Containers, systemd and ~~Prometheus~~ Kubernetes: Running Zeek in 2026

Arne Welzel

About Me

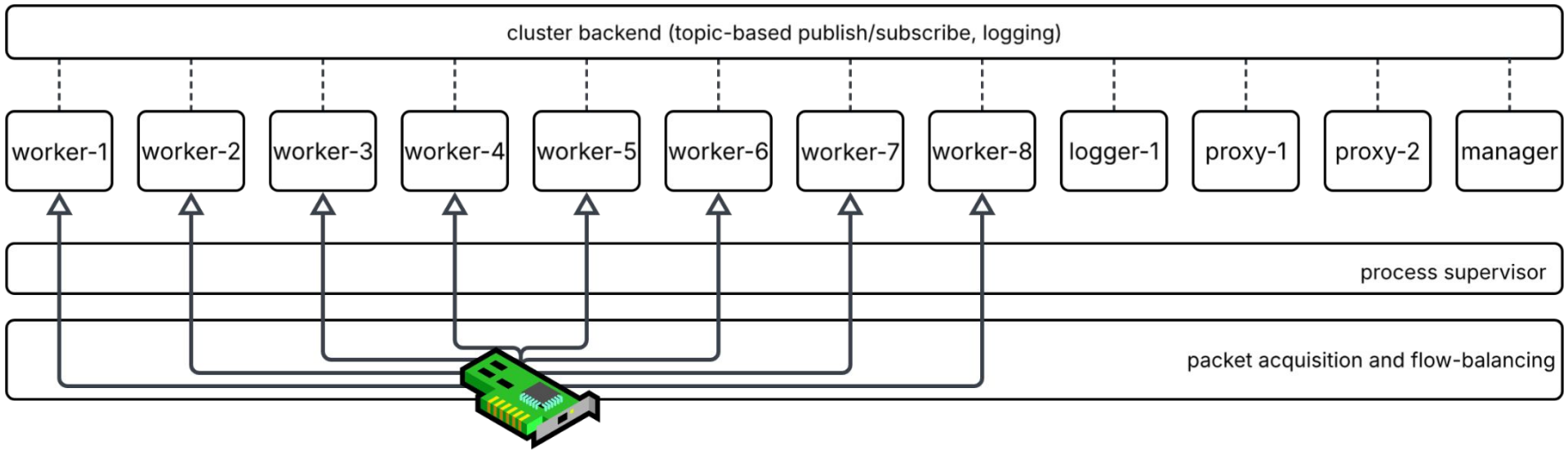
- Zeek since mid-2022, Corelight
- Zeek Cluster on ZeroMQ
- Also systems engineering, integration, performance…

Containers, systemd and Kubernetes: Running Bro in 2016

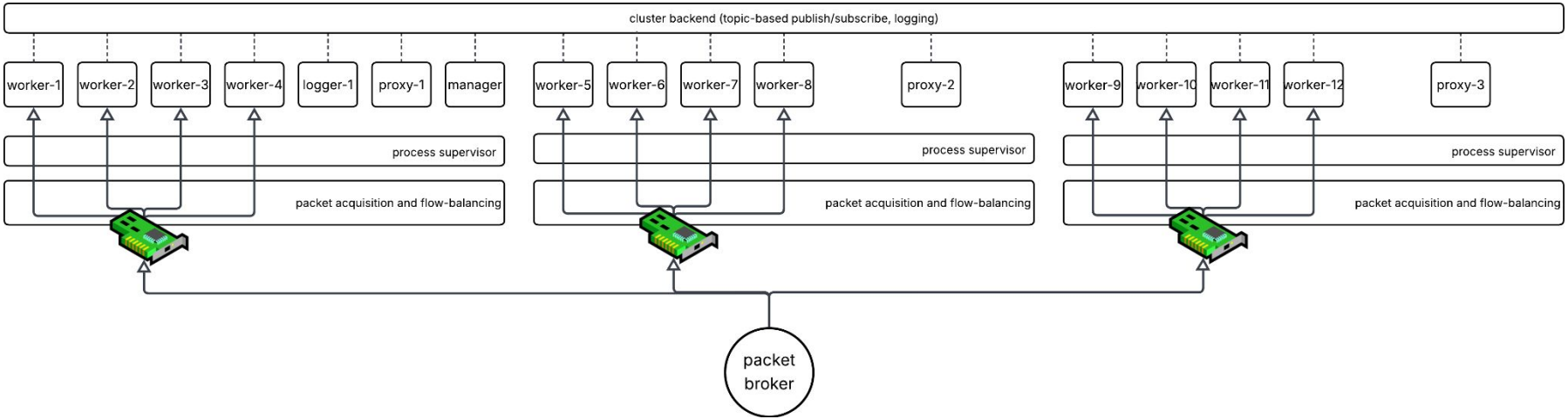
Overview

- Zeek is a Multi-process Application
- Running a Zeek Cluster
- Zeek Appliance
- zeek-systemd-generator and zeek.conf in 8.1
- systemd in a container?!
- The zeek/zeek-demo repository
- Zeek Nodes as Kubernetes Nodes?
- Question / Discussion

Zeek is a Multi-process Application (Single Node)

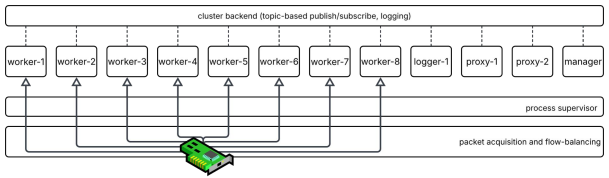


Zeek is a Multi-process Application (Multi Node)



Running a Zeek Cluster: Manually

```
$ zeek-cluster-layout-generator -L 1 -P 2 -W 8 -o cluster-layout.zeek

$ export ZEEKPATH=$(pwd):$ZEEKPATH

$ mkdir manager  && cd manager  && CLUSTER_NODE=manager  zeek local &
$ mkdir logger-1 && cd logger-1 && CLUSTER_NODE=logger-1 zeek local &
$ mkdir proxy-1  && cd proxy-1  && CLUSTER_NODE=proxy-1  zeek local &
$ mkdir proxy-2  && cd proxy-2  && CLUSTER_NODE=proxy-2  zeek local &

$ mkdir worker-1 && cd worker-1 && CLUSTER_NODE=worker-1 zeek -i af_packet::eth0 local &
...
$ mkdir worker-8 && cd worker-8 && CLUSTER_NODE=worker-8 zeek -i af_packet::eth0 local &
```



Running a Zeek Cluster: Tooling

BroControl / ZeekControl

```
root@ubuntu24:~#
root@ubuntu24:~# zeekctl deploy
checking configurations ...
installing ...
creating policy directories ...
installing site policies ...
generating cluster-layout.zeek ...
generating local-networks.zeek ...
generating zeekctl-config.zeek ...
generating zeekctl-config.sh ...
stopping ...
stopping workers ...
stopping proxy ...
stopping manager ...
stopping logger ...
starting ...
starting logger ...
starting manager ...
starting proxy ...
starting workers ...
root@ubuntu24:~#
```

```
[logger]
type=logger
host=10.0.0.10

[manager]
type=manager
host=10.0.0.10

[proxy-1]
type=proxy
host=10.0.0.10

[worker-1]
type=worker
host=10.0.0.11
interface=eth0

[worker-2]
type=worker
host=10.0.0.12
interface=eth0
```

Commits on Nov 10, 2010

Initial import from Bro SVN

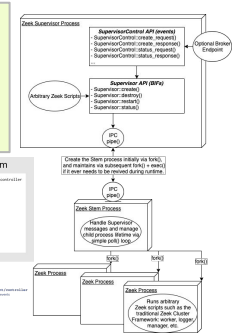
r7107. afd99a4

jsiwek committed on Nov 10, 2010

Supervisor API (2019-) / Management Framework (2021-)

