Cross-platform/cross-hypervisor
virtio vsock use in go

Usermode networking in CodeReady Containers

Christophe Fergeau
Senior Software Engineer
Introduction

- Christophe Fergeau <cforgeau@redhat.com>
- Working at Red Hat
- Member of the CodeReady Containers team
- Previously worked in the virtualization team (SPICE)
What we’ll discuss today

- CodeReady Containers (aka crc)
- User-mode networking
- vsock usage in go
  - Linux
  - macOS
  - Windows
CodeReady Containers
What is CodeReady Containers?

▸ Runs a Red Hat OpenShift 4 cluster on your laptop or desktop
  • "Red Hat® OpenShift® is an enterprise-ready Kubernetes container platform built for an open hybrid cloud strategy."
▸ Meant for development and testing on a throw-away local cluster
▸ Works on Linux, macOS and Windows
Under the hood

- Go binary + pre-generated virtual machine image
- Uses native hypervisors
  - QEMU+KVM on linux
  - HyperKit on macOS
  - Hyper-V on Windows
- User-mode stack for VM networking
User-mode networking
Why?

- Simplifies VM networking
- Consistent IP addressing
- Works around strict firewalls/VPNs
gvisor-tap-vsock

- https://github.com/containers/gvisor-tap-vsock
- Users:
  - crc
  - podman-machine
- Based on gVisor
  - "gVisor is an application kernel, written in Go, that implements a substantial portion of the Linux system call interface."
How does it work?
How does it work? (2)

- Need code for this on Linux, macOS, Windows
- More details regarding the networking in the containers devroom tomorrow!
- This presentation will focus on the virtio-vsock communication
Using virtio-vsock on multiple platforms
What is virtio-vsock?

- [https://wiki.qemu.org/Features/VirtioVsock](https://wiki.qemu.org/Features/VirtioVsock)
- POSIX Sockets API which allows applications in the guest and host to communicate
- Host/VMs each have their own CID (“address”)
- Applications connect to/listen on a given port
Linux - How to use virtio-vsock?

- Configure the VM to have a vsock device
- On the host:
  - load the vhost_vsock kernel module
  - /dev/vsock must be accessible by the user

```
[host]$ ncat --vsock --listen 2222
hello host!
hello vm!
```

```
[vm]$ ncat --vsock 2 2222
hello host!
hello vm!
```
Linux - virtual machine configuration

```xml
<vsock model='virtio'>
  <cid auto='yes'/>
</vsock>
```

```go
import "libvirt.org/go/libvirtxml"

func addVSock(domain *libvirtxml.Domain) {
  domain.Devices.VSock = &libvirtxml.DomainVSock{
    Model: "virtio",
    CID: &libvirtxml.DomainVSockCID{
      Auto: "yes",
    },
  }
}
```
Linux - Communication over virtio-vsock

- Low-level AF_VSOCK support in go
- But external package needed for vsock implementations of net.Conn and net.Listener
  - import "github.com/mdlayher/vsock"
package main

import {
    "fmt"
    "io"
    "os"
    "github.com/mdlayher/vsock"
}

func main() {
    cid := vsock.Host
    port := 1234

    netConn, err := vsock.Dial(uint32(cid), uint32(port))
    if err != nil {
        panic("failed to dial")
    }
    defer netConn.Close()

    fmt.Printf("client: %s\n", netConn.LocalAddr())
    fmt.Printf("server: %s\n", netConn.RemoteAddr())

    if _, err := io.Copy(netConn, os.Stdin); err != nil {
        panic("failed to send data")
    }
}
package main

import {
    "io"
    "os"
    "github.com/mdlayher/vsock"
}

func main() {
    port := 1234

    listener, err := vsock.Listen(uint32(port))
    if err != nil {
        panic("failed to listen")
    }
    defer listener.Close()

    // Accept a single connection, and receive stream from that connection.
    conn, err := listener.Accept()
    if err != nil {
        panic("failed to accept")
    }
    defer conn.Close()

    if _, err := io.Copy(os.Stdout, conn); err != nil {
        panic("failed to receive data")
    }
}
macOS

- hyperkit ... -s $pciSlot,virtio-sock,guest_cid=3,path=~/.hyperkit
- vsock communication will happen over the unix socket located at:
  fmt.Sprintf("~/hyperkit/%08x.%08x", cid, port)
- If using port 1234 with cid 3: ~/hyperkit/00000003.000004d2
package main

import {
    "fmt"
    "net"
    "path/filepath"
}

func Listen(cid, port int) (net.Listener, error) {
    hyperkitStateDir := "/Users/teuf/.hyperkit"
    vsockSocket := fmt.Sprintf("%08x.%08x", cid, port)

    return net.Listen("unix", filepath.Join(hyperkitStateDir, vsockSocket))
}
Windows

- Uses the `hf_vsock` kernel module in the Linux VM
- Communication goes over the native Hyper-V VMBus
- The go code running on the Windows host uses [https://github.com/linuxkit/virtsock/tree/master/pkg/hvsock](https://github.com/linuxkit/virtsock/tree/master/pkg/hvsock)
- This package provides `net.Conn` and `net.Listener` implementations for Hyper-V Sockets
Windows

- vssock communication must be explicitly enabled in the registry

- `HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\Virtualization\GuestCommunicationServices\` must contain a key named following the well-known GUID template `HV_GUID_VSOCK_TