# VSOCK: VM ↔ host socket with minimal configuration

DevConf.CZ 2020

Stefano Garzarella <sgarzare@redhat.com>

Software Engineer @ Red Hat

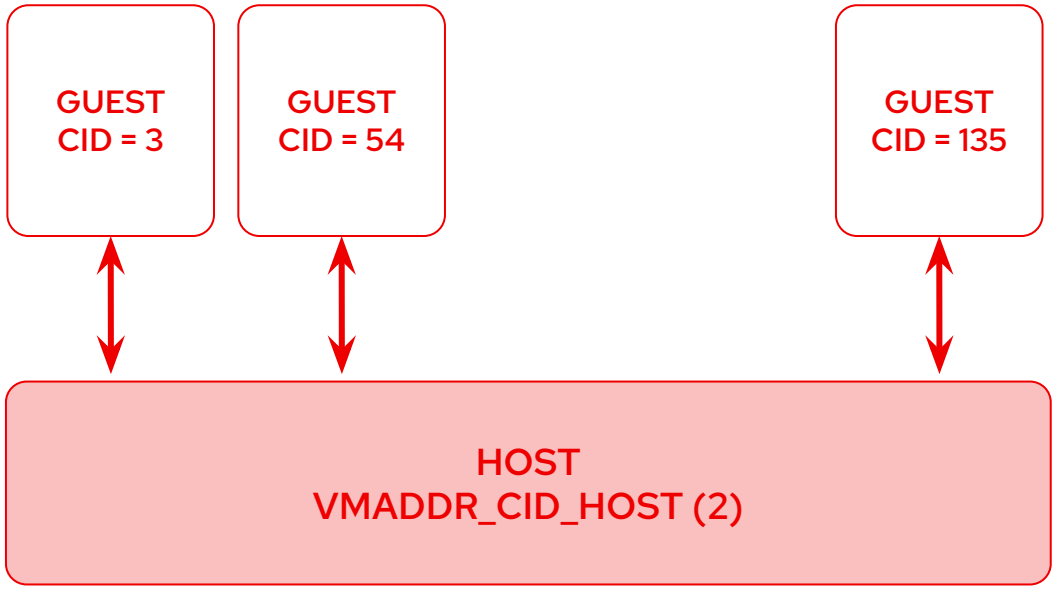


Red Hat

# Agenda

- Overview
- Use cases
- VSOCK transports
- New features
  - Multi transports (nested VMs)
  - Local communication
  - Network namespaces
- Tools supporting AF_VSOCK
- Languages providing AF_VSOCK bindings
- Demos

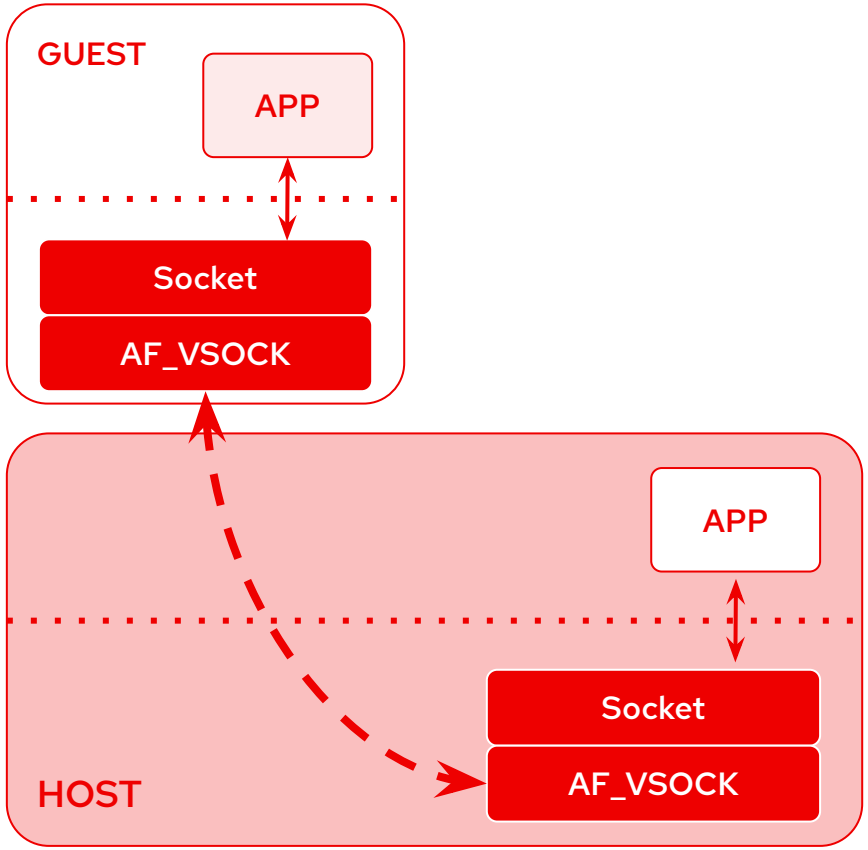
# Overview




VSOCK - **VM** **Sock**et

- POSIX Socket API
- CID - Context IDentifier
  - Peer (guest or host) identifier
  - Well defined CIDs
    - VMADDR_CID_ANY (0xFFFFFFFF)
    - VMADDR_CID_LOCAL (1)
    - VMADDR_CID_HOST (2)
- Minimal configuration
  - Host
    - assigns a CID to each guest
      - chosen by user or management tool
  - Guest
    - no configuration needed