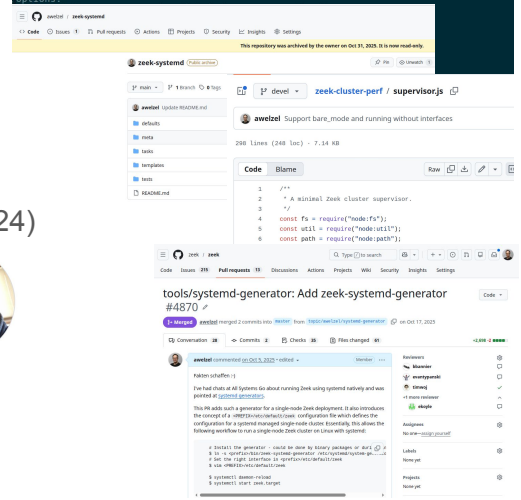
```
event zeek_init()
    {
    if ( ! Supervisor::is_supervisor() )
        return;

    Broker::listen("127.0.0.1", 9999/tcp);

    local cluster: table[string] of Supervisor::ClusterEndpoint;
    cluster["manager"] = [$role=Supervisor::MANAGER, $host=127.0.0.1, $p=
    cluster["logger"] = [$role=Supervisor::LOGGER, $host=127.0.0.1, $p=10
    cluster["proxy"] = [$role=Supervisor::PROXY, $host=127.0.0.1, $p=1000
    cluster["worker"] = [$role=Supervisor::WORKER, $host=127.0.0.1, $p=10

    for ( n, ep in cluster )
        {
        local sn = Supervisor::NodeConfig($
        sn$cluster = cluster;
        sn$directory = n;

        if ( ep?$interface )
            sn$interface = ep$interface;

        local res = Supervisor::create(sn);

        if ( res != "" )
            print fmt("supervisor failed to create node '%s
        }
    }
```

```
[manager]
role = manager

[logger]
role = logger

[worker-01]
role = worker
interface = lo

[worker-02]
role = worker
interface = eth0
```



My own attempts (2020-)

```
$ python3 run-mini-cluster.py  --help
usage: run-mini-cluster.py [-h] [--log-level {debug,info,error}]
                           [--tmp-scripts TMP SCRIPTS]
                           [--run-for-seconds RUN_FOR_SECONDS] [--workers WORKERS]
                           [--proxies PROXIES] [--loggers LOGGERS] [--env [ENV ...]]
                           [--workdir WORKDIR] [--interface INTERFACE]
                           scripts [scripts ...]

positional arguments:
  scripts

options:
```



Corelight (2015-)

```
# Ansible, templates, Shell, ...
$ bro-restart-all
$ sv restart bro-worker-01

$ zeek supervisor.zeek

And more Seth ideas!
```

ESnet (2019-2024)

Vlad Grigorescu
grigorescu

dopheide-esnet

```
# Ansible, systemd, ?
$ systemctl
$ journalctl
```

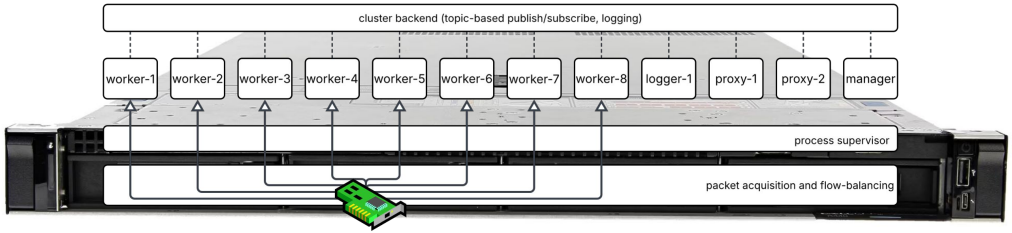
Running a Zeek Cluster: Status Quo

- ZeekControl
  - Does a lot more than just process management!
  - Lots of scripts, nohup, state files
  - Memory limits via ulimit, CPU pinning via taskset
  - Cronjob to restart crashed processes
  - Multi-node first
- Management Framework
  - Started sometime in 2019
  - Number of known users today: 1 (ESnet)
- Custom Implementations

```
[Unit]
Description=Zeek

[Service]
User=zeek
Type=forking
Restart=on-failure
RestartSec=5
ExecStart=/opt/zeek/bin/zeekctl deploy
ExecStop=/opt/zeek/bin/zeekctl stop

[Install]
WantedBy=multi-user.target
```

Zeek Appliance

- Corelight, Security Onion, Malcolm, …
- Small systems with a few workers
  - 4 workers, 1 logger, 1 proxy, 1 manager
- Large systems with huge NICs
  - 100+ workers, more loggers, more proxies, …



