Exploration of Linux Container Network Monitoring and Visualization

Alban Crequy

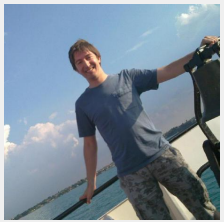
https://goo.gl/iDL8te

ContainerCon Europe - October 2016

kinvolk.io

Alban Crequy

- Worked on the rkt container run-time
- Contributed to systemd



https://github.com/alban

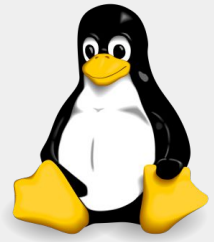
kinvolk.io

Berlin-based software company building *foundational* Linux technologies

Some examples of what we work on...

rkt

systemd
The Linux System & Service Manager



OSTree
git for operating system binaries

Find out more about us...

Blog: https://kinvolk.io/blog

Github: https://github.com/kinvolk

Twitter: https://twitter.com/kinvolkio

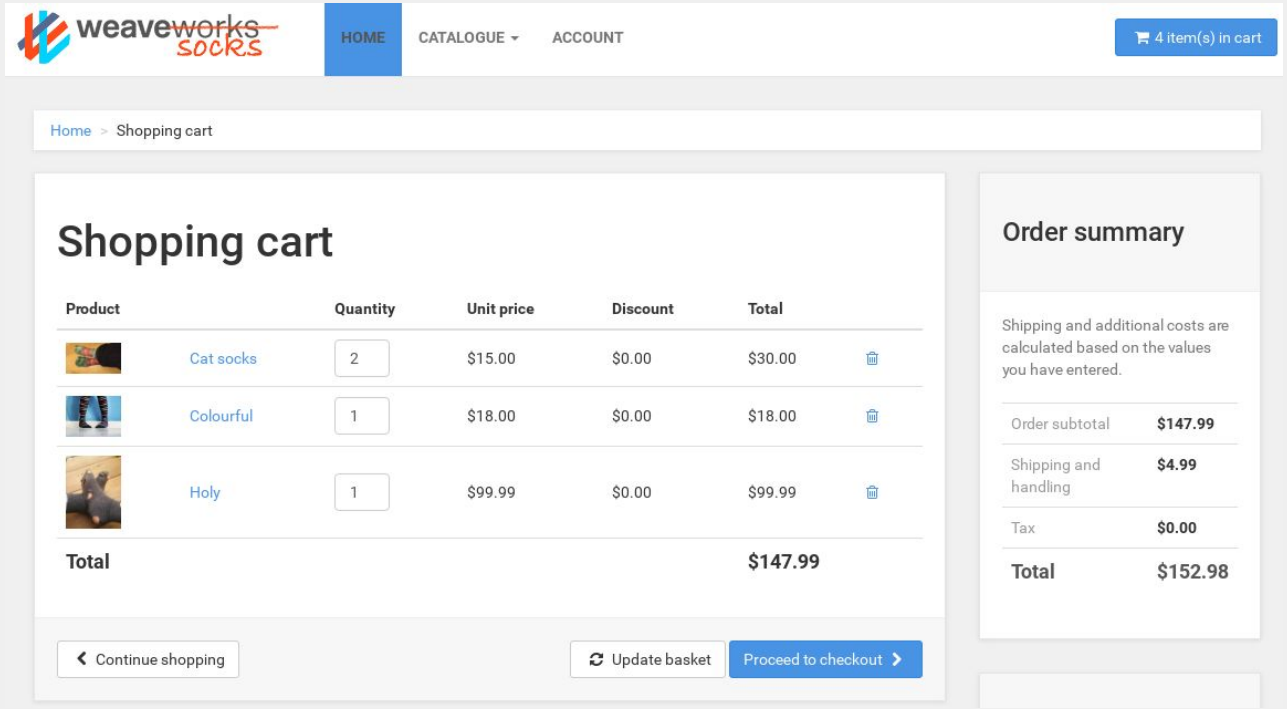
Email: hello@kinvolk.io

Plan

- First use-case: visualizing tcp connections
  - Microservices application with containers: Weave Socks
  - CoreOS Linux, Kubernetes, Weave Scope
- Using /proc & conntrack
  - Limitations
  - proc connector, eBPF & kprobes
- Next use cases:
  - L7, HTTP: eBPF & kprobes
  - Simulating degraded networks with traffic control