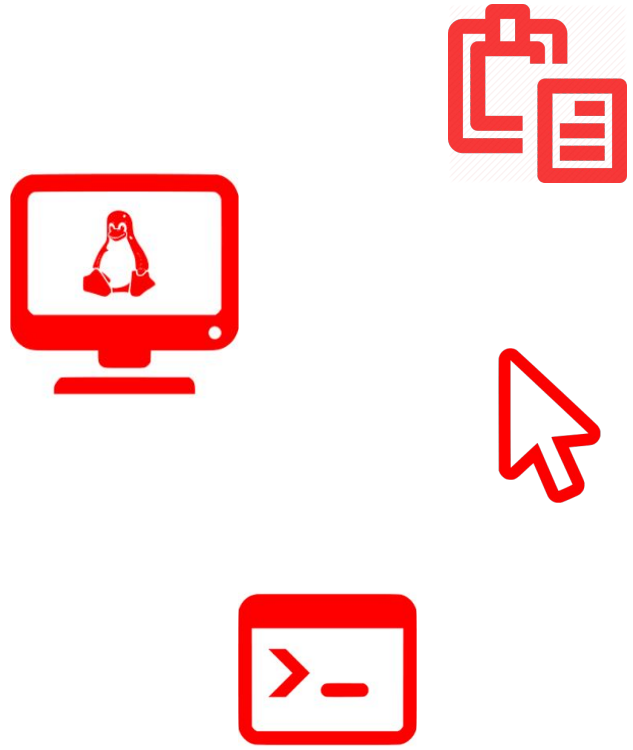
# POSIX Socket API



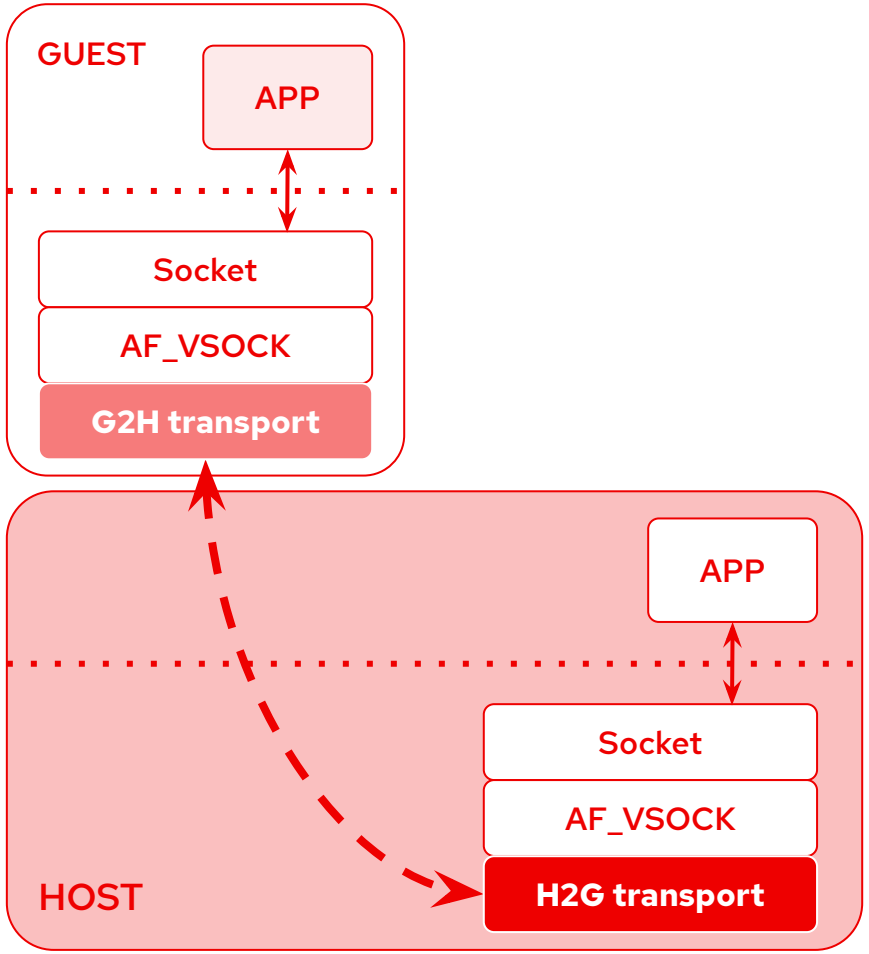

- VSOCK supports socket API
  - socket(), bind(), listen(), connect(), send(), recv(), ...
- AF_VSOCK address family
  - CID - Context IDentifier [32 bit]
  - port [32 bit]
- TCP/IP applications require few changes to be adapted
  - Peer address (AF_VSOCK address family)
    - <CID, port>
  - SOCK_STREAM and SOCK_DGRAM supported
    - SOCK_DGRAM is transport dependent

# Use cases

- Network applications
- Guest agents & Hypervisor services
  - clipboard sharing
  - mouse integration
  - automatic adjustment of video resolution
  - guest control
  - remote console
- Real cases
  - QEMU guest agent
  - Kata container agent
  - Android Debug Bridge (adb)