Zeek Appliance: Process Supervisor

- Process Management
- Memory Limits
- Sandboxing
- CPU Affinity

- Linux-only is fine

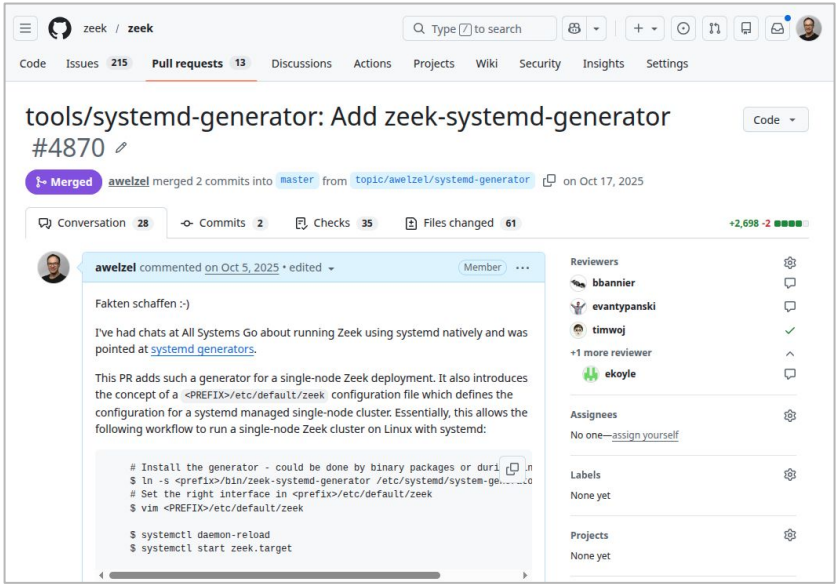
- Well-known, ubiquitous, widely supported, off-the-shelf, …


systemd

zeek-systemd-generator and zeek.conf

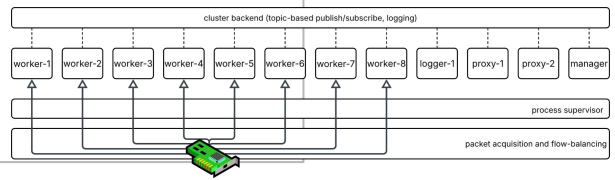
- Zeek 8.1 includes too [tools/systemd-generator]
  - [systemd.generator(7)] - run during boot and at daemon-reload time

```
$ cat <PREFIX>/etc/zeek/zeek.conf
interface = af_packet::wlp0s20f3
workers = 2
worker_memory_max = 1024M

# Generate unit files
$ systemctl daemon-reload


# Start all Zeek processes
$ systemctl start zeek.target
```



zeek-systemd-generator and zeek.conf

```
$ tree /run/systemd/generator/ | grep zeek
├── zeek-archiver.service
├── zeek-logger@.service
├── zeek-manager.service
├── zeek-proxy@.service
├── zeek-setup.service
├── zeek.target
├── zeek.target.wants
│   ├── zeek-archiver.service -> ../zeek-archiver.service
│   ├── zeek-logger@1.service -> ../zeek-logger@.service
│   ├── zeek-manager.service -> ../zeek-manager.service
│   ├── zeek-proxy@1.service -> ../zeek-proxy@.service
│   ├── zeek-setup.service -> ../zeek-setup.service
│   ├── zeek-worker@1.service -> ../zeek-worker@.service
│   └── zeek-worker@2.service -> ../zeek-worker@.service
├── zeek-worker@1.service.d
│   └── 10-zeek-systemd-generator.conf
├── zeek-worker@2.service.d
│   └── 10-zeek-systemd-generator.conf
└── zeek-worker@.service
```