The demo application

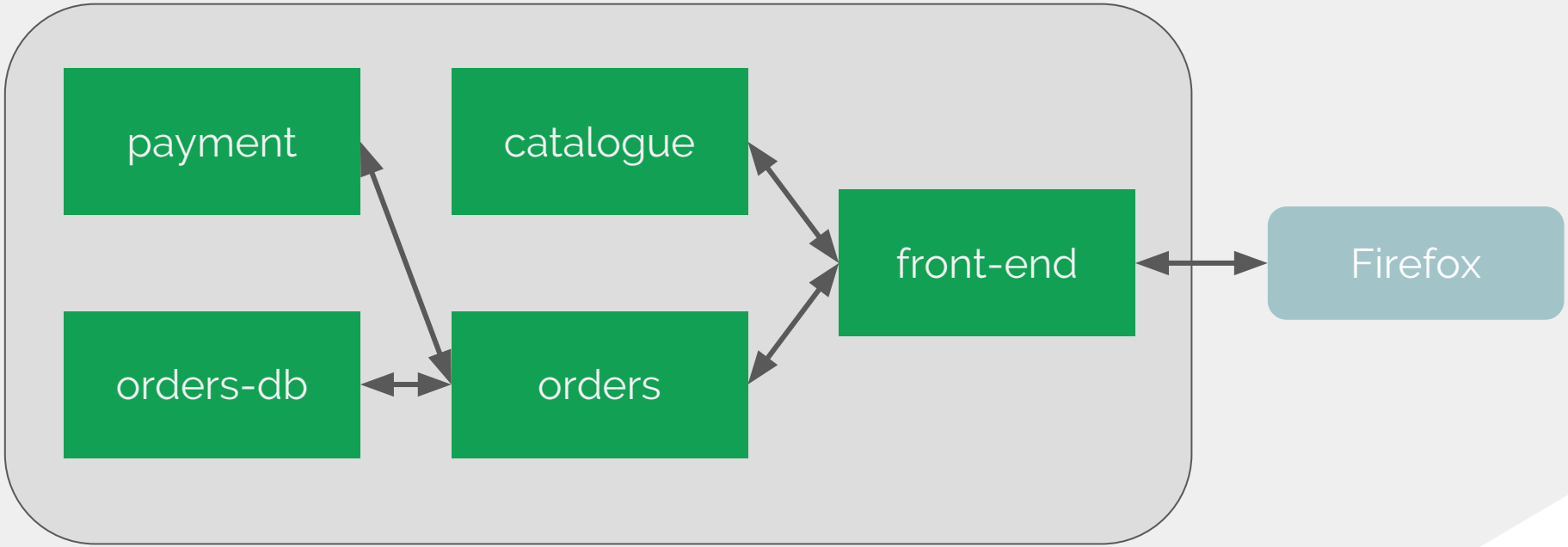
microservices-demo

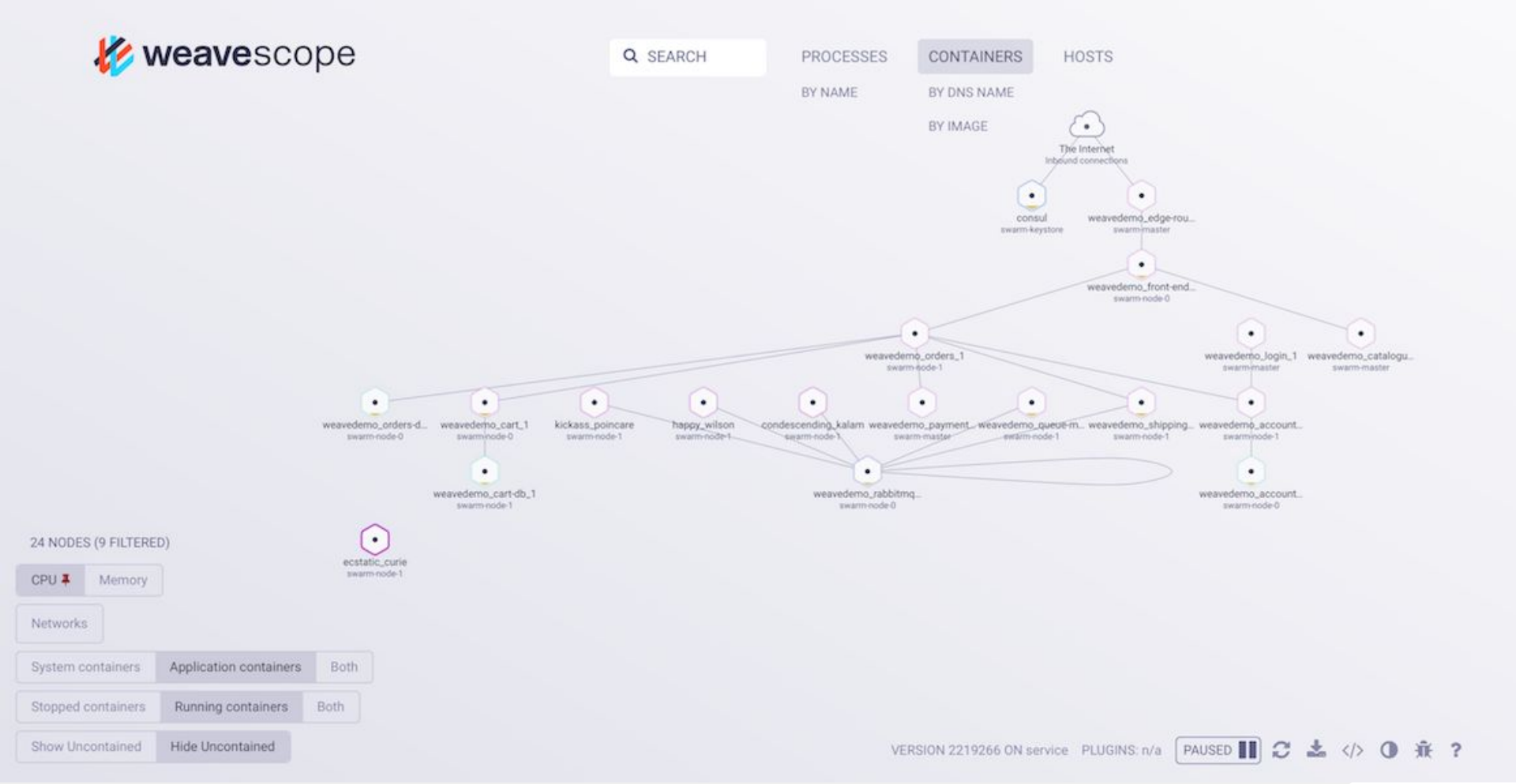


https://github.com/microservices-demo/microservices-demo

kinvolk.io

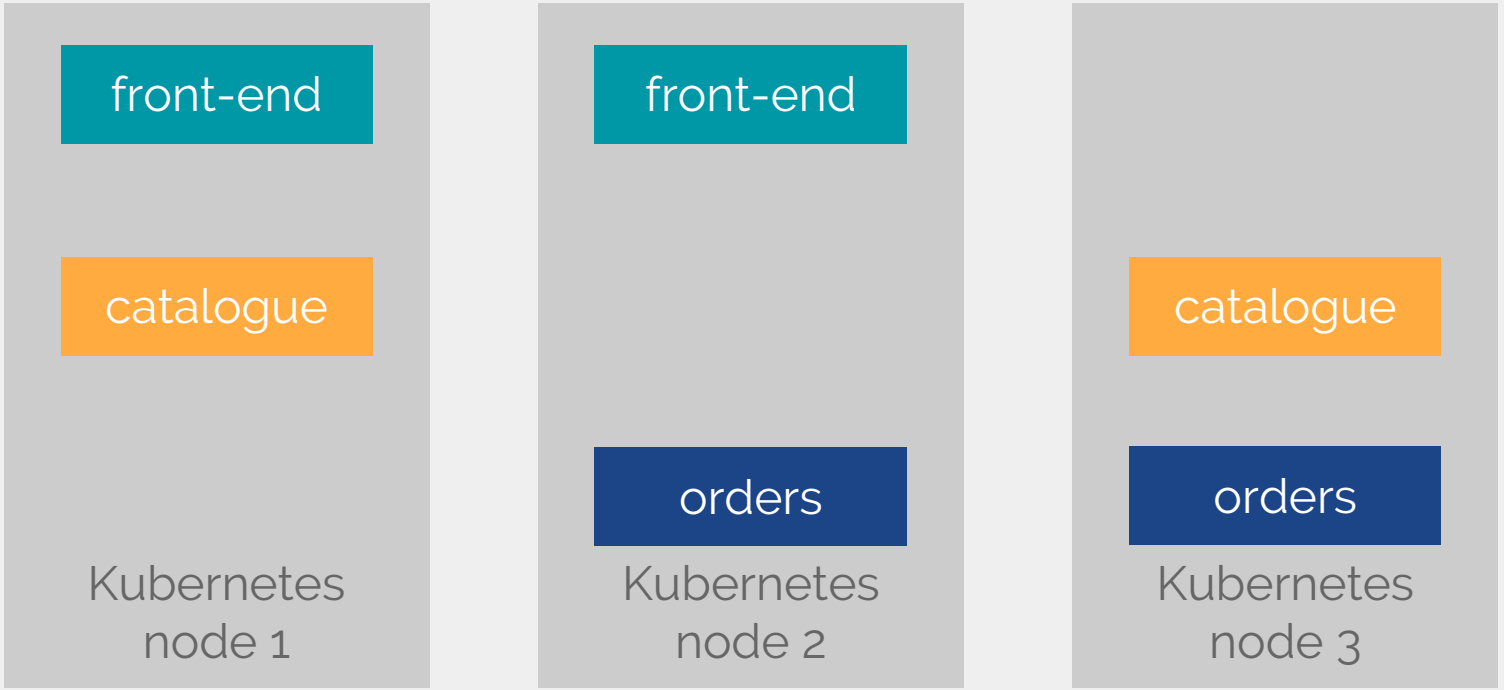
Some micro-services



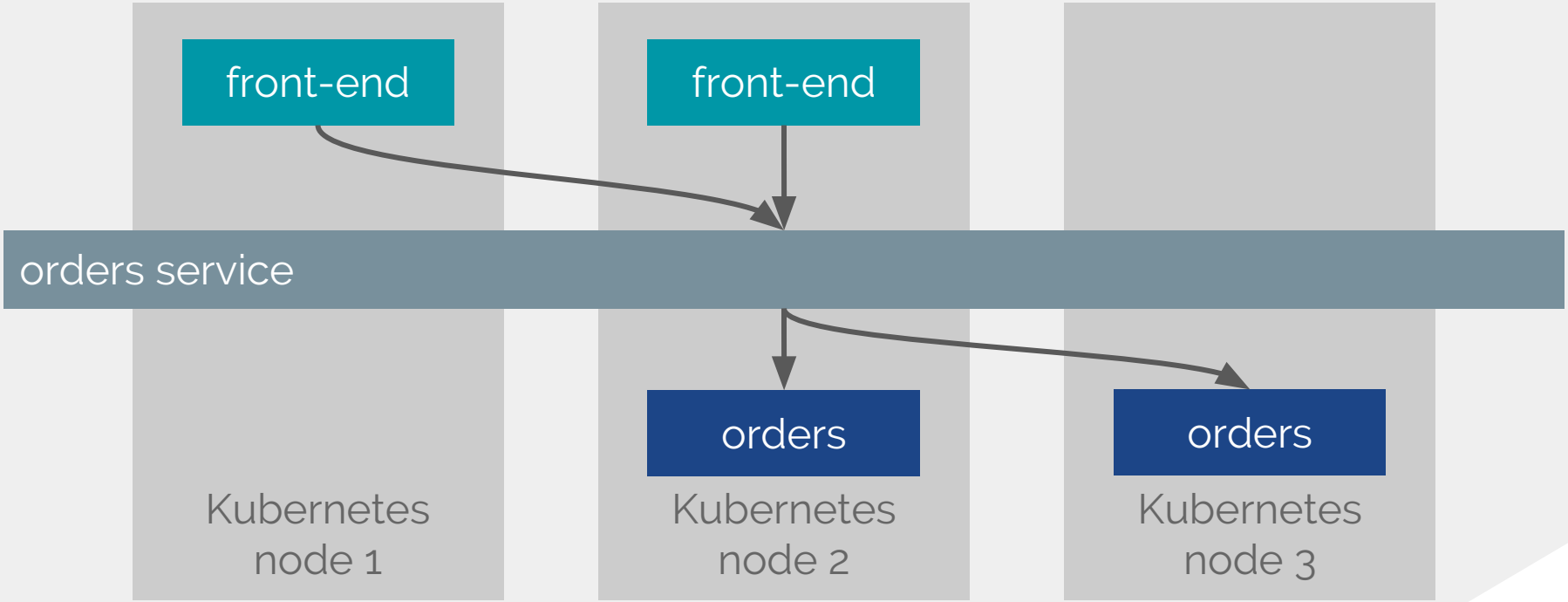

weavescope
SEARCH
PROCESSES
CONTAINERS
HOSTS
BY NAME
BY DNS NAME
BY IMAGE
The Internet
Inbound connections
consul
swarm-keystore
weavedemo_edge-rou...
swarm-master
weavedemo_front-end...
swarm-node-0
weavedemo_orders_1
swarm-node-1
weavedemo_login_1
swarm-master
weavedemo_catalogu...
swarm-master
weavedemo_orders-d...
swarm-node-0
weavedemo_cart_1
swarm-node-0
kickass_poincare
swarm-node-1
happy_wilson
swarm-node-1
condescending_kalam
swarm-node-1
weavedemo_payment...
swarm-master
weavedemo_queue-m...
swarm-node-1
weavedemo_shipping...
swarm-node-1
weavedemo_account...
swarm-node-1
weavedemo_cart-db_1
swarm-node-1
weavedemo_rabbitmq...
swarm-node-0
weavedemo_account...
swarm-node-0
ecstatic_curie
swarm-node-1
24 NODES (9 FILTERED)
CPU
Memory
Networks
System containers
Application containers
Both
Stopped containers
Running containers
Both
Show Uncontained
Hide Uncontained
VERSION 2219266 ON service
PLUGINS: n/a
PAUSED

Orchestrating containers With Kubernetes

Kubernetes Replica Sets



Kubernetes Services



Kubernetes Services

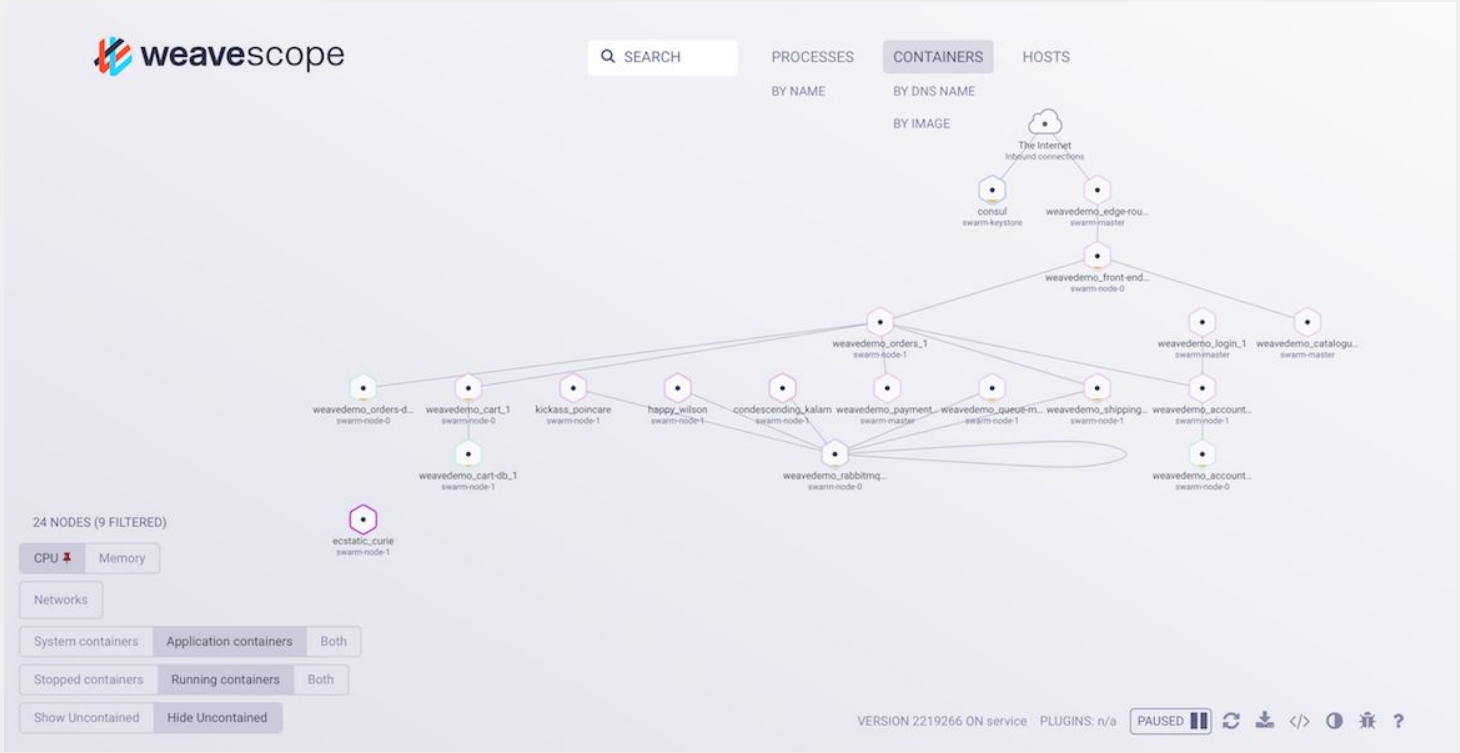
Proxying the traffic from the virtual service IP to a Kubernetes pod

Several implementations possible:

