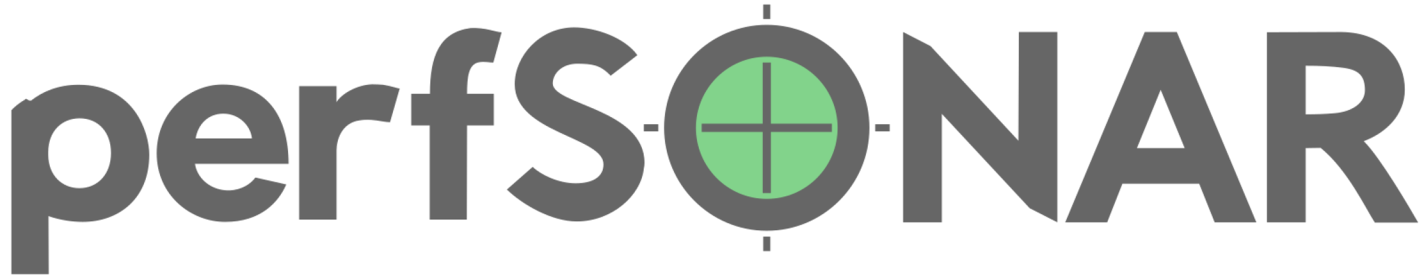


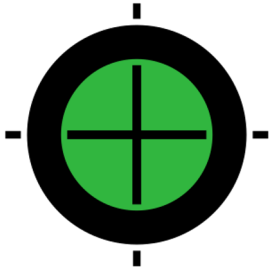
Contain Yourself!

Running perfSONAR in Containers

Mark Feit · Internet2 / perfSONAR Development Team · mfeit@internet2.edu

4th Annual European perfSONAR User Workshop

perfSONAR is developed by a partnership of



A Light Introduction to Virtualization and Containment

More Computers!

- Computing used to be expensive. *Really* expensive.
  - Time was sometimes rented – What's old is new again.

- Users like having their own sandbox.
  - Others don't affect them
  - They don't affect others

- The Unix philosophy: Do one thing, do it well

System-Level Virtualization

- Serial batch was effectively-isolated.
- Give a tenant (user) the experience of having a computer all to themselves.
- Run multiple operating systems (same or different) at once.

Virtualization Milestones

IBM Compatible Time Sharing System — 1961
- Essentially an operating system

IBM CP-40 — 1966
- Multiple Operating Systems
- Evolved into the VM/370 family

Virtualization on Microcomputers — 2000s

Containers

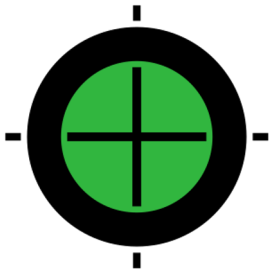
- Group of processes that run in a set of compartmented spaces on a system:

File system	User IDs
Process Control Group (cgroup) and IDs	Network
Inter-Process Communication	Time
Unix Timesharing (host name/domain)	

- Same kernel as the root namespaces
- Enough to look like a separate host without the overhead

Container Milestones

Unix `chroot`	1982
FreeBSD Jail	2000
Solaris Zones	2005
Linux Containers (LXC)	2008
Docker	2013



perfSONAR in Containers

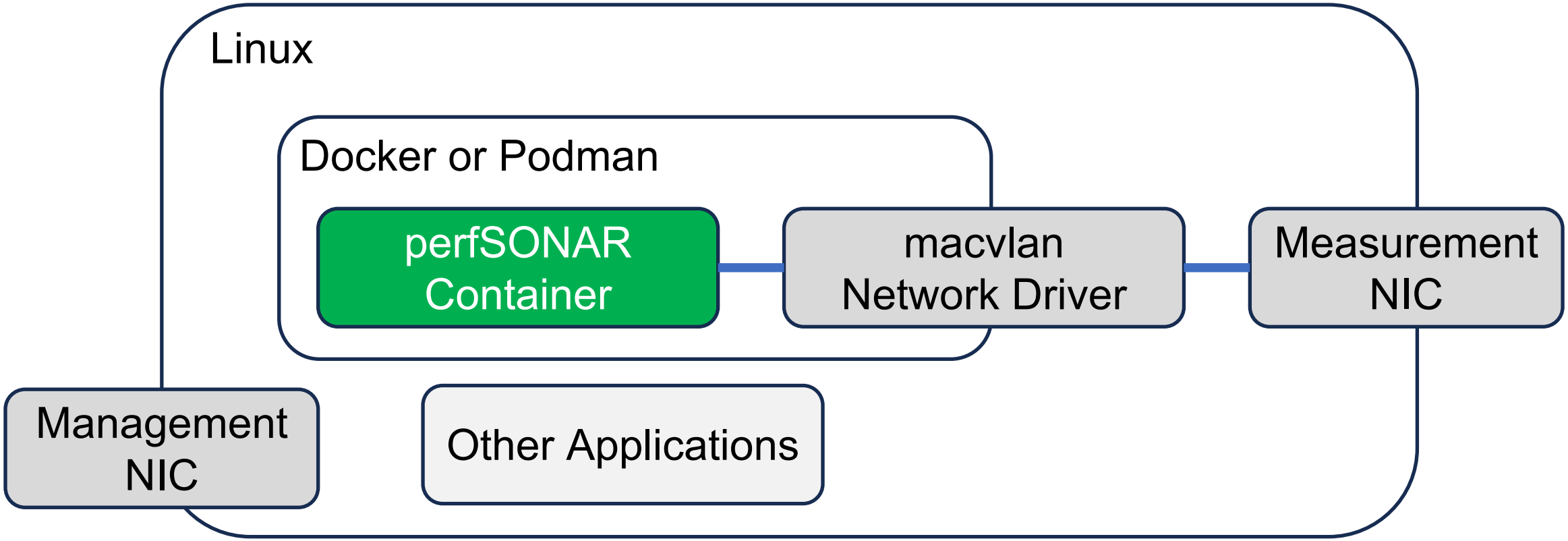
Why Run perfSONAR in a Container?

- Easy to install / rebuild / move elsewhere / delete
- Multiple, fully-isolated perfSONAR nodes on a single system
  - Solves resource management problems
- Automation
- Some security benefits
- Decouples perfSONAR's OS choices from yours (Mostly)

Why Not Run perfSONAR in a Container?

- If using more-exotic networking features (e.g., TCP congestion control algorithms), the host's kernel must support them.
- Occasional IPv6 packet loss
- Not always necessary

My Preferred System Architecture



The Container / Driver / NIC pattern may be repeated.

The `macvlan` Network Driver

- Binds a host interface directly into a container
- Bypasses additional container networking code
- Negligible performance difference vs. bare metal
- No address assigned on the host
  - Helps prevent external access to the host OS

Big MACVLAN

There Are Other Ways

- Use the default `bridge` network driver and expose ports

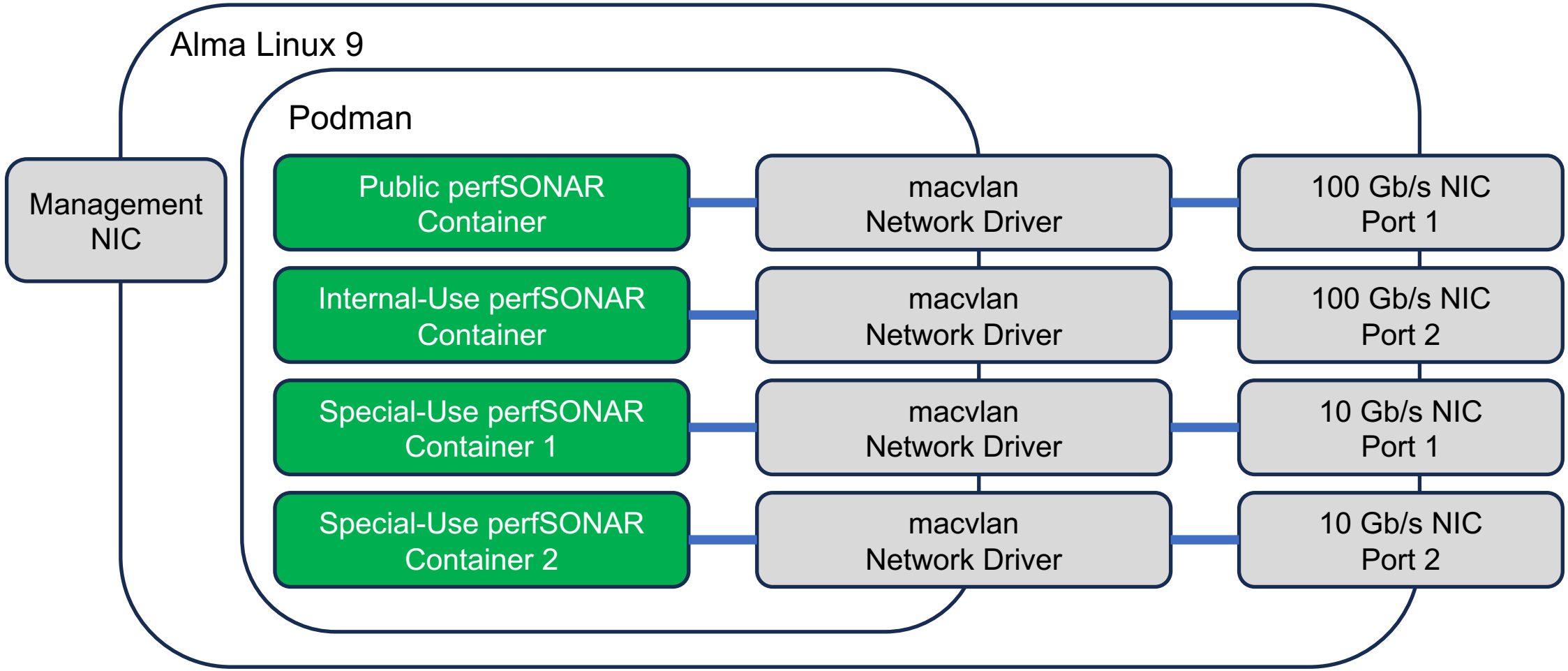
- Better when…
  - There's only one interface
  - Not sharing the host for other applications