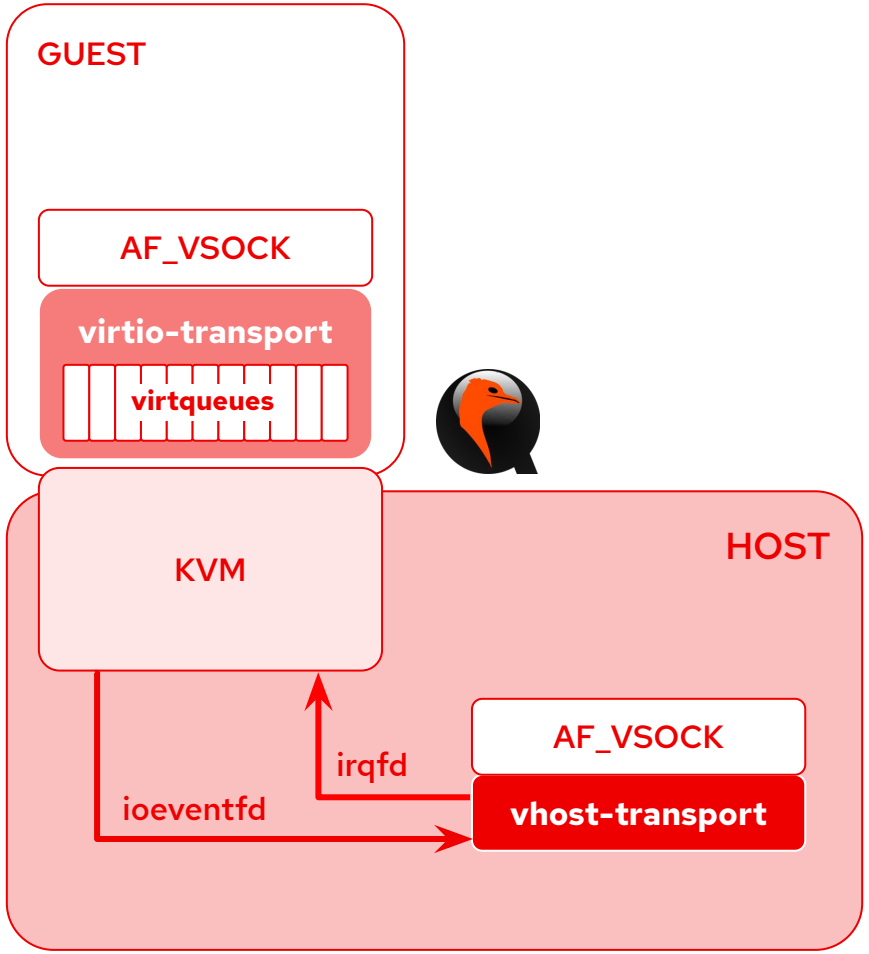


# VSOCK transports




- Implement the communication channel between guest and host
  - hypervisor-dependent
- guest → host (**G2H**) transports
  - virtio-transport
  - vmci-transport
  - hyperv-transport
- host → guest (**H2G**) transports
  - vhost-transport
  - vmci-transport

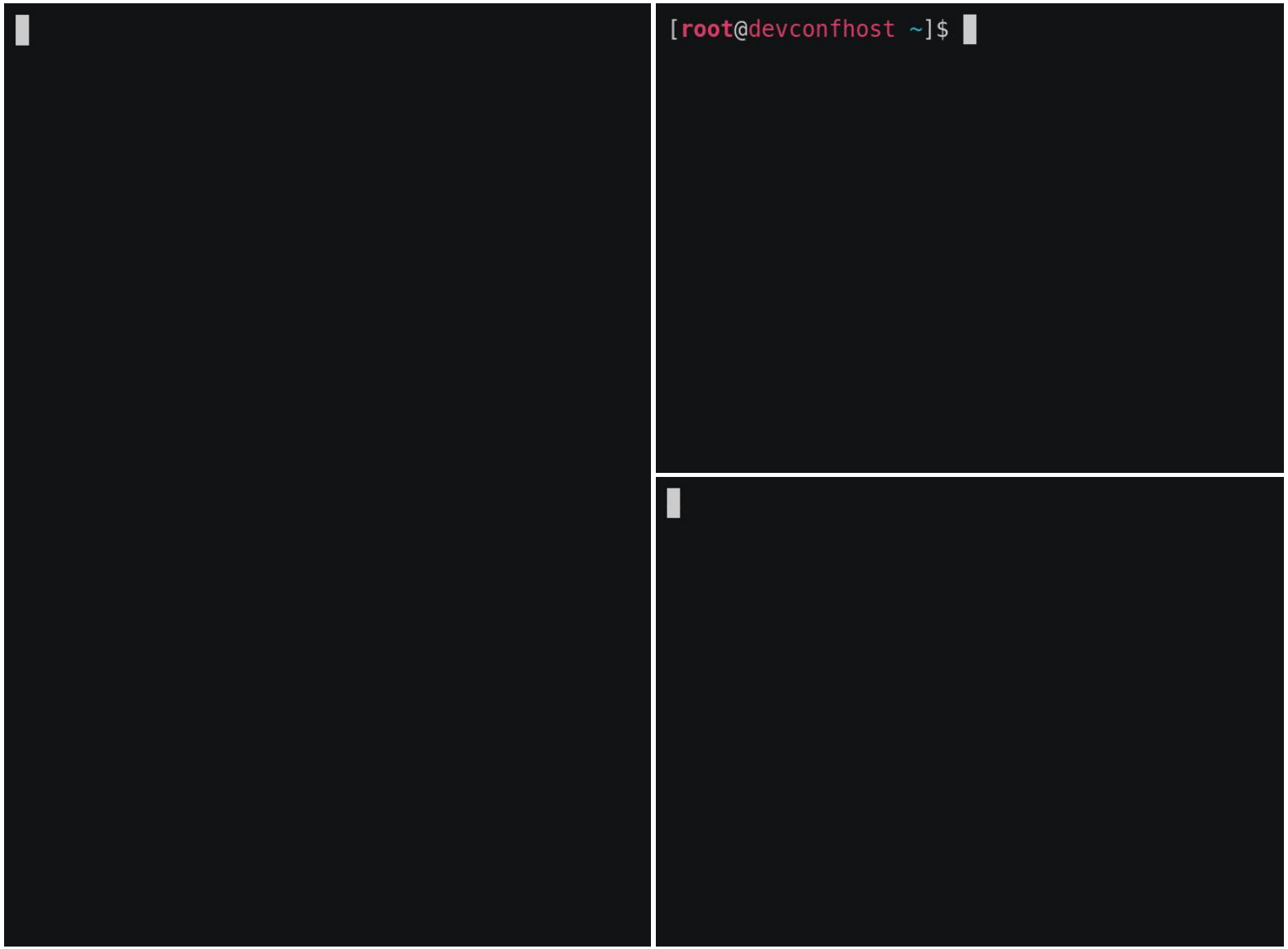
# QEMU/KVM transports




- virtio/vhost transports
  - socket layer interface (AF_VSOCK)
  - virtio-vsock device
- vhost-transport
  - in-kernel virtio-vsock device emulation
    - ioeventfd
    - irqfd
- virtio-transport
  - virtio-vsock device driver
    - virtqueues