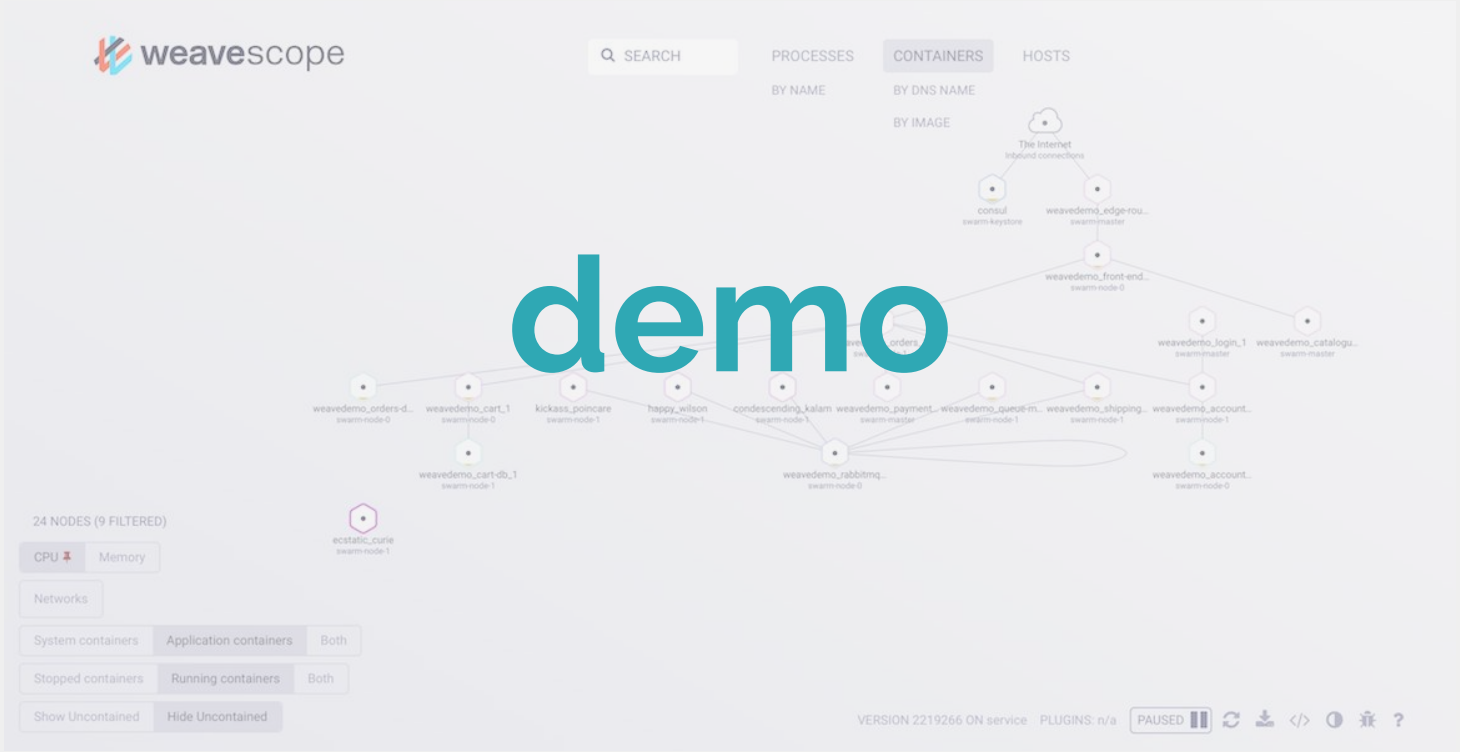
- Userspace proxy in kube-proxy
- Iptables rules (Destination NAT) installed by kube-proxy
- Cilium implements a load balancer based on eBPF (tc level)

Weave Scope

Weave Scope



Weave Scope



demo

kinvolk.io

procfs

procfs files

- /proc/$PID
- /proc/$PID/ns/net — network namespace
- /proc/$PID/fd/ — file descriptors
- /proc/$PID/net/tcp — tcp connections

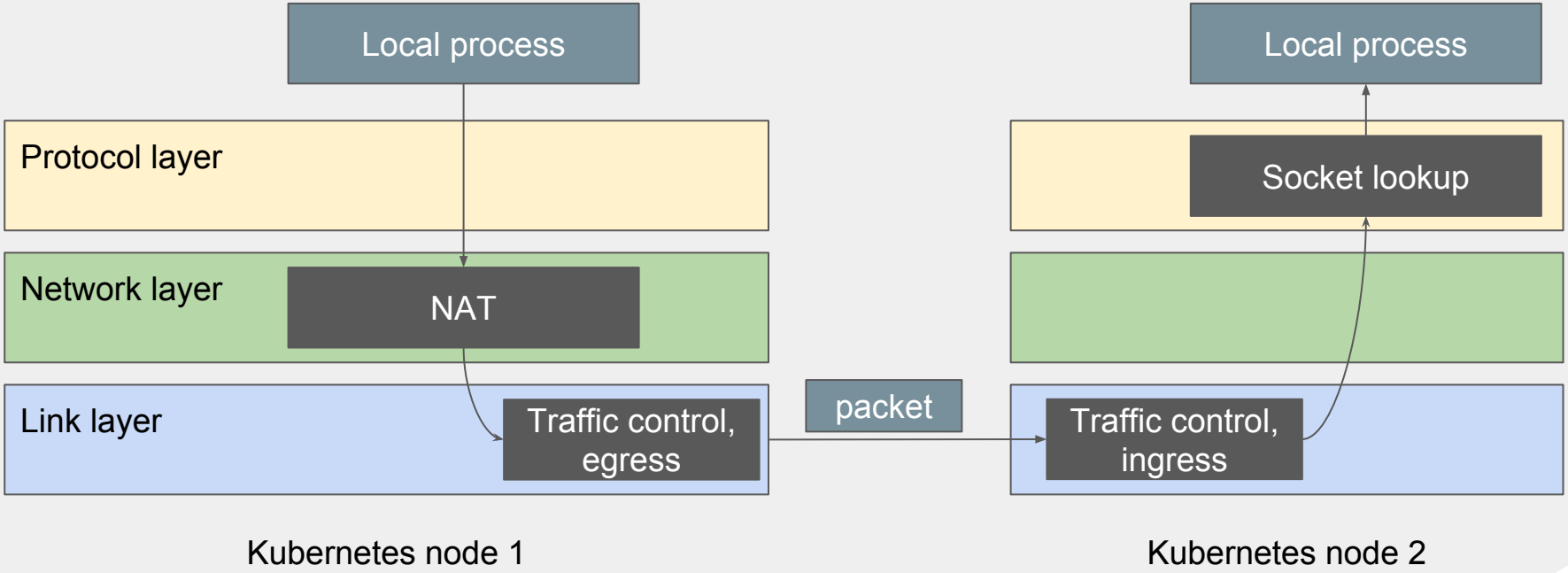
procfs files

```
$ nc 85.239.127.90 80


$ ls -l /proc/$(pidof nc)/ns/net
lrwxrwxrwx. 1 alban alban 0 Oct  2 16:42 /proc/25343/ns/net -> 'net:[4026531969]'


$ ls -l /proc/$(pidof nc)/fd/
total 0
lrwx------. 1 alban alban 64 Oct  2 16:35 0 -> /dev/pts/2
lrwx------. 1 alban alban 64 Oct  2 16:35 2 -> /dev/pts/2
lrwx------. 1 alban alban 64 Oct  2 16:35 3 -> 'socket:[8707290]'

$ cat  /proc/$(pidof nc)/net/tcp
  sl  local_address rem_address   st tx_queue rx_queue tr tm->when retrnsmt   uid  timeout inode
  25: 0C00A8C0:984A 5A7FEF55:0050 01 00000000:00000000 00:00000000 00000000  1000        0 8707290 1 ffff88020286e300 23 0 0 10 -1

$ printf '%02X' 90 127 239 85 ; echo -n ':' ; printf '%04X' 80 ; echo
5A7FEF55:0050
```

procfs limitations

- No notifications
- Need to read procfs for
  - new processes
  - new network namespaces
  - new sockets
  - every second?
- CPU intensive for systems with high number of processes
- Missing short-lived connections
- Issues with packet modifications (e.g. DNAT)

Packet modifications



Netlink

Netlink sockets

```
socket(AF_NETLINK, SOCK_RAW, NETLINK_...);
```

Several Netlink sockets:

- NETLINK_ROUTE
- NETLINK_INET_DIAG
- NETLINK_SELINUX
- NETLINK_CONNECTOR
- NETLINK_NETFILTER
- ...

conntrack

conntrack -E

- Use NETLINK_NETFILTER sockets to subscribe to Conntrack events from the kernel
- Is aware of NAT rewritings

```
$ nc 85.239.127.90 80


$ conntrack -E -p tcp
 [UPDATE] tcp      6 432000 ESTABLISHED src=172.17.0.2 dst=85.239.127.90 sport=33874 dport=80 src=85.239.127.90 dst=192.168.0.12 sport=80 dport=33874
[ASSURED]
```

conntrack limitations

- Conntrack events don't include:
  - Process ID
  - Network namespace ID
    - Conntrack zones included but not necessary used by container run-times
- So harvesting procfs regularly still necessary

Other kind of Netlink sockets?

NETLINK_INET_DIAG

```
socket(AF_NETLINK, SOCK_RAW, NETLINK_INET_DIAG);
```

- Fetch information about sockets
  - Used by ss (“another utility to investigate sockets”)
  - Basic bytecode to filter the sockets (e.g. “INET_DIAG_BC_JMP”)
- But no notification mechanism
  - Patch “sock_diag: notify packet socket creation/deletion” (2013) rejected

Kernel Connector

```
socket(AF_NETLINK, SOCK_RAW, NETLINK_CONNECTOR);
```

Several Kernel Connector agents:

- Device mapper
- HyperV
- Proc connector

Proc connector

```
bind(sockfd, ...CN_IDX_PROC...);
sendmsg(sockfd, ...PROC_CN_MCAST_LISTEN...)
```

- Since Linux v2.6.15 (January 2006)

Notifications for:

- fork
- exec
- exit

Proc connector

Missing:

- network namespace
  - RFC patch “proc connector: add namespace events” last month
    https://lkml.org/lkml/2016/9/8/588
- Sockets

