bpf -I/usr/include
-I/usr/include/x86_64-linux-gnu -c userns.c -o userns.o
ignat@dev:~$ /sbin/bpftool gen skeleton userns.o >
userns.skel.h
ignat@dev:~$ gcc -o load load.c -lbpf
ignat@dev:~$
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ clang -O2 -target bpf -I/usr/include
-I/usr/include/x86_64-linux-gnu -c userns.c -o userns.o
ignat@dev:~$ /sbin/bpftool gen skeleton userns.o >
userns.skel.h
ignat@dev:~$ gcc -o load load.c -lbpf
ignat@dev:~$ sudo ./load
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ clang -O2 -target bpf -I/usr/include
-I/usr/include/x86_64-linux-gnu -c userns.c -o userns.o
ignat@dev:~$ /sbin/bpftool gen skeleton userns.o >
userns.skel.h
ignat@dev:~$ gcc -o load load.c -lbpf
ignat@dev:~$ sudo ./load
```

```
ignat@dev:~$
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ clang -O2 -target bpf -I/usr/include
-I/usr/include/x86_64-linux-gnu -c userns.c -o userns.o
ignat@dev:~$ /sbin/bpftool gen skeleton userns.o >
userns.skel.h
ignat@dev:~$ gcc -o load load.c -lbpf
ignat@dev:~$ sudo ./load
```

```
ignat@dev:~$ unshare --user --map-root-user
unshare: unshare failed: Operation not permitted
ignat@dev:~$
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ clang -O2 -target bpf -I/usr/include
-I/usr/include/x86_64-linux-gnu -c userns.c -o userns.o
ignat@dev:~$ /sbin/bpftool gen skeleton userns.o >
userns.skel.h
ignat@dev:~$ gcc -o load load.c -lbpf
ignat@dev:~$ sudo ./load
```

```
ignat@dev:~$ unshare --user --map-root-user
unshare: unshare failed: Operation not permitted
ignat@dev:~$ cp /usr/bin/unshare ./trusted
ignat@dev:~$
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ clang -O2 -target bpf -I/usr/include
-I/usr/include/x86_64-linux-gnu -c userns.c -o userns.o
ignat@dev:~$ /sbin/bpftool gen skeleton userns.o >
userns.skel.h
ignat@dev:~$ gcc -o load load.c -lbpf
ignat@dev:~$ sudo ./load
```

```
ignat@dev:~$ unshare --user --map-root-user
unshare: unshare failed: Operation not permitted
ignat@dev:~$ cp /usr/bin/unshare ./trusted
ignat@dev:~$ ./trusted --user --map-root-user
root@dev:~#
```

CLOUDFLARE

Using the new userns_create hook from Linux 6.1

```
ignat@dev:~$ clang -O2 -target bpf -I/usr/include
-I/usr/include/x86_64-linux-gnu -c userns.c -o userns.o
ignat@dev:~$ /sbin/bpftool gen skeleton userns.o >
userns.skel.h
ignat@dev:~$ gcc -o load load.c -lbpf
ignat@dev:~$ sudo ./load
```

```
ignat@dev:~$ unshare --user --map-root-user
unshare: unshare failed: Operation not permitted
ignat@dev:~$ cp /usr/bin/unshare ./trusted
ignat@dev:~$ ./trusted --user --map-root-user
root@dev:~# exit
logout
ignat@dev:~$
```

CLOUDFLARE

Conclusions

CLOUDFLARE

Links

CLOUDFLARE

Thank you

Questions?