# DEMO

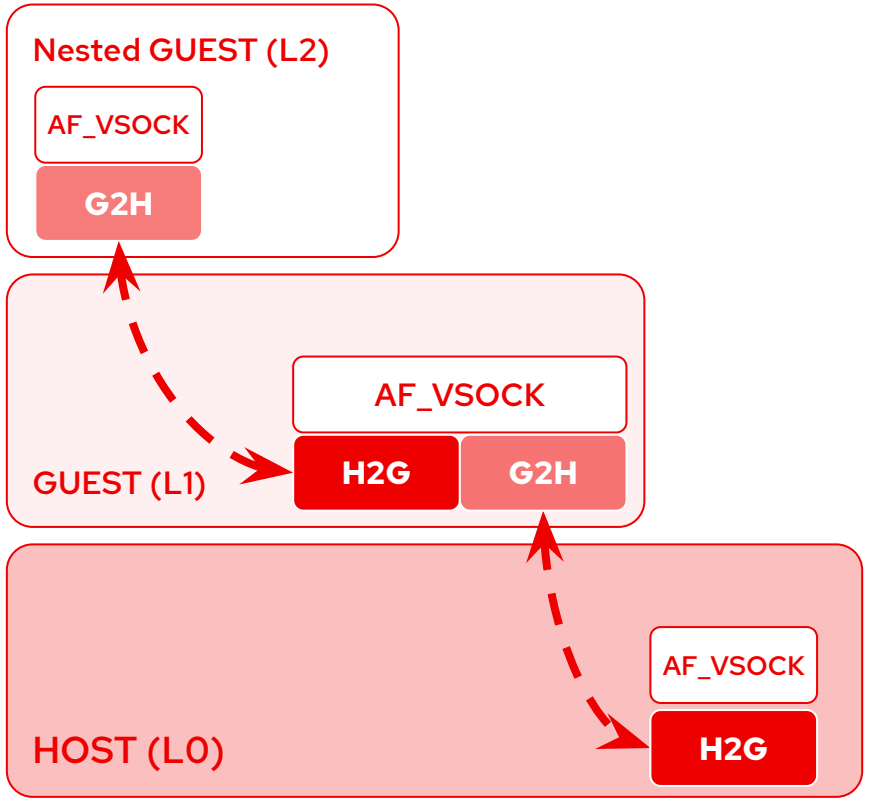





Red Hat

# Multi transports



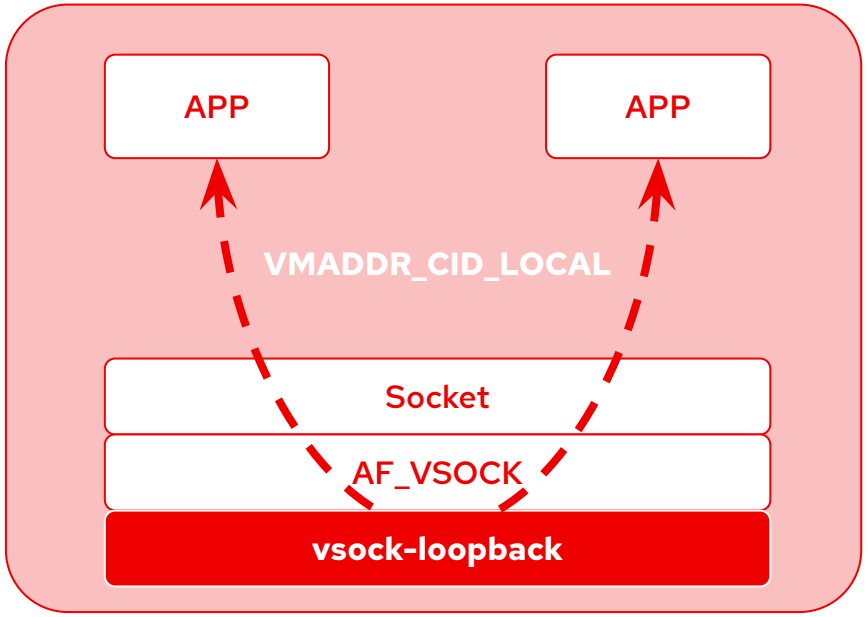

- Useful for nested virtualization
- Available from Linux v5.5
- 2 types of transports loadable together
  - host → guest (**H2G**) transport
    - vhost-transport
  - guest → host (**G2H**) transport
    - virtio-transport

```
[PATCH net-next v2 00/15] vsock: add multi-transports support
https://patchwork.ozlabs.org/cover/1194660/
```