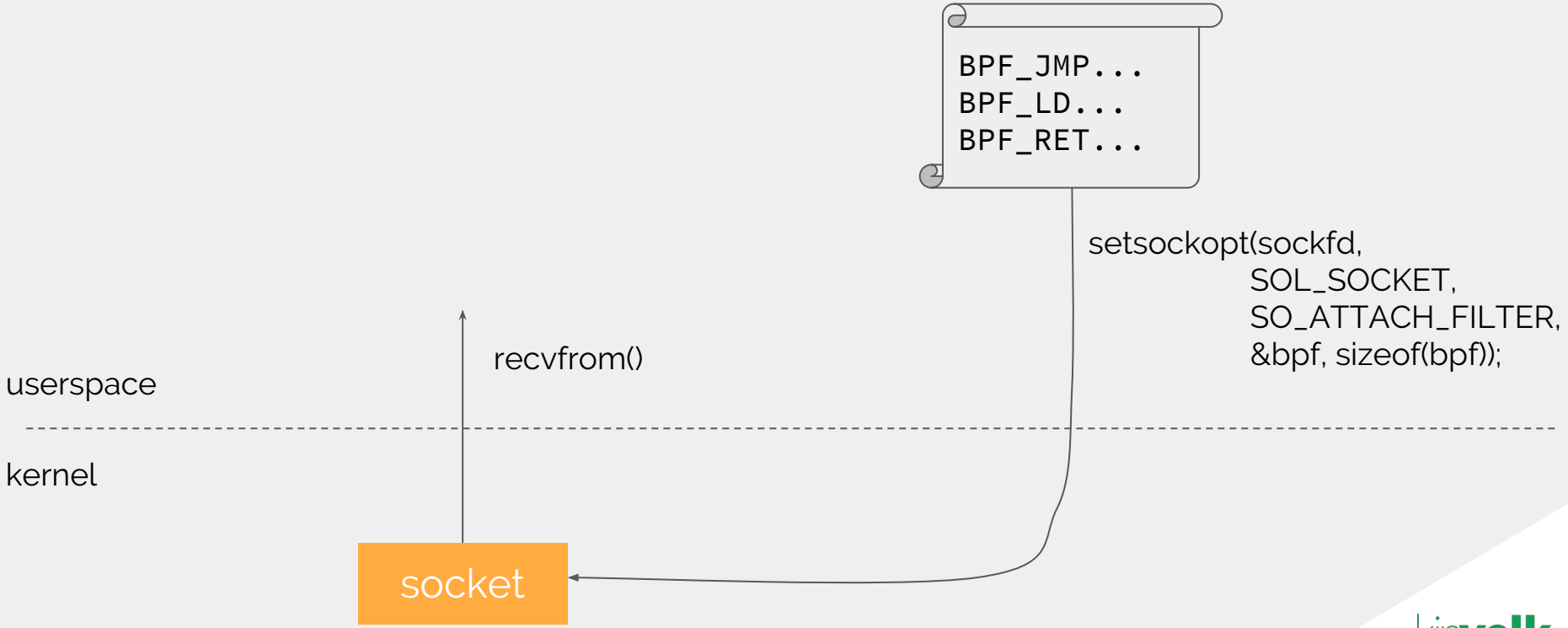
So harvesting procfs regularly still necessary

Proc connector

demo

BPF

Classic BPF (cBPF)

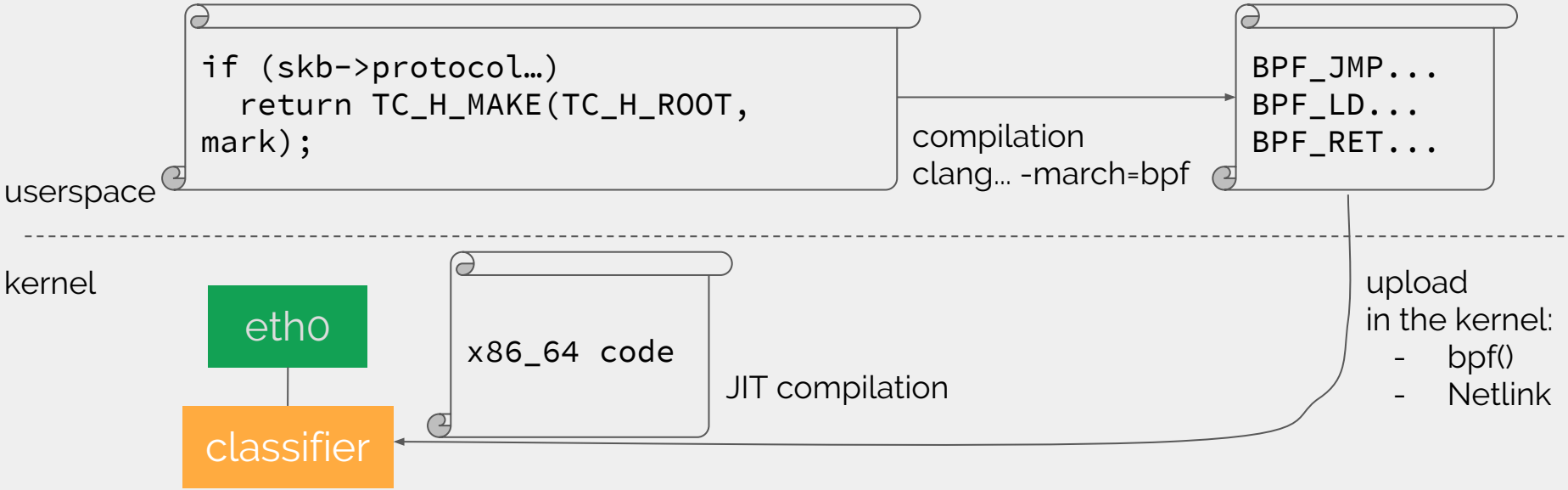


Extended BPF (or eBPF)

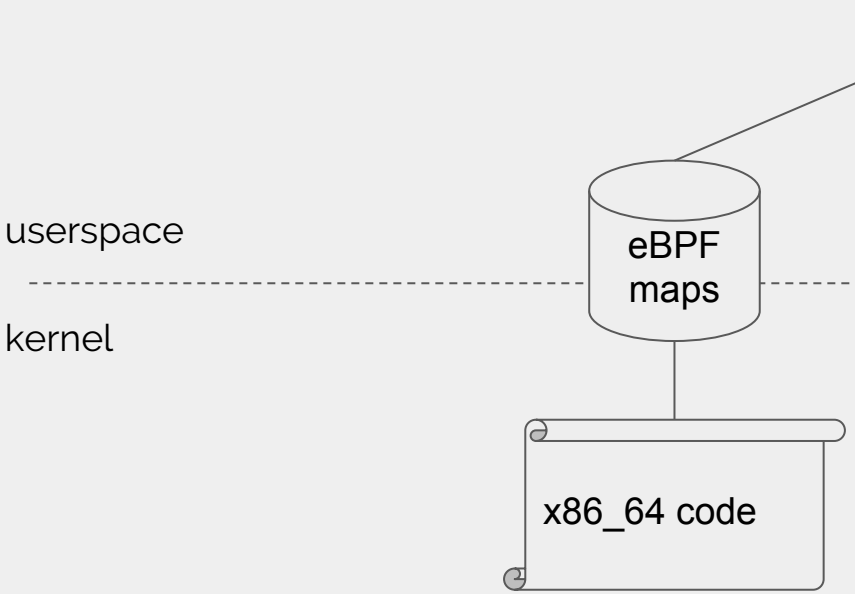
Program type:

- BPF_PROG_TYPE_SOCKET_FILTER
- BPF_PROG_TYPE_KPROBE
- BPF_PROG_TYPE_SCHED_CLS
- BPF_PROG_TYPE_SCHED_ACT
- BPF_PROG_TYPE_TRACEPOINT (Linux >= 4.7)
- BPF_PROG_TYPE_XDP

eBPF classifier for qdiscs



eBPF maps



- Keep context between calls
- Report statistics to userspace

Tracepoints with eBPF

- BPF_PROG_TYPE_TRACEPOINT since Linux 4.7
- Find the list of tracepoints in /sys/kernel/debug/tracing/events
- Stable API
- But limited tracepoints

```
# cat /sys/kernel/debug/tracing/events/skb/kfree_skb/format
name: kfree_skb
ID: 1135
format:
        field:unsigned short common_type;       offset:0;       size:2; signed:0;
        field:unsigned char common_flags;       offset:2;       size:1; signed:0;
        field:unsigned char common_preempt_count;       offset:3;       size:1; signed:0;
        field:int common_pid;   offset:4;       size:4; signed:1;

        field:void * skbaddr;   offset:8;       size:8; signed:0;
        field:void * location;  offset:16;      size:8; signed:0;
        field:unsigned short protocol;  offset:24;      size:2; signed:0;

print fmt: "skbaddr=%p protocol=%u location=%p", REC->skbaddr, REC->protocol, REC->location
```

kprobes with eBPF

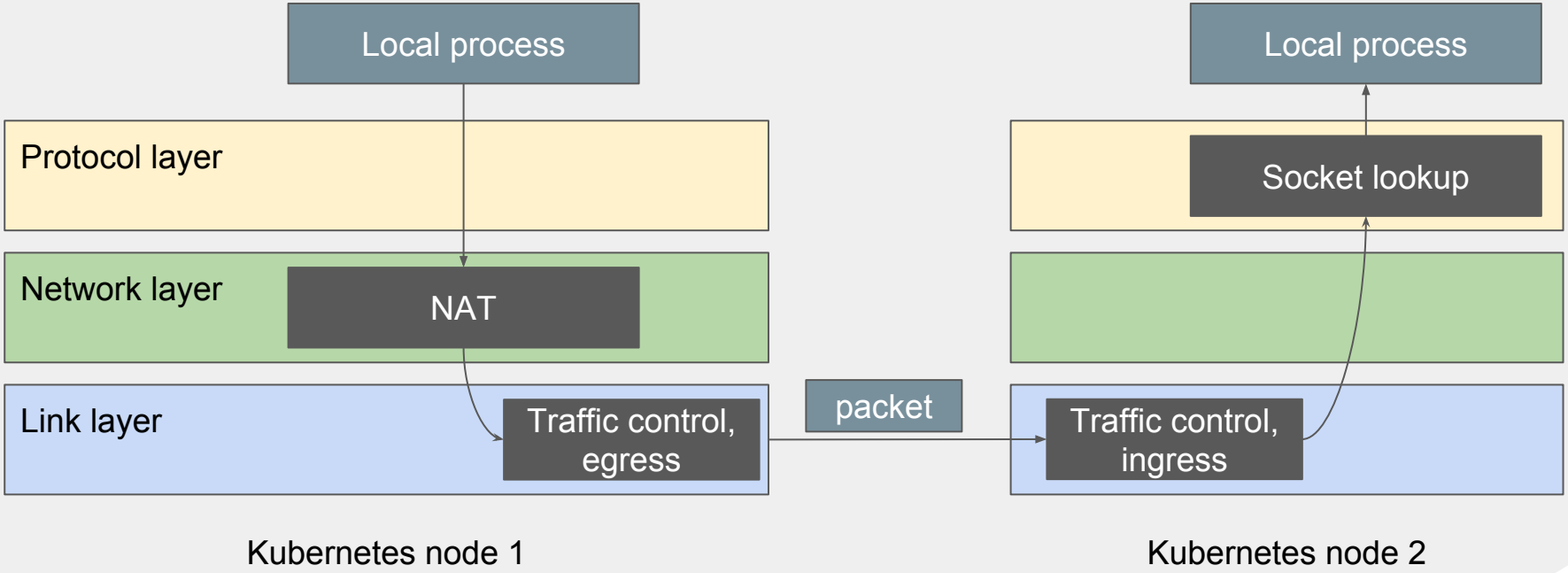
- BPF_PROG_TYPE_KPROBE since Linux 4.1
- No ABI guarantees
- Probe any kernel function

Socket events with kprobe / eBPF

- BPF Compiler Collection (BCC)
  - bcc/examples/tracing/tcpv4connect.py
  - Iago's tcp4tracer (WIP)
    - Get connection tuple, pid, netns
    - tcp_v4_connect
    - tcp_close
    - inet_csk_accept

```
TYPE        PID    COMM             SADDR            DADDR            SPORT  DPORT  NETNS
connect     7736   nc               192.168.35.25    185.46.139.24    45426  80     4026531969
close       7736   nc               192.168.35.25    185.46.139.24    45426  80     4026531969
```