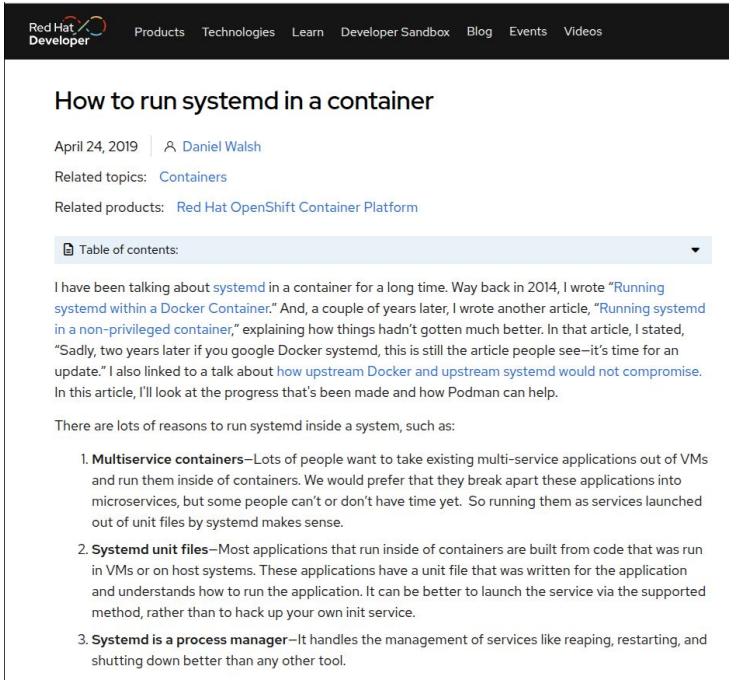


zeek-systemd-generator and zeek.conf

```
$ cat /run/systemd/generator/zeek-worker@.service
[Unit]
Description=Zeek Worker %i
SourcePath=/opt/zeek-dev-prod/etc/zeek/zeek.conf
After=zeek-setup.service
After=zeek-manager.service
After=zeek-logger@.service
After=zeek-proxy@.service
StopPropagatedFrom=zeek.target
StartLimitIntervalSec=0

[Service]
SyslogIdentifier=zeek-worker-%i
Type=exec
Nice=0
MemoryMax=1024M
User=zeek
Group=zeek
WorkingDirectory=/opt/zeek-dev-prod/var/spool/zeek/worker-%i
ReadWritePaths=/opt/zeek-dev-prod/var/spool/zeek/worker-%i
CapabilityBoundingSet=CAP_NET_RAW
AmbientCapabilities=CAP_NET_RAW
Environment=PATH=/opt/zeek-dev-prod/bin:/usr/local/bin:/usr/bin:/bin
Environment=ZEEKPATH=/opt/zeek-dev-prod/var/spool/zeek/generated-scripts:/opt/zeek-dev-prod/share/zeek:/opt/zeek-dev-prod/share/zeek/po
licy:/opt/zeek-dev-prod/share/zeek/site:/opt/zeek-dev-prod/share/zeek/builtin-plugins
Environment=CLUSTER_NODE=worker-%i
ExecStart=/opt/zeek-dev-prod/bin/zeek -i ${INTERFACE} policy/misc/systemd-generator -C local frameworks/cluster/backend/zeromq
Slice=zeek-workers.slice
Restart=always
RestartSec=1
```

systemd in a container?!

systemd in a container?!



Red Hat Developer | Products | Technologies | Learn | Developer Sandbox | Blog | Events | Videos

How to run systemd in a container

April 24, 2019 | Daniel Walsh

Related topics: Containers

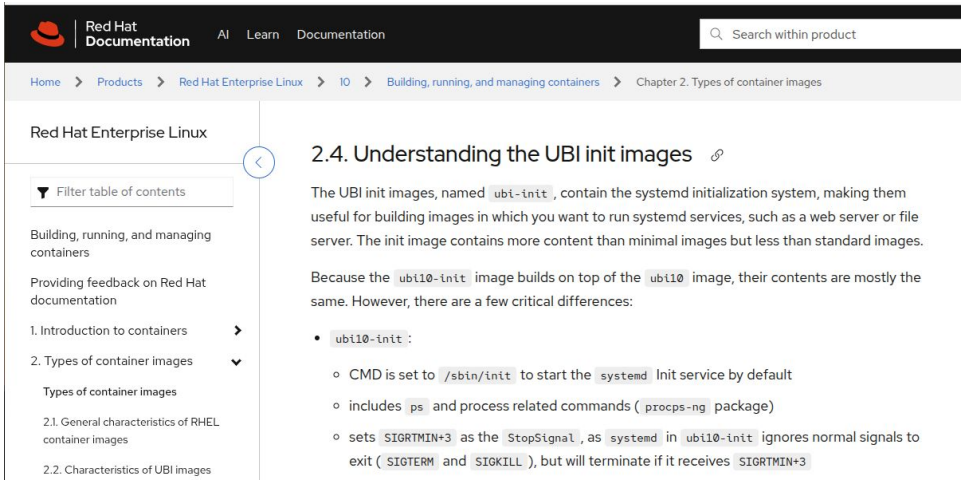
Related products: Red Hat OpenShift Container Platform