Red Hat

# Local communication

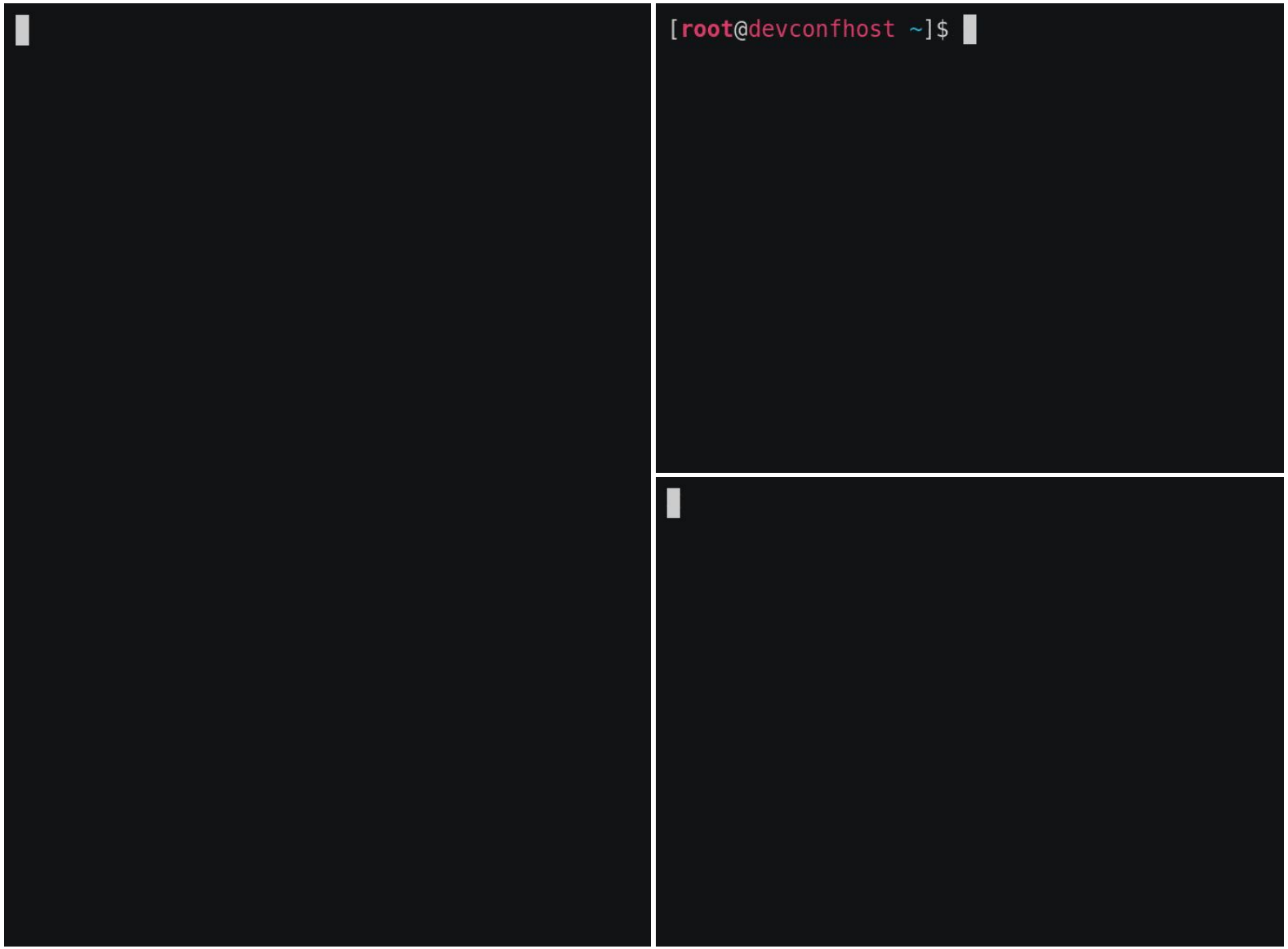



- Local communication without VMs
  - Tests
  - Debug
- **vsock-loopback**
  - New transport
  - Will be available in Linux v5.6
- CIDs
  - **VMADDR_CID_LOCAL** (1)
    - New well-know CID for loopback
  - Local Guest CID
    - If G2H is loaded
  - VMADDR_CID_HOST (2)
    - If H2G is loaded and G2H is not loaded

`[PATCH net-next v2 0/6] vsock: add local transport support`
https://patchwork.ozlabs.org/cover/1207012/