Packet modifications



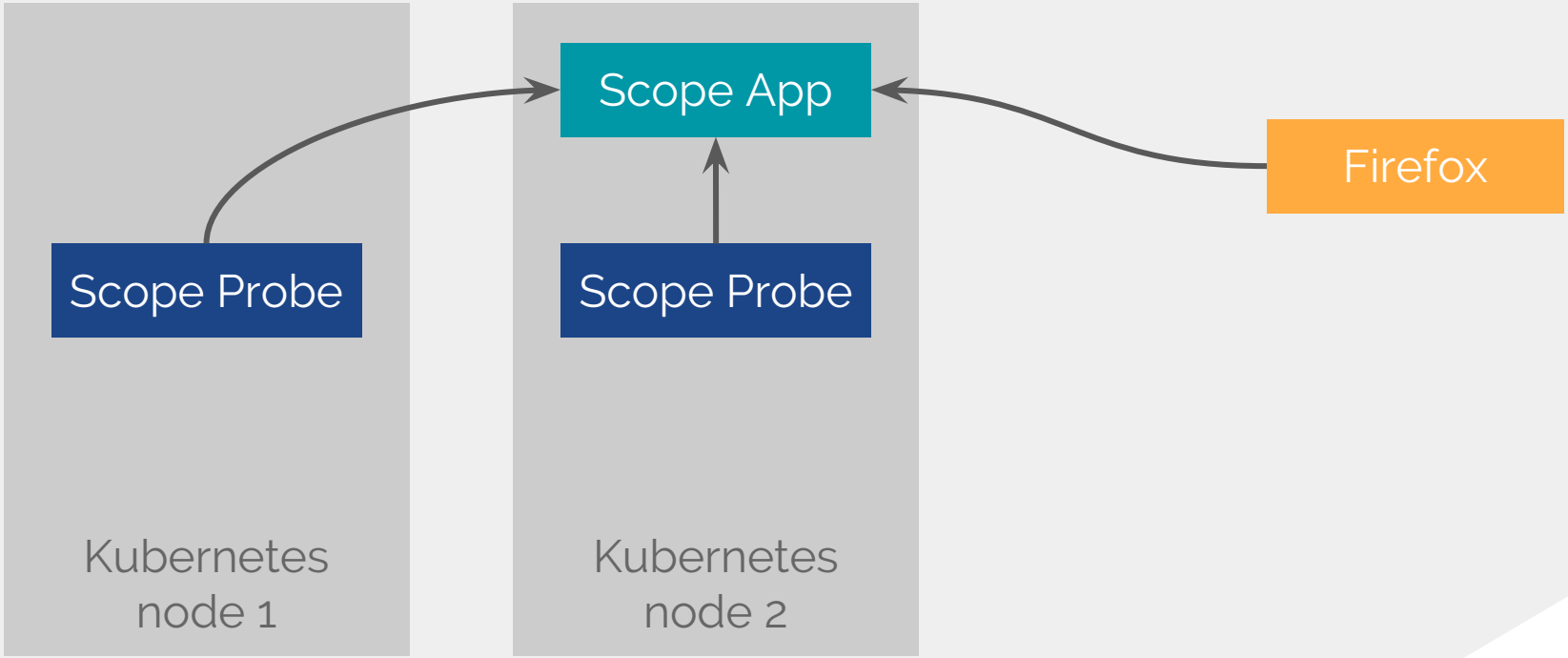
tcp4tracer & NAT

- The connection tuple from the process' point of view is not enough
  - NAT
  - Kubernetes Services
- Iago's tcp4tracer (WIP)
  - nf_nat_ipv4_manip_pkt
  - nf_nat_tcp_manip_pkt

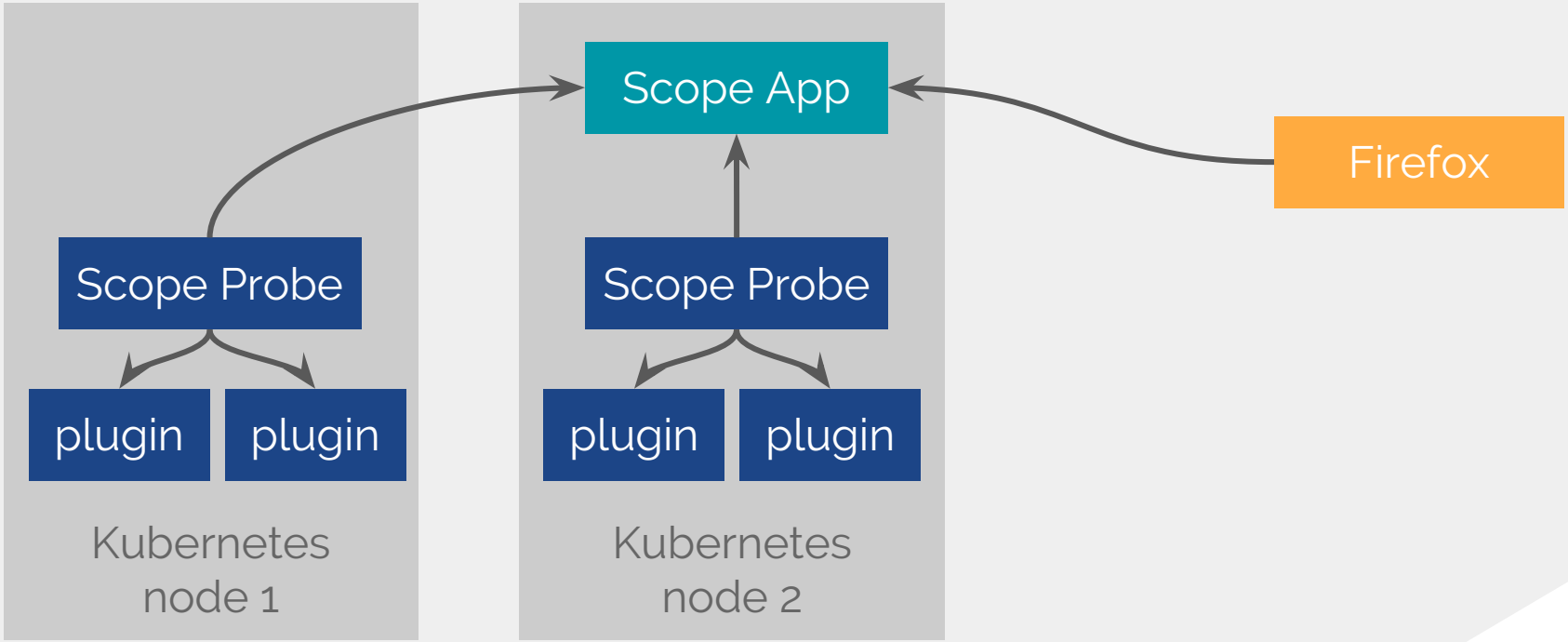
TYPE	PID	COMM	SADDR	DADDR	SPORT	DPORT	NETNS
connect	19727	nc	172.16.28.11	185.46.139.27	55366	80	4026532470
tcp_nat	19727	nc	0.0.0.0	0.0.0.0	55366	80	0
ip_nat	19727	nc	192.168.35.25	185.46.139.27	0	0	0