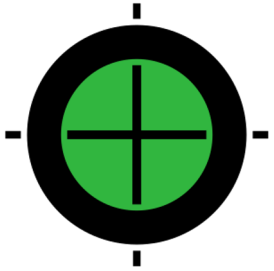
perfSONAR on Internet2's Backbone

- 49+ PoPs

- Up to four perfSONAR containers per PoP
  - Public
  - Internal-Use
  - Special uses as directed by Internet2 Network Engineering

PoP Container Architecture at Internet2





Deployment

Pick Your Environment

- Docker
- Podman

- Nothing shown will be Podman-specific

perfSONAR's Docker Image

`docker.io/perfsonar/testpoint`

`docker.io/perfsonar/testpoint:systemd`

- Either one works.

- 5.0.x – Based on CentOS 7
- 5.1.x – Based on Ubuntu 22

perfSONAR vs. the Docker Orthodoxy

- Docker favors composition, i.e., one service per container.

- Entire perfSONAR testpoint is in one container.
  - pScheduler
  - pSConfig
  - LS Registration Daemon
  - Underlying services (PostgreSQL, {O,T}WAMP, etc.)

A Word About Control Groups (Cgroups)

- Linux construct that allows a group of processes to be resource-constrained (CPU, memory, I/O, processes).

- v1 – CentOS 7, Alma/Rocky 8, Ubuntu 20
  - Containers **must** be run privileged `--privileged`
  - Setting swappiness is supported `--memory-swappiness=N`

- v2 – Alma/Rocky 9, Debian11+, Ubuntu 22+
  - Must share host's Cgroup volume into the container if contained system is v2

```
--volume /sys/fs/cgroup:/sys/fs/cgroup:ro
```

- v2 systems have `/sys/fs/cgroup/cgroup.controllers`

Managing Resources

- Processor Cores
  - How many?
  - Which ones? (Bus proximity to NIC)
- Memory
  - How much?
  - How much swap? (Usually the same as physical RAM)
- Network Interface Card

Customizations (Dockerfile)

```
FROM perfsonar/testpoint
```

pSConfig Mesh Configuration

```
RUN psconfig remote add https://mesh.example.edu/mesh.json
```

pScheduler Limit Configuration

```
COPY limits.conf /etc/pscheduler/limits.conf
RUN chown root.pscheduler /etc/pscheduler/limits.conf
RUN chmod 444 /etc/pscheduler/limits.conf
```

LS Registration Daemon Configuration (Maybe)

```
COPY lsregistrationdaemon.conf /etc/perfsonar/lsregistrationdaemon.conf
RUN chown root.perfsonar /etc/perfsonar/lsregistrationdaemon.conf
RUN chmod 444 /etc/perfsonar/lsregistrationdaemon.conf
```

Creating a Docker Network

```
docker create network
     --driver=macvlan
     --opt parent=eno3
     --subnet=192.0.2.0/24
     --gateway=192.0.2.1
     --ipv6
     --subnet=2001:db8::/32
     --gateway=2001:db8::1
     my-perfsonar-net
```