# DEMO

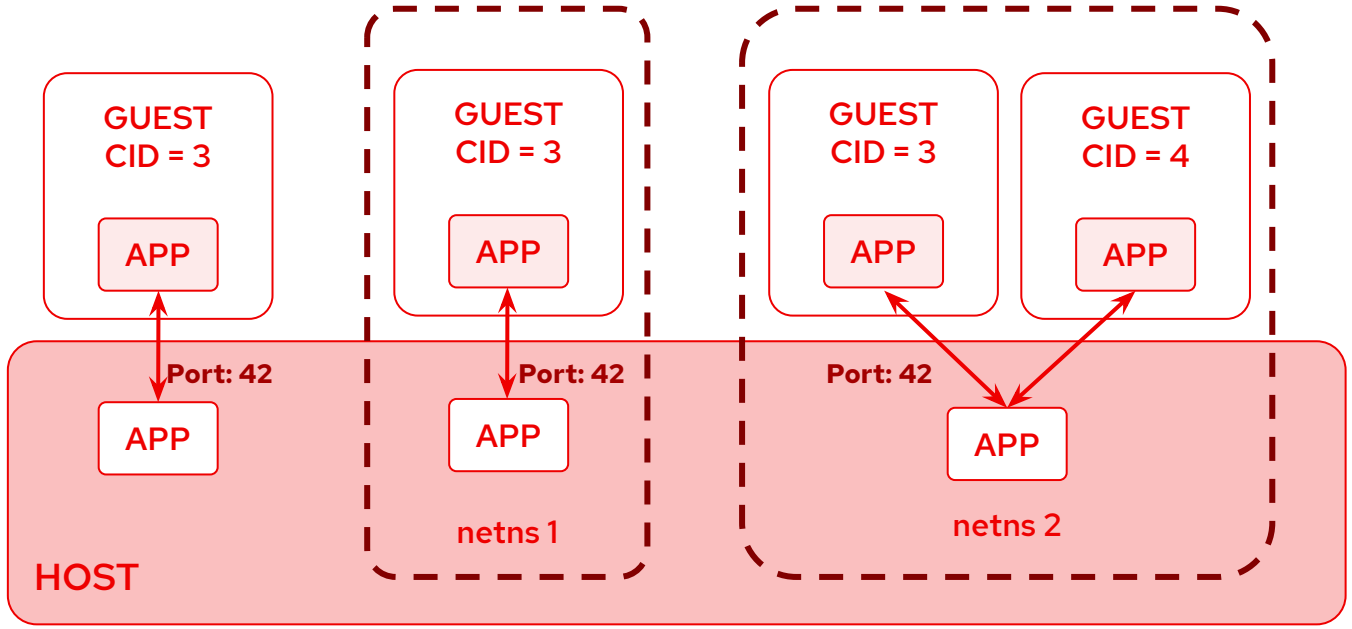





Red Hat

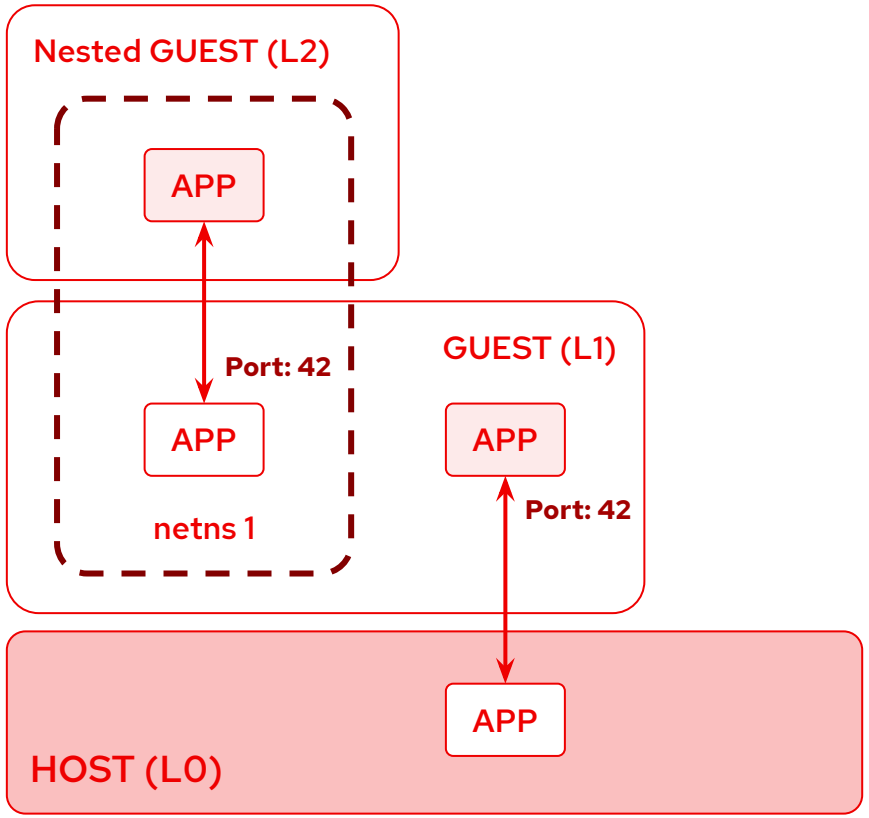
# Network namespaces (1/2)




- Work in progress
  - RFC sent
- Useful in the **host**:
  - Partition VMs
    - between VMMs
    - finer granularity
  - Assign same CID to VMs
  - Applications listening on the same port

[RFC PATCH 0/3] vsock: support network namespace
https://patchwork.ozlabs.org/cover/1202235/

# Network namespaces (2/2)

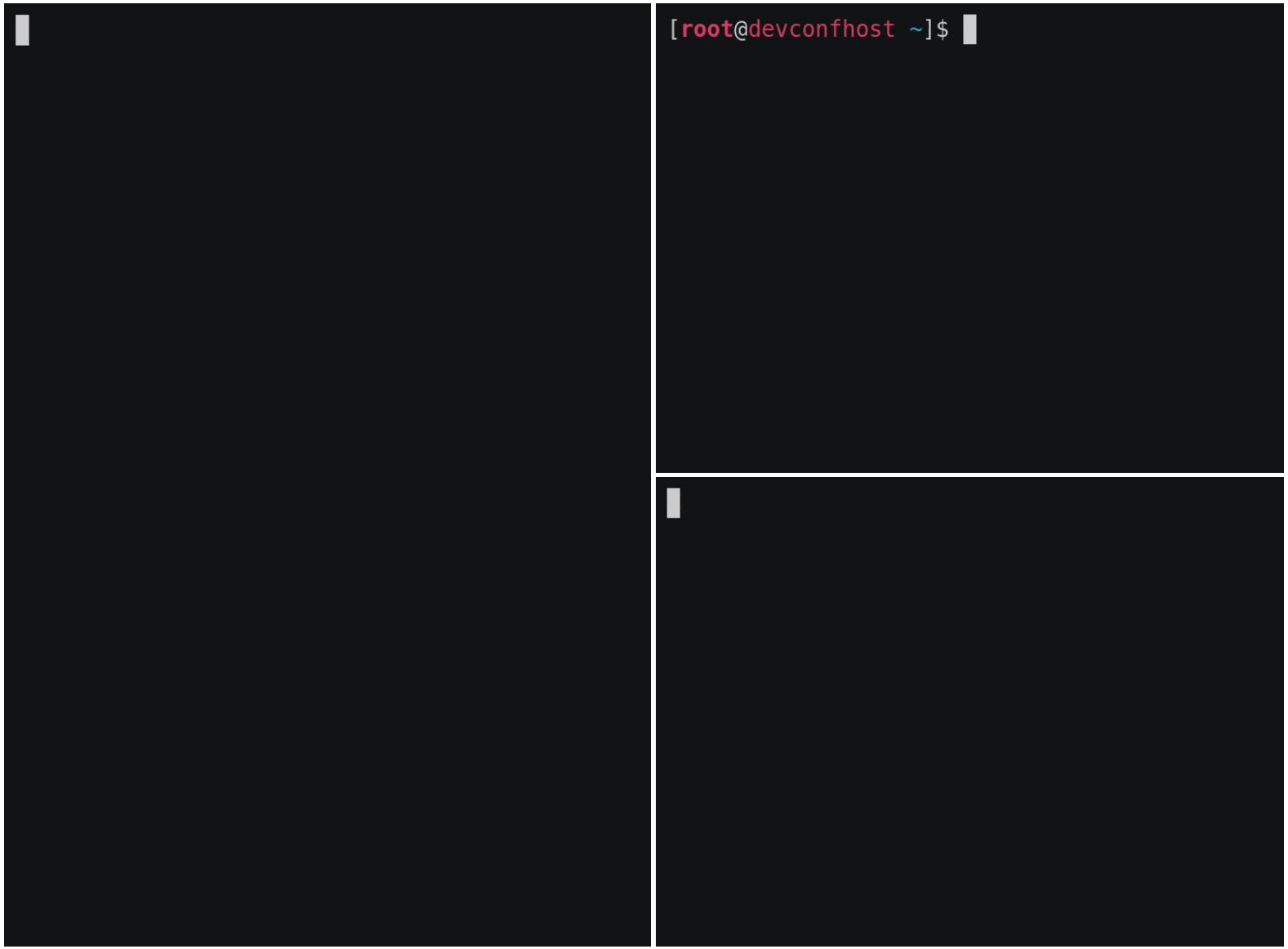



- Useful in a nested VM environment
- **L1 guest**:
  - isolate host and guest applications
    - Listening on same port
    - Listening on any CID
      - VMADDR_CID_ANY