Table of contents:

I have been talking about systemd in a container for a long time. Way back in 2014, I wrote "Running systemd within a Docker Container." And, a couple of years later, I wrote another article, "Running systemd in a non-privileged container," explaining how things hadn't gotten much better. In that article, I stated, "Sadly, two years later if you google Docker systemd, this is still the article people see—it's time for an update." I also linked to a talk about how upstream Docker and upstream systemd would not compromise. In this article, I'll look at the progress that's been made and how Podman can help.

There are lots of reasons to run systemd inside a system, such as:

1. **Multiservice containers**—Lots of people want to take existing multi-service applications out of VMs and run them inside of containers. We would prefer that they break apart these applications into microservices, but some people can't or don't have time yet. So running them as services launched out of unit files by systemd makes sense.
2. **Systemd unit files**—Most applications that run inside of containers are built from code that was run in VMs or on host systems. These applications have a unit file that was written for the application and understands how to run the application. It can be better to launch the service via the supported method, rather than to hack up your own init service.
3. **Systemd is a process manager**—It handles the management of services like reaping, restarting, and shutting down better than any other tool.



Red Hat Documentation | AI | Learn | Documentation | Search within product

Home > Products > Red Hat Enterprise Linux > 10 > Building, running, and managing containers > Chapter 2. Types of container images

Red Hat Enterprise Linux

Filter table of contents

Building, running, and managing containers

Providing feedback on Red Hat documentation

1. Introduction to containers

2. Types of container images

Types of container images

2.1. General characteristics of RHEL container images

2.2. Characteristics of UBI images

2.4. Understanding the UBI init images

The UBI init images, named `ubi-init`, contain the systemd initialization system, making them useful for building images in which you want to run systemd services, such as a web server or file server. The init image contains more content than minimal images but less than standard images.

Because the `ubi10-init` image builds on top of the `ubi10` image, their contents are mostly the same. However, there are a few critical differences:

- `ubi10-init`:
  - CMD is set to `/sbin/init` to start the `systemd` Init service by default
  - includes `ps` and process related commands (`procps-ng` package)
  - sets `SIGRTMIN+3` as the `StopSignal`, as `systemd` in `ubi10-init` ignores normal signals to exit (`SIGTERM` and `SIGKILL`), but will terminate if it receives `SIGRTMIN+3`

https://developers.redhat.com/blog/2019/04/24/how-to-run-systemd-in-a-container
https://github.com/j8r/dockerfiles/tree/master/systemd/debian
https://systemd.io/CONTAINER_INTERFACE/
https://docs.redhat.com/en/documentation/red_hat_enterprise_linux/10/html/building_running_and_managing_containers/types-of-container-images#understanding-the-ubi-init-images

The zeek/zeek-demo repository

- Docker/Podman compose
- Zeek
- Prometheus, Grafana, Loki

UDP-based traffic mirroring using gopacket-based capture-fwd tool:

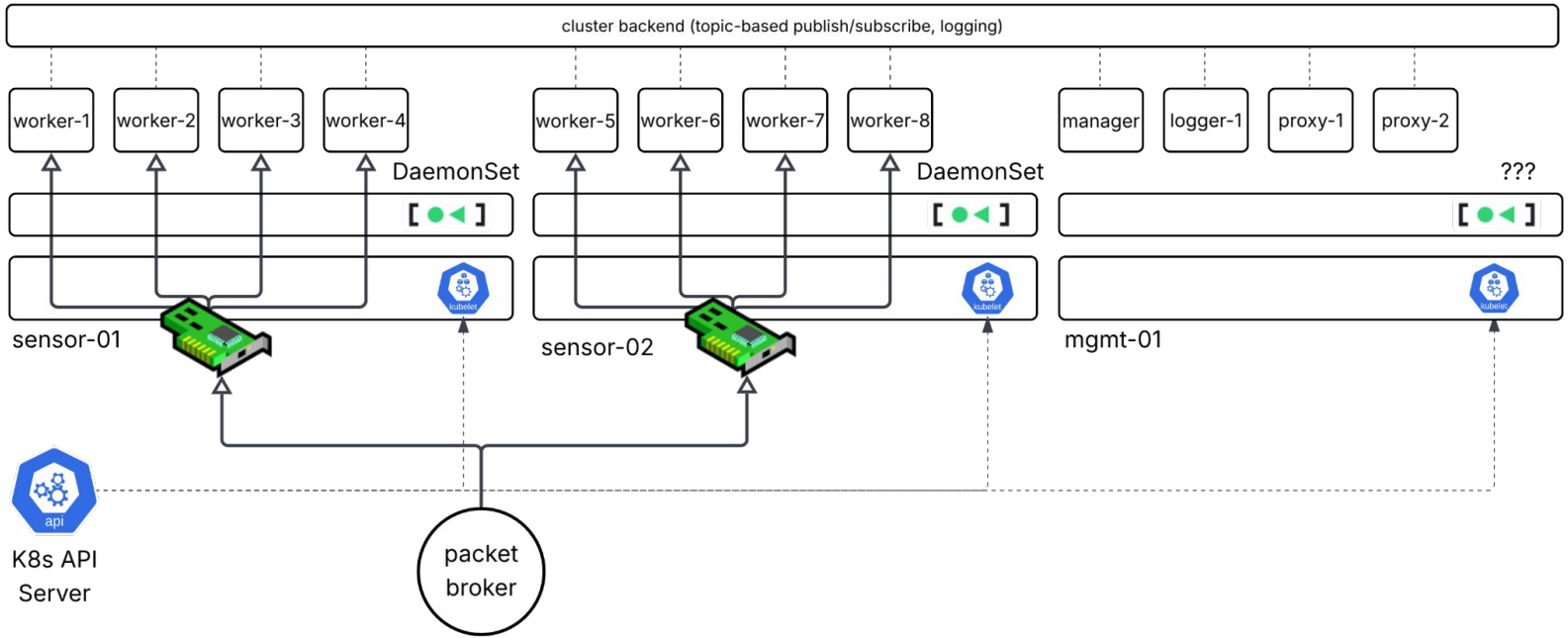
```
$ taskset -c 0 ./capture-fwd -debug \
  -i wlp0s20f3 \
  -encap vxlan \
  -destIp 127.0.0.1 \
  -destPort 4789
```

Time for a demo?

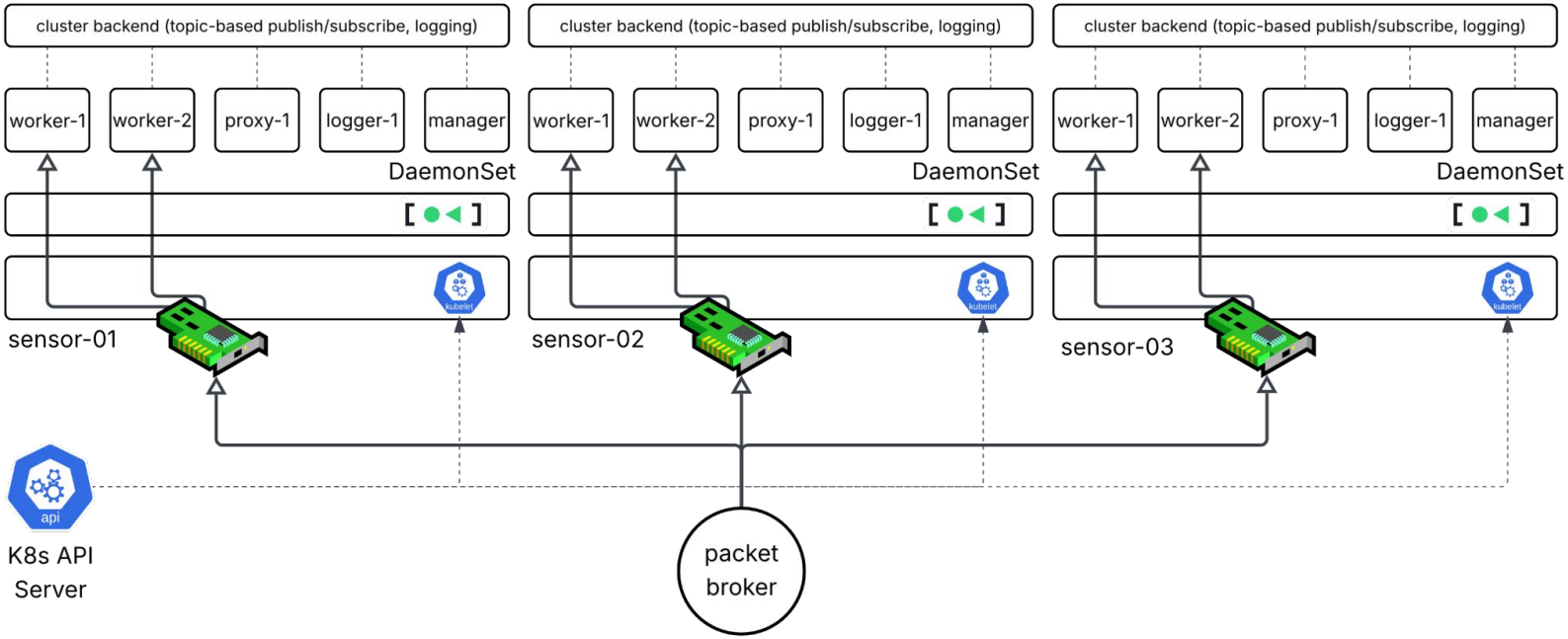
```
# docker-compose-udp/compose.yaml
services:
  zeek:
    build:
      context: ../containers/zeek-systemd-udp
      dockerfile: Dockerfile

    # The Zeek container uses systemd inside. Using privileged: true
    # is the easiest to make this "just work with Docker. The Zeek
    # processes launched by systemd are unprivileged.
    privileged: true
    stop_signal: SIGRTMIN+3
    stop_grace_period: 30s
    tty: true
    environment:
      - container=docker
    tmpfs:
      - /run
      - /tmp

    # Bind-mount the etc/zeek directory for the configuration.
    volumes:
      - ./etc/zeek:/usr/local/zeek/etc/zeek

    # Port forwarding for VXLAN encapsulated mirror traffic on the standard
    # port. By default, Docker uses a proxy process to forward packets. Put
    # the following into /etc/docker/daemon.json to use the kernel instead:
    #
    #     "userland-proxy": false
    #
    # This avoids packet reordering caused by the proxy process and its
    # threads moving between processes.
    ports:
      - "127.0.0.1:4789:4789/udp"
    networks:
      - monitoring
      - telemetry
```

Zeek Nodes as Kubernetes Nodes?

Zeek Nodes as Kubernetes Nodes?



Zeek Nodes as Kubernetes Nodes?




```
$ kubectl label node sensor-01 \
    node.zeek.org/profile=standalone
$ kubectl label node sensor-02 \
    node.zeek.org/profile=standalone
$ kubectl label node sensor-03 \
    node.zeek.org/profile=standalone
```

```
$ kubectl set image daemonset/zeek-standalone \
    zeek=ghcr.io/awelzel/zeek-dev-systemd-udp:8.1.1
```

Zeek Nodes as Kubernetes Nodes?

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: zeek-standalone-config
  namespace: zeek
data:
  zeek.conf: |
    interface = af_packet::eth0
    workers = 2

---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: zeek-standalone
  namespace: zeek
spec:
  selector:
    matchLabels:
      app: zeek-standalone
  template:
    metadata:
      labels:
        app: zeek-standalone
```

```
    spec:
      nodeSelector:
        node.zeek.org/profile: standalone

      # Allow sniffing on the host network.
      hostNetwork: true
      dnsPolicy: ClusterFirstWithHostNet

      containers:
        - name: zeek
          image:
ghcr.io/awelzel/zeek-dev-systemd-udp:latest
          command: ["/sbin/init"]
          securityContext:
            privileged: true
          volumeMounts:
            - name: zeek-config
              mountPath: /usr/local/zeek/etc/zeek/zeek.conf
              subPath:  zeek.conf
      volumes:
        - name: zeek-config
          configMap:
            name: zeek-standalone-config
```

```
$ kubectl apply -f zeek-standalone.yaml
$ kubectl label node sensor-01 node.zeek.org/profile=standalone
```

Questions / Discussion

arne.welzel@corelight.com