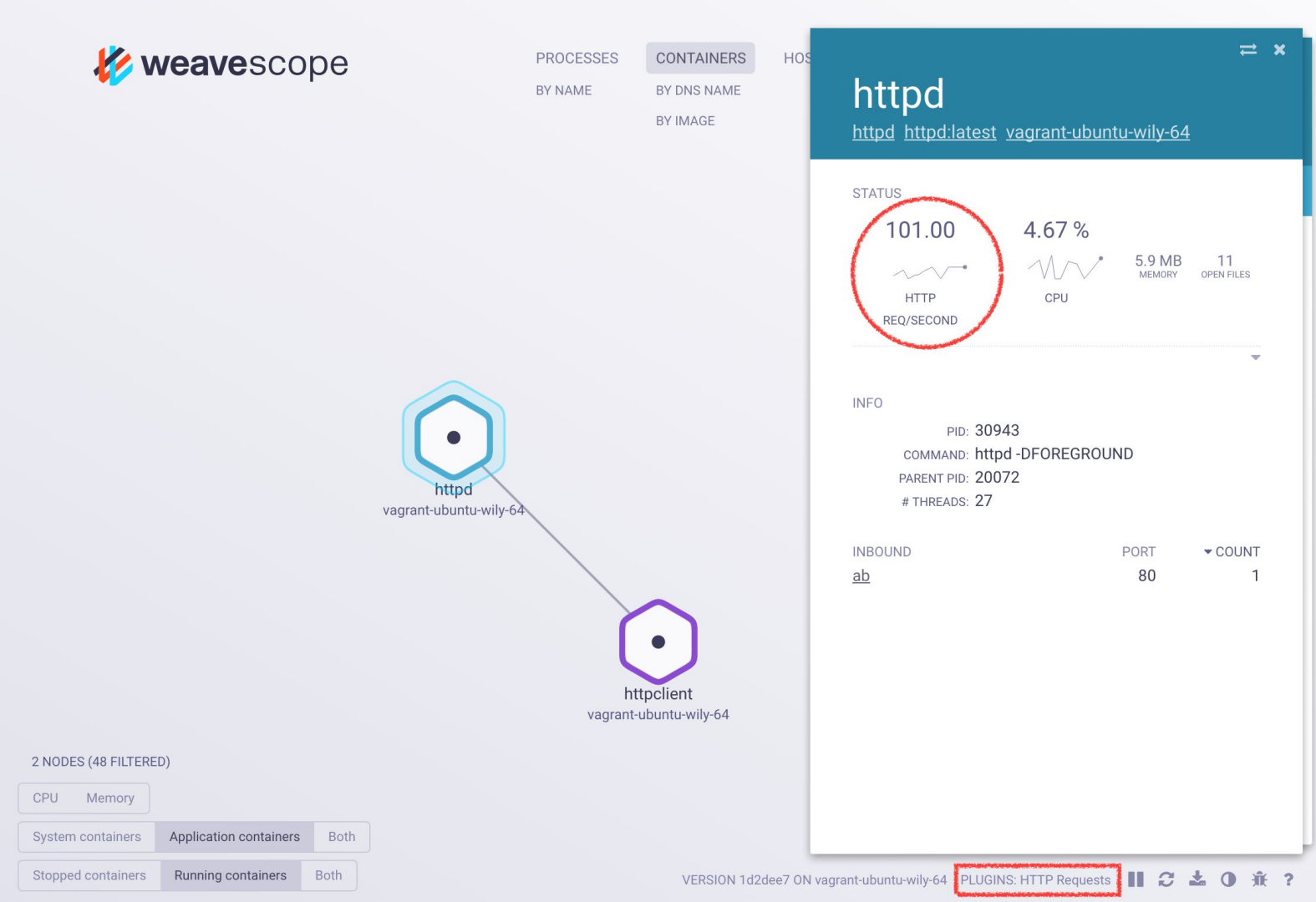
More metrics

Weave Scope architecture



Weave Scope plugins



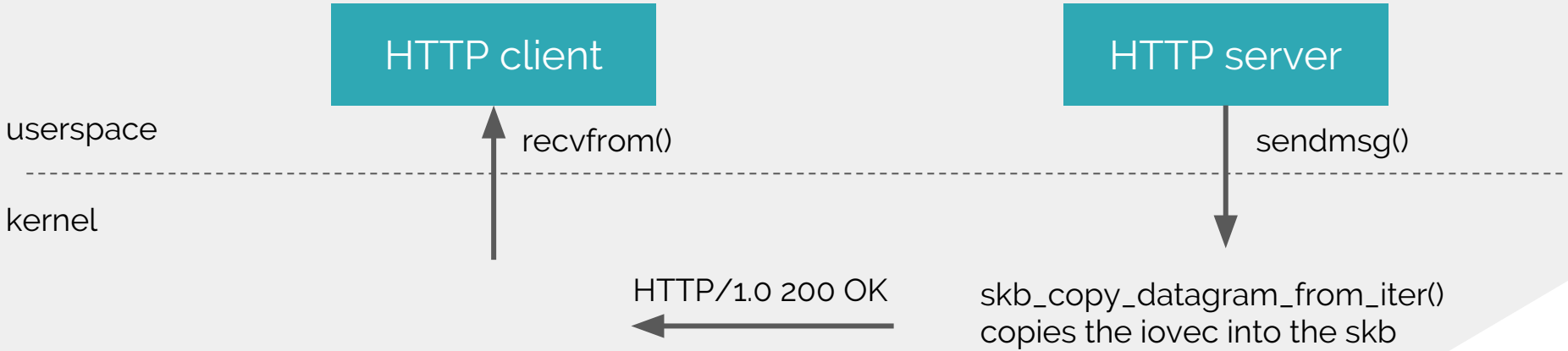

weavescope
PROCESSES
CONTAINERS
HOS
BY NAME
BY DNS NAME
BY IMAGE
httpd
httpd:latest
vagrant-ubuntu-wily-64
STATUS
101.00
HTTP
REQ/SECOND
4.67 %
CPU
5.9 MB
MEMORY
11
OPEN FILES
INFO
PID: 30943
COMMAND: httpd -DFOREGROUND
PARENT PID: 20072
THREADS: 27
INBOUND
PORT
COUNT
ab
80
1
httpd
vagrant-ubuntu-wily-64
httpclient
vagrant-ubuntu-wily-64
2 NODES (48 FILTERED)
CPU
Memory
System containers
Application containers
Both
Stopped containers
Running containers
Both
VERSION 1d2dee7 ON vagrant-ubuntu-wily-64
PLUGINS: HTTP Requests

HTTP requests plugin

- Number of HTTP requests per second
- Without instrumenting the application
- eBPF kprobe on skb_copy_datagram_iter

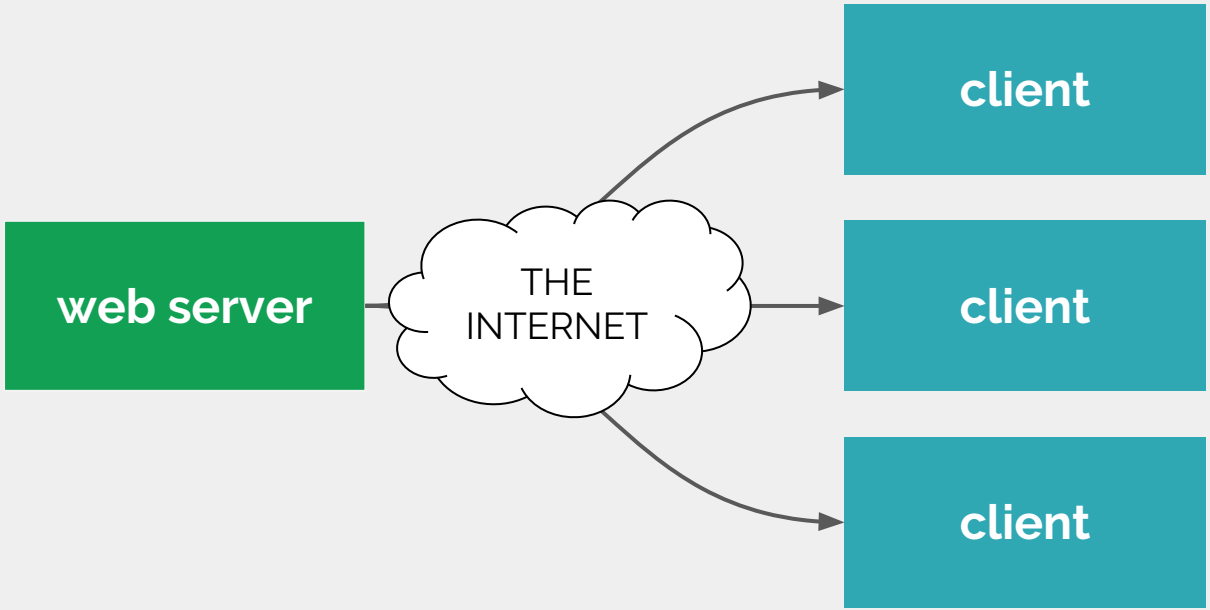
HTTP responses plugin

- Number of HTTP responses by category (404, etc.)
- Without instrumenting the application
- eBPF kprobe on skb_copy_datagram_from_iter
- Using an eBPF map to track the context between kprobe & kretprobe



Testing degraded networks

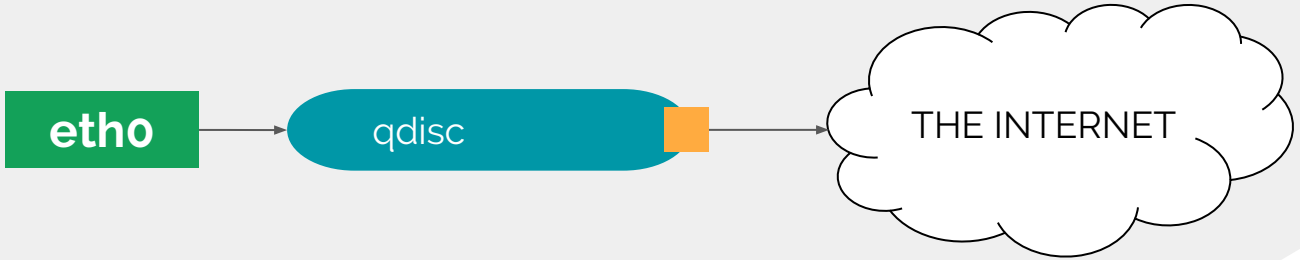
Traffic control, why?



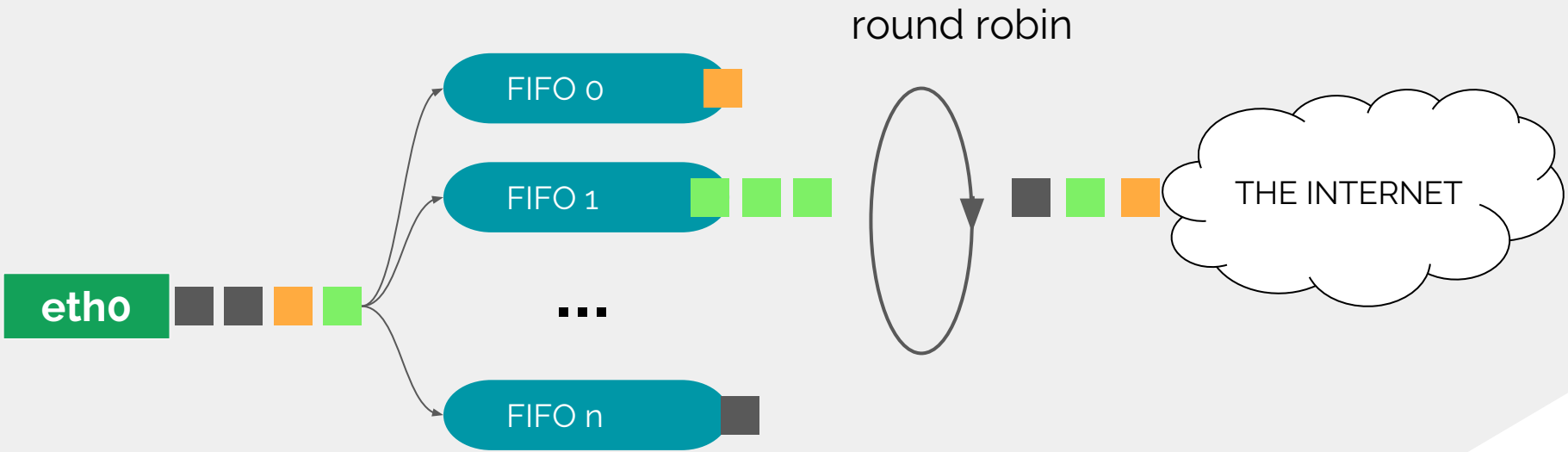
- fair distribution of bandwidth
- reserve bandwidth to specific applications
- avoid bufferbloat

Queuing disciplines (qdisc)

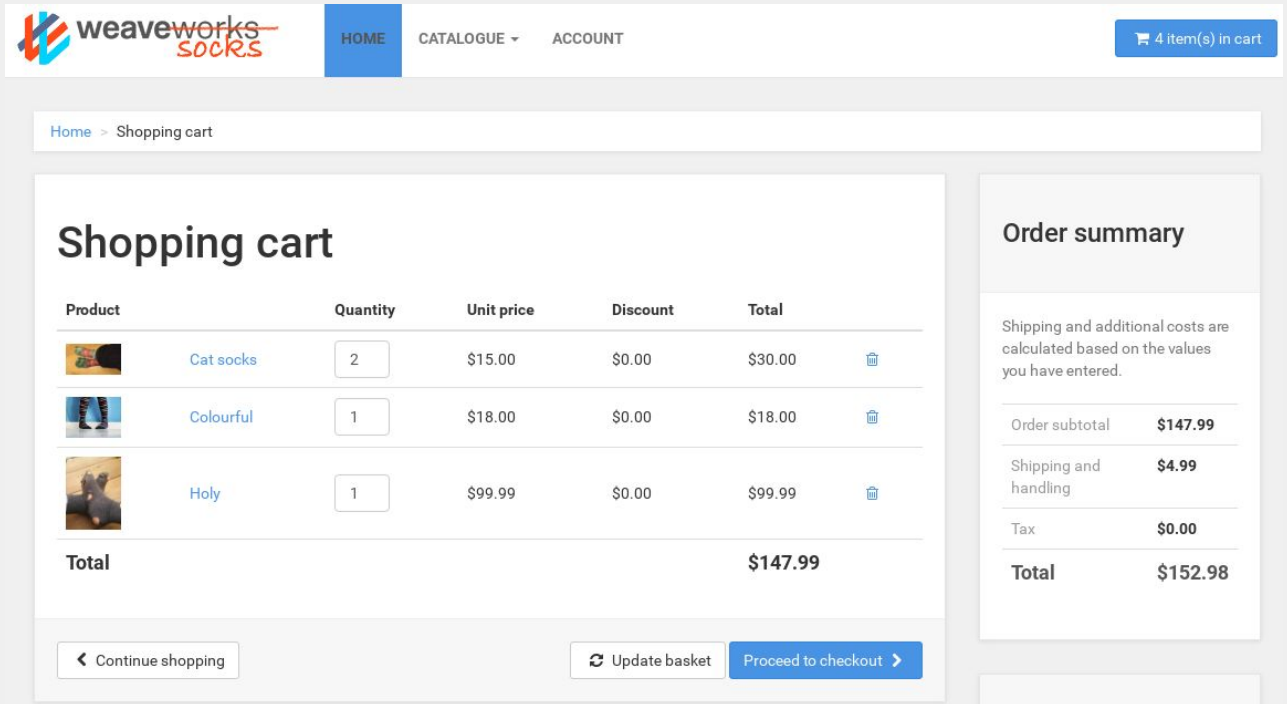
- Network scheduling algorithm
  - which packet to emit next?
  - when?
- Configurable at run-time:
  - /sbin/tc
  - Netlink
- Default on new network interfaces: sysctl net.core.default_qdisc