[RFC PATCH 0/3] vsock: support network namespace
https://patchwork.ozlabs.org/cover/1202235/

# DEMO






Red Hat

# Tools supporting AF_VSOCK

- **wireshark** >= 2.40 [2017-07-19]
- **iproute2** >= 4.15 [2018-01-28]
  - ss
- **tcpdump**
  - merged in master [2019-04-16]
- **nmap** >= 7.80 [2019-08-10]
  - ncat
- **nbd**
  - nbdkit >= 1.15.5 [2019-10-19]
  - libnbd >= 1.1.6 [2019-10-19]
- **iperf-vsock**
  - iperf3 fork
  - https://github.com/stefano-garzarella/iperf-vsock
- **socat-vsock**
  - socat fork
  - https://github.com/stefano-garzarella/socat-vsock

# Languages providing AF_VSOCK bindings

- **C**
  - glibc >= 2.18 [2013-08-10]
- **Python**
  - python >= 3.7 alpha 1 [2017-09-19]
- **Golang**
  - https://github.com/mdlayher/vsock
- **Rust**
  - libc crate >= 0.2.59 [2019-07-08]
    - struct sockaddr_vm
    - VMADDR_* macros
  - nix crate >= 0.15.0 [2019-08-10]
    - VSOCK supported in the socket API (nix::sys::socket)

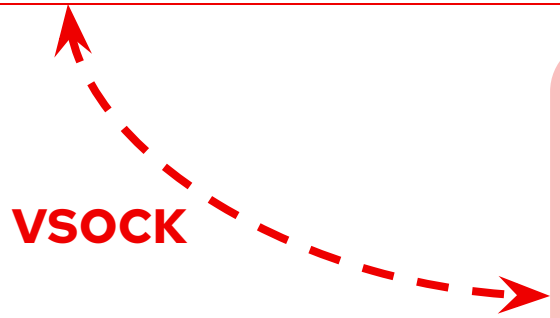
# Python example

**GUEST**

```
# Client running in the guest
import socket

s = socket.socket(socket.AF_VSOCK, socket.SOCK_STREAM)
s.connect((socket.VMADDR_CID_HOST, 1234))

s.send(b'Hello, world')
```




**HOST**

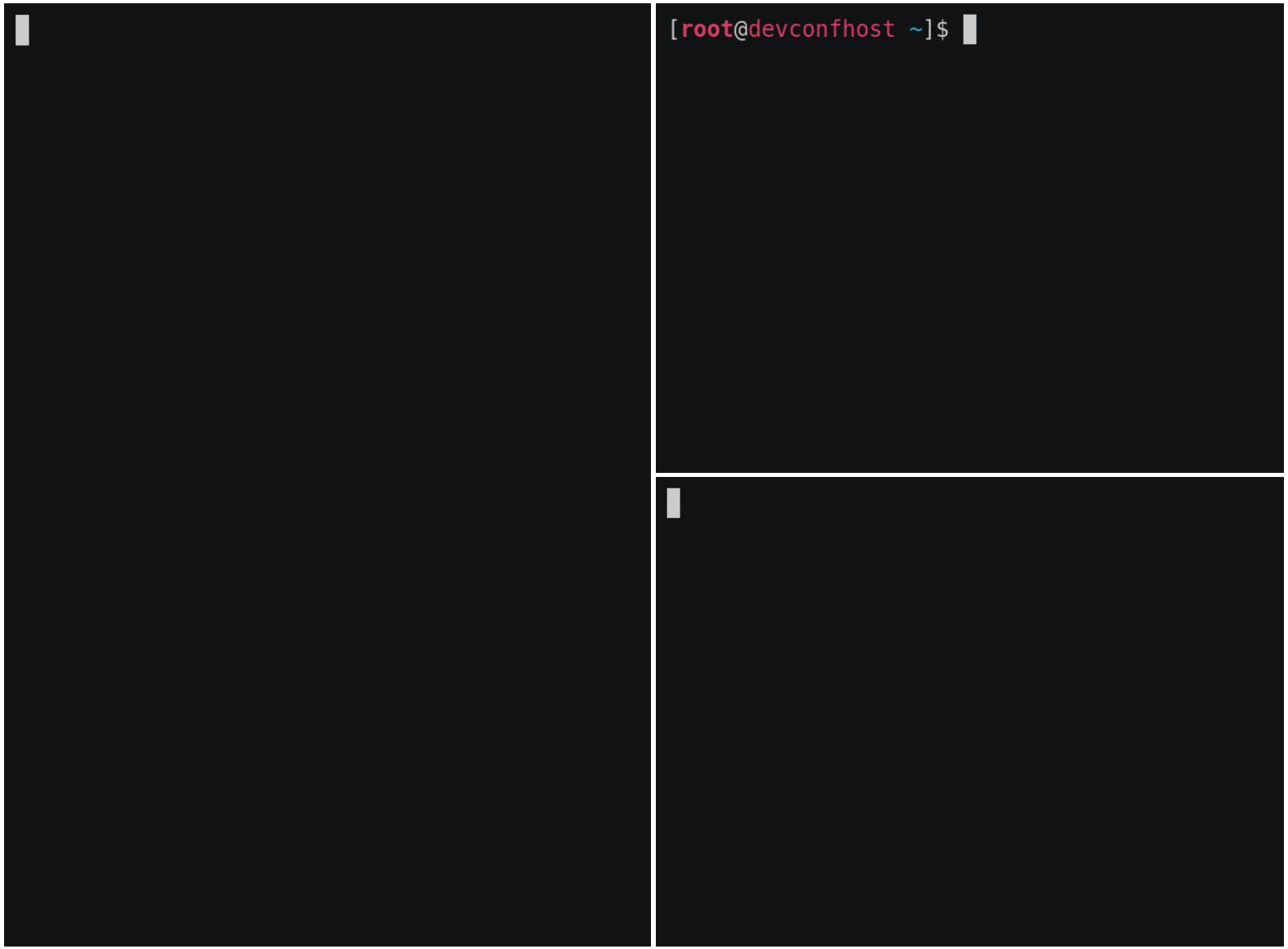
```
# Server running in the host
import socket

s = socket.socket(socket.AF_VSOCK, socket.SOCK_STREAM)
s.bind((socket.VMADDR_CID_ANY, 1234))
s.listen()
client, addr = s.accept()

data = client.recv(1024)
print("CID: {} port:{} data: {}".format(addr[0], addr[1], data))
```