Optional: Enable IPv6

Optional: IPv6 Subnet

Optional: IPv6 Gateway

Name of Network

Running the Container

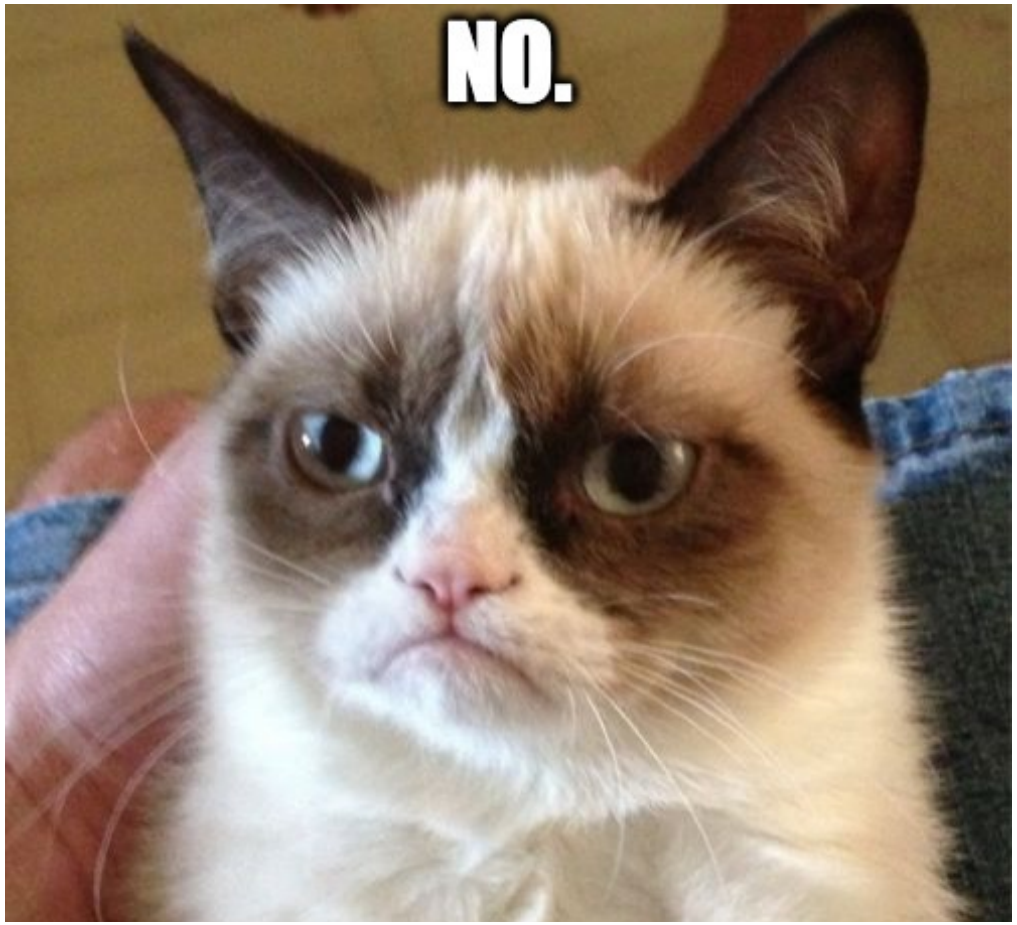
```
docker run
    --detach
    --restart=unless-stopped
    --name=my-perfsonar
    --label='My perfSONAR'
    --hostname=perfsonar.foo.org
    --cpuset-cpus=1,3,5,7
    --memory=16gb
    --memory-swap=16gb
    --memory-swappiness=0
    --network=my-perfsonar-net
    --ip=192.0.2.2
    --ip6=2001:db8::2
    CONTAINER_IMAGE
```

Pitfall: Small IPv4 Networks

- Small IPv4 networks (`/31`, `/32`) can cause difficulties.
- Docker and Podman understand them
- Modules for some supporting programs do not
  - Ansible
  - Salt Stack

Kubernetes?

- Requires tight control over network interfaces and addressing.
- Does not support the `macvlan` network driver or using existing Docker networks.
- Placement of the container can be uncertain.
- Directing traffic with load balancers will distort measurements.



Question and answer icon by iconosphere from The Noun Project

Questions and Answers

mfeit@internet2.edu