Stochastic Fairness Queueing (sfq)



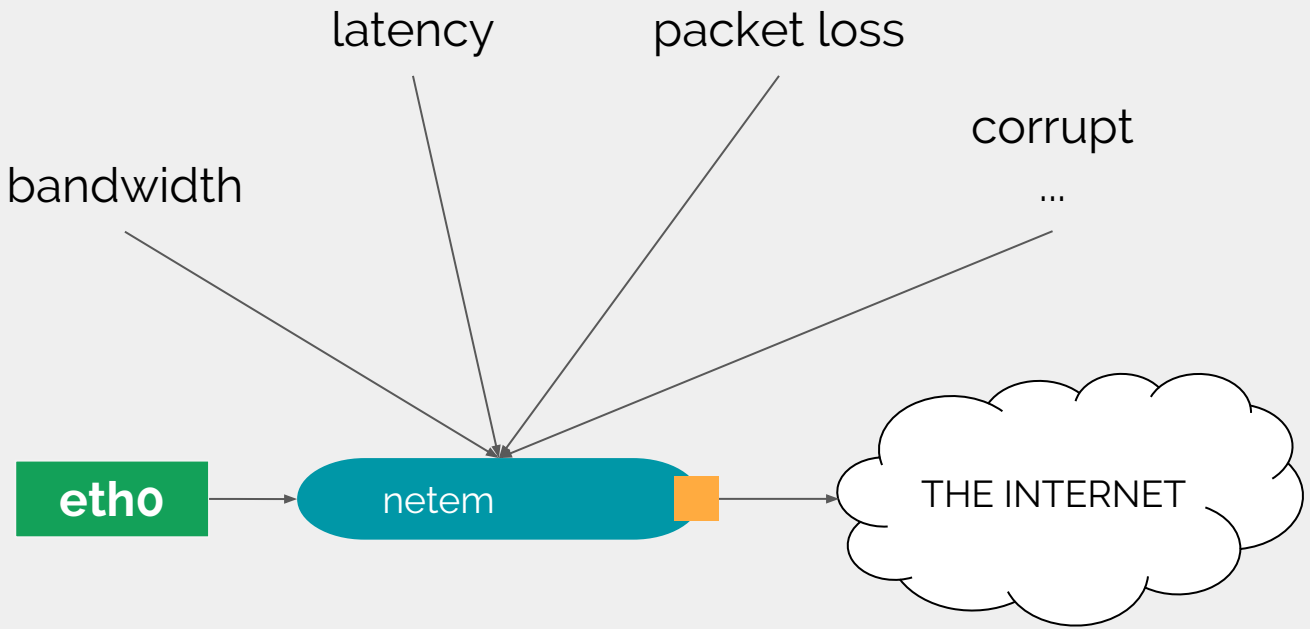
Demo



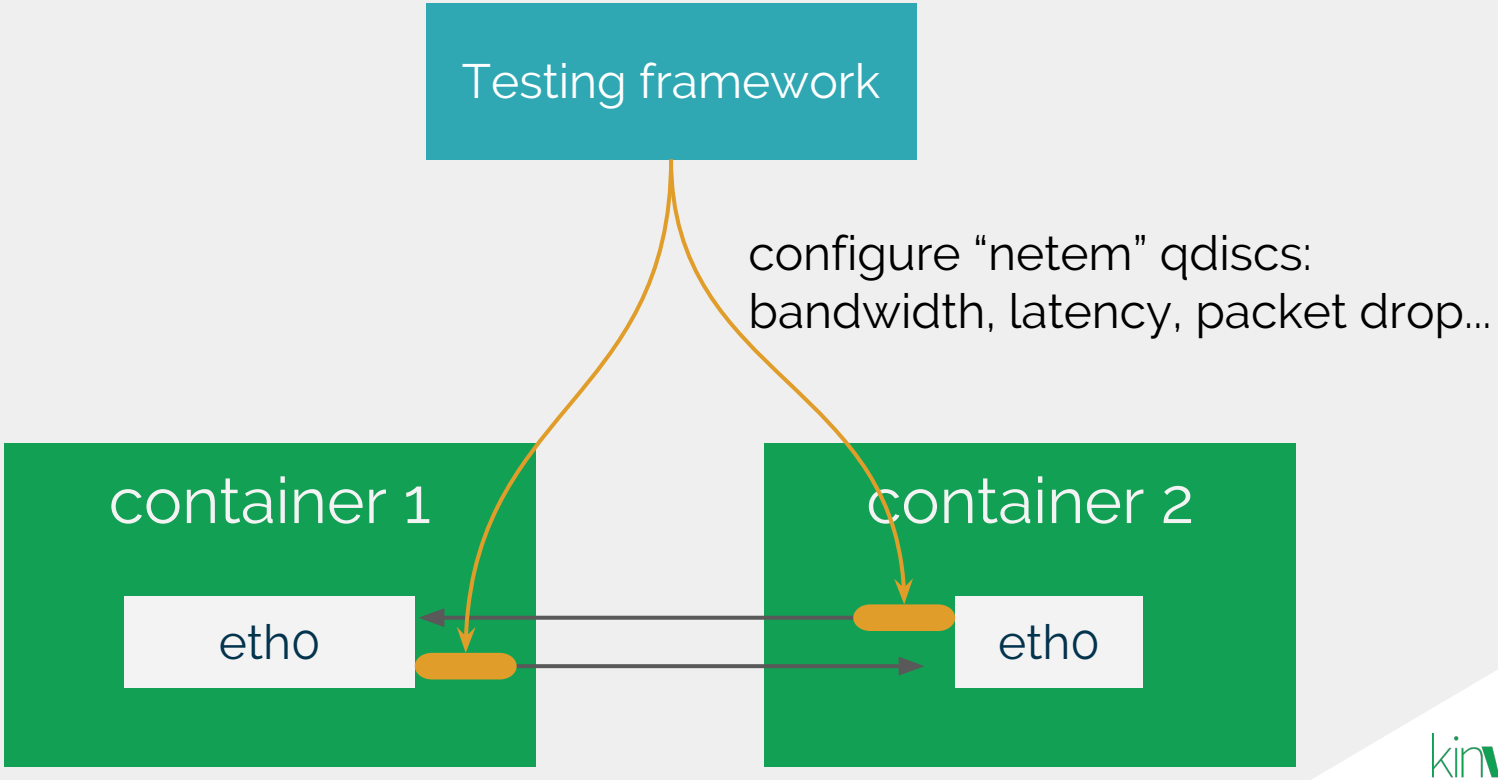
Reproduce this demo yourself: `https://github.com/kinvolk/demo`

kinvolk.io

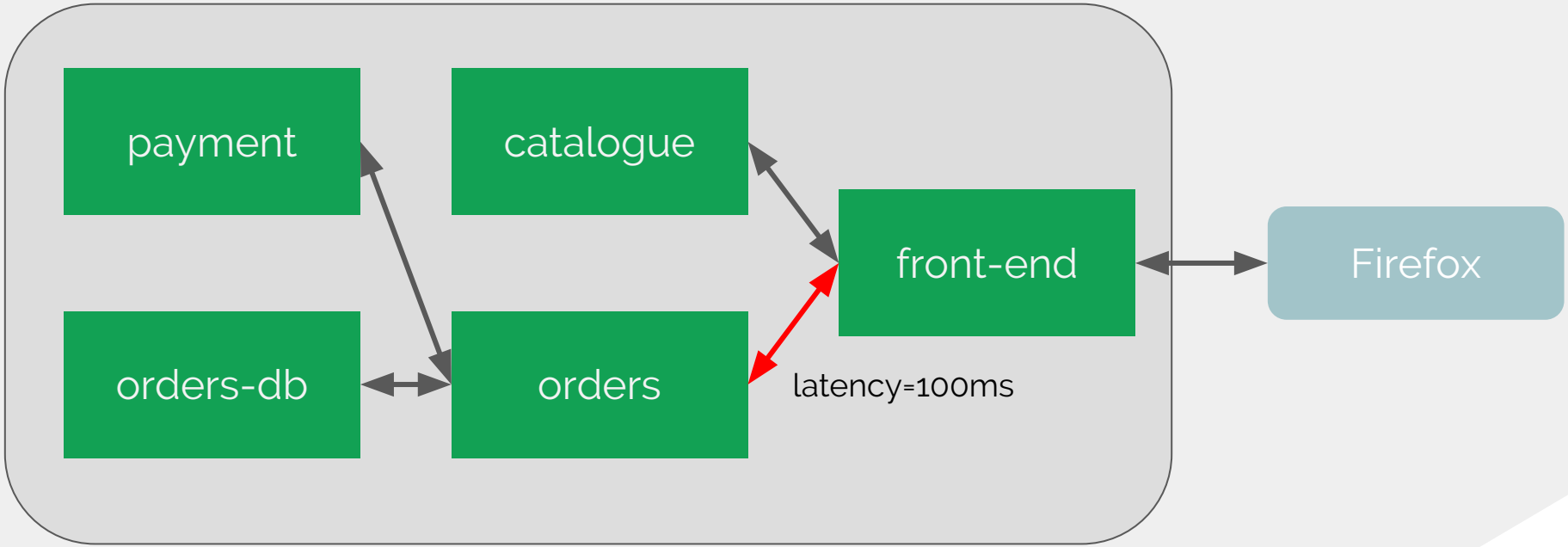
Network emulator (netem)



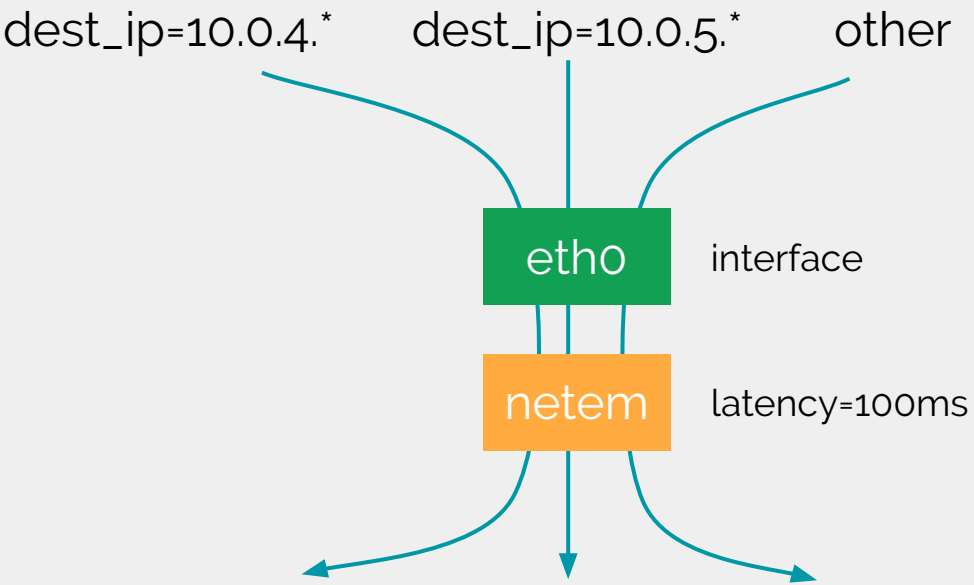
Testing with containers



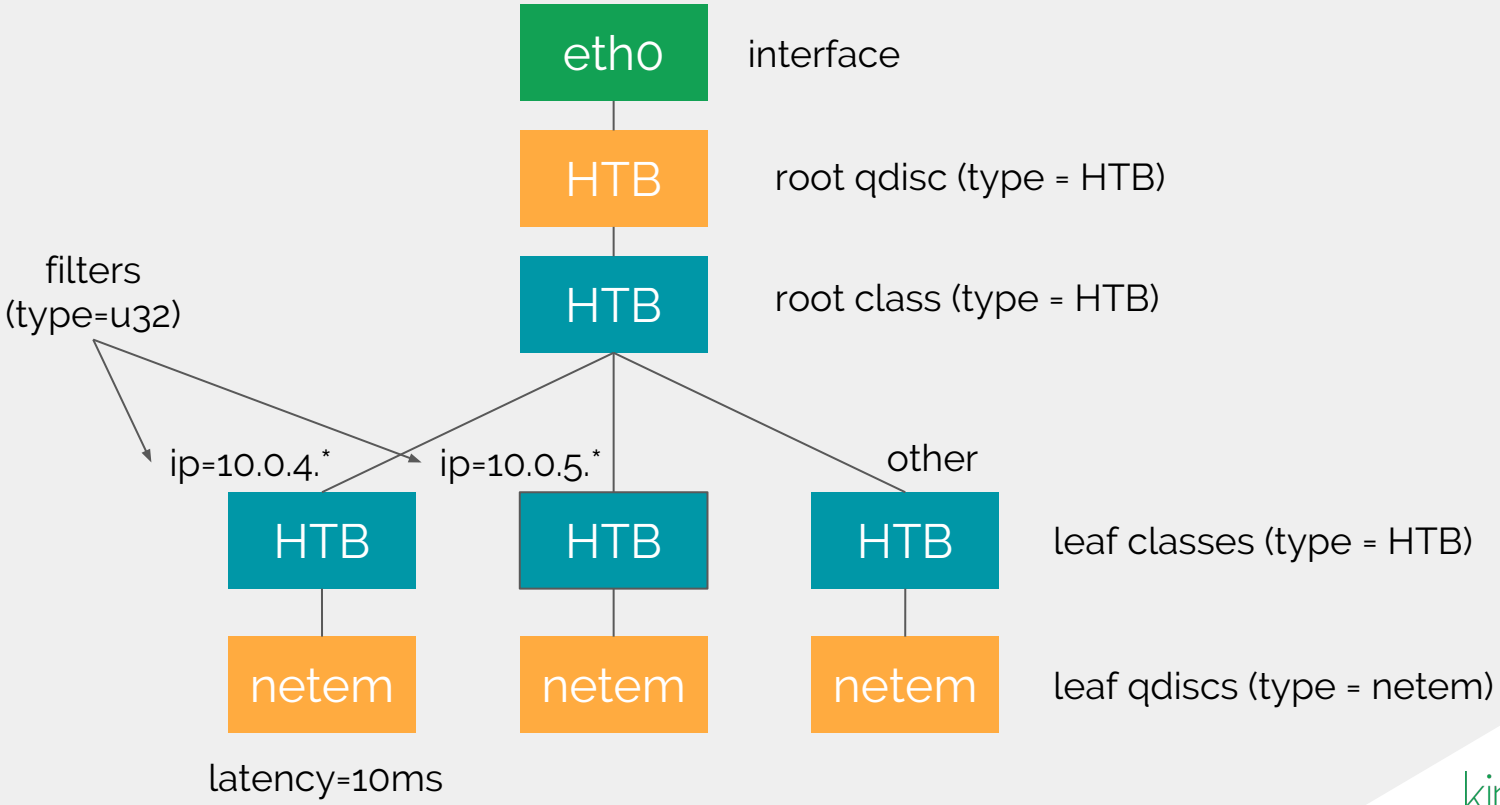
Add latency on a specific connection



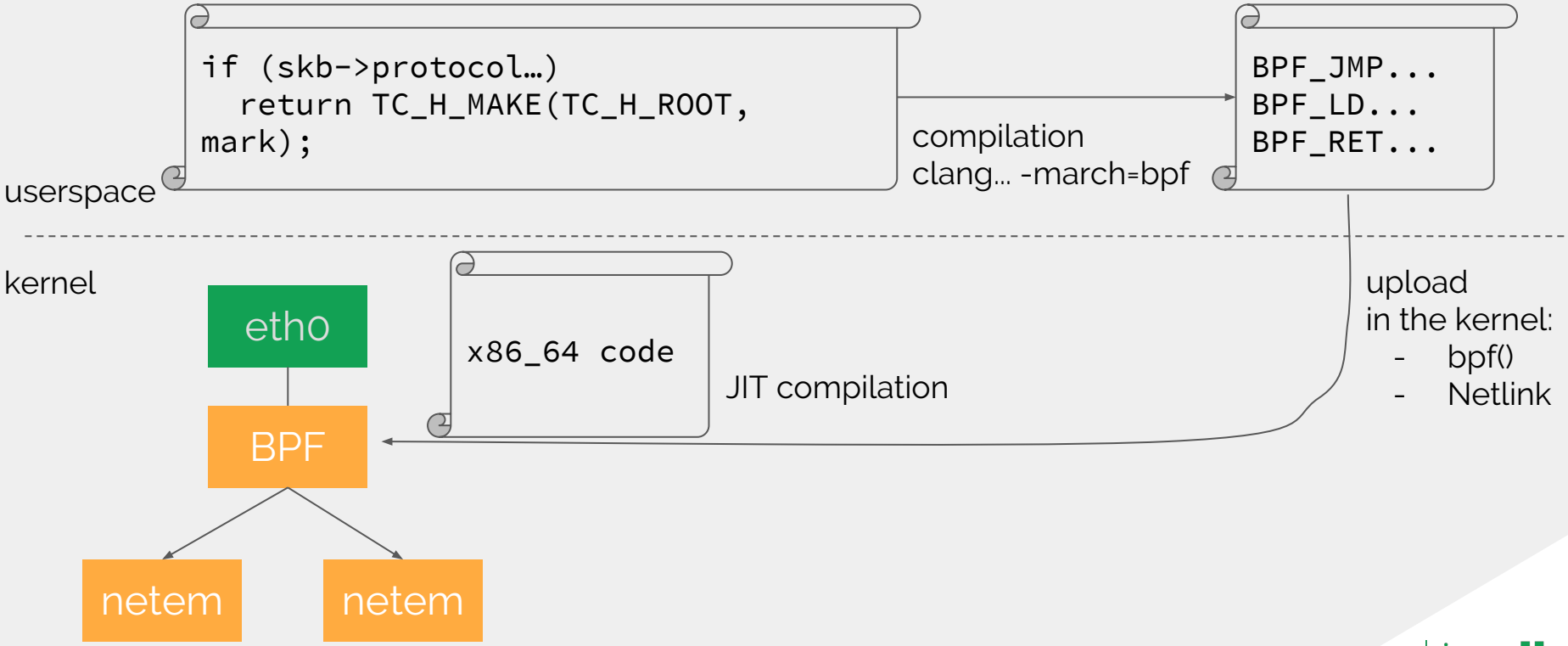
How to define classes of traffic



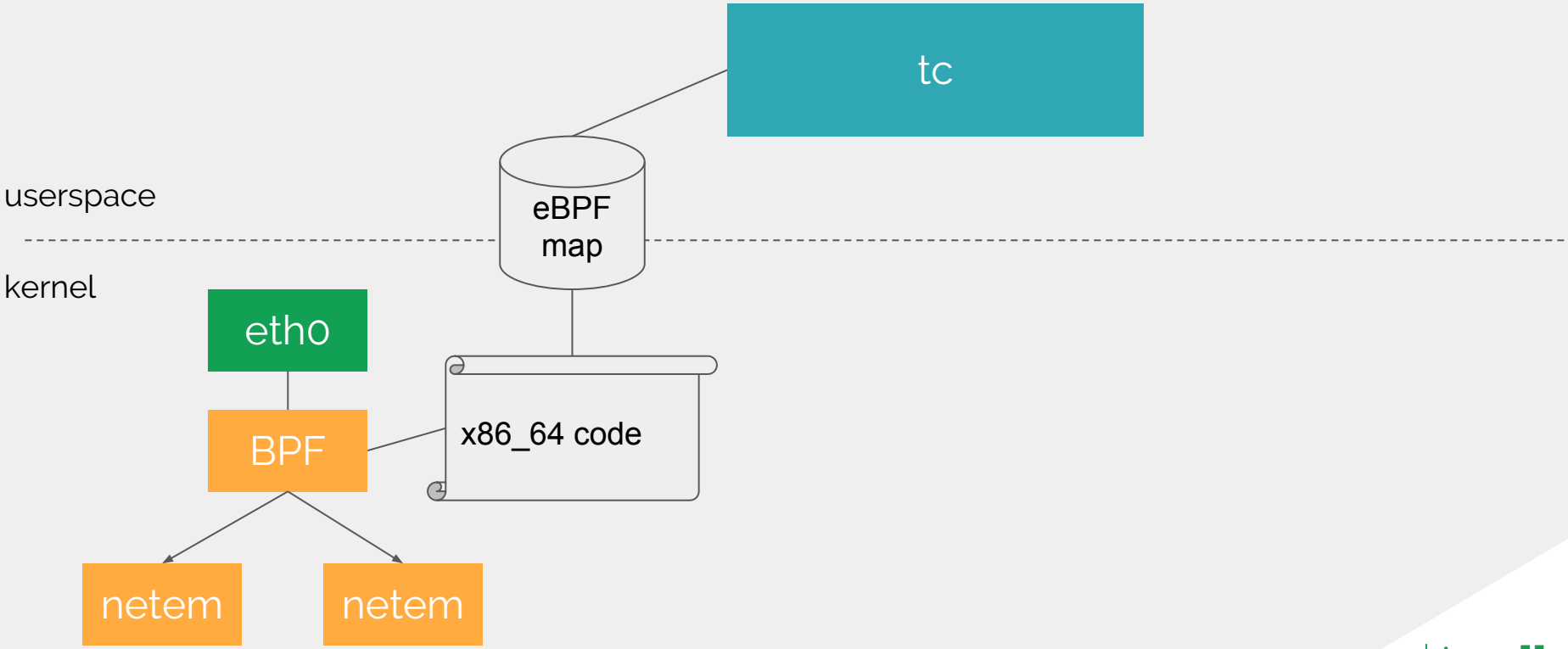
u32: filter on content



Filtering with cBPF/eBPF



eBPF maps



Questions?

The slides: https://goo.gl/iDL8te