# DEMO






Red Hat

# Conclusions

- VSOCK is very useful for a point to point connection between guests and host
- Existing TCP/IP applications can be easily adapted
- Several tools and languages support VSOCK
- Nested VMs (Linux v5.5)
- Loopback (Linux v5.6)
- Network namespace (available soon)

# Thank you!

Stefano Garzarella <**sgarzare@redhat.com**>

Blog: https://**stefano-garzarella.github.io/**

IRC: **sgarzare** on #qemu irc.oftc.net

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat



Red Hat

# Demos

# Host-Guest communication demo

```
devconfhost$ virt-install --name devconfguest --ram 2048
--vcpus 2 --import --os-variant fedora31 --disk
path=/images/devconfguest.qcow2,bus=virtio --vsock
cid.address=33 --graphics none

... VM booting ...

devconfguest$ nc -v --vsock 2 1234
Ncat: Version 7.80SVN ( https://nmap.org/ncat )
Ncat: Connection to 2.

Hey, I'm the guest!
devconfguest$ nc -v --vsock -l 1234
Ncat: Version 7.80SVN ( https://nmap.org/ncat )
Ncat: Listening on 4294967295:1234
Ncat: Connection from a client on vsock socket.
Ncat: Connection from 2:130178752.

Hey, I'm the host!
```

```
devconfhost$ echo "Host-Guest communication"
Host-Guest communication
devconfhost$ nc -v --vsock -l 1234
Ncat: Version 7.80SVN ( https://nmap.org/ncat )
Ncat: Listening on 4294967295:1234
Ncat: Connection from a client on vsock socket.
Ncat: Connection from 33:33825030.

Hey, I'm the guest!
devconfhost$ nc -v --vsock 33 1234
Ncat: Version 7.80SVN ( https://nmap.org/ncat )
Ncat: Connection to 33.

Hey, I'm the host!
```

# Local communication demo

```
devconfhost$ echo "Local communication"
Local communication
devconfhost$ nc -v --vsock -l 1234
Ncat: Version 7.80SVN ( https://nmap.org/ncat )
Ncat: Listening on 4294967295:1234
Ncat: Connection from a client on vsock socket.
Ncat: Connection from 1:130178753.

Hey, we are local!
devconfhost$ nc -v --vsock -l 1234
Ncat: Version 7.80SVN ( https://nmap.org/ncat )
Ncat: Listening on 4294967295:1234
Ncat: Connection from a client on vsock socket.
Ncat: Connection from 1:130178754.

Hey, we are local again!
```

```
devconfhost$ nc -v --vsock 1 1234
Ncat: Version 7.80SVN ( https://nmap.org/ncat )
Ncat: Connection to 1.

Hey, we are local!
devconfhost$ nc -v --vsock 2 1234
Ncat: Version 7.80SVN ( https://nmap.org/ncat )
Ncat: Connection to 2.

Hey, we are local again!
```

# Network namespace demo

```
devconfhost$ echo "Network namespace"
Network namespace
devconfhost$ ip netns add ns1
devconfhost$ ip netns exec ns1 nc -v --vsock -l 1234
Ncat: Version 7.80SVN ( https://nmap.org/ncat )
Ncat: Listening on 4294967295:1234
Ncat: Connection from a client on vsock socket.
Ncat: Connection from 33:2230749232.

Hey, we are on ns1
```

```
devconfhost$ ip netns exec ns1 qemu-system-x86_64
-machine accel=kvm -cpu host -smp 2 -m 1G -drive
file=/images/devconfguest2.qcow2,if=virtio -device
vhost-vsock-pci,guest-cid=33 --nographic

... VM booting ...

devconfguest2$ nc -v --vsock 2 1234
Ncat: Version 7.80SVN ( https://nmap.org/ncat )
Ncat: Connection to 2.

Hey, we are on ns1
```

# Python example demo

```
devconfguest$ python
Python 3.7.5 (default, Dec 15 2019, 17:54:26)
[GCC 9.2.1 20190827 (Red Hat 9.2.1-1)] on linux
Type "help", "copyright", "credits" or "license" for
more information.
>>> import socket
>>> s = socket.socket(socket.AF_VSOCK,
socket.SOCK_STREAM)
>>> s.connect((socket.VMADDR_CID_HOST, 1234))
>>> s.send(b'Hello, world')
12
>>>
```

```
devconfhost$ echo "Python example"
Python example
devconfhost$ python
Python 3.7.5 (default, Dec 15 2019, 17:54:26)
[GCC 9.2.1 20190827 (Red Hat 9.2.1-1)] on linux
Type "help", "copyright", "credits" or "license" for
more information.
>>> import socket
>>> s = socket.socket(socket.AF_VSOCK,
socket.SOCK_STREAM)
>>> s.bind((socket.VMADDR_CID_ANY, 1234))
>>> s.listen()
>>> client, addr = s.accept()
>>> data = client.recv(1024)
>>> print("CID: {} port:{} data: {}".format(addr[0],
addr[1], data))
CID: 33 port:33825032 data: b'Hello, world'
>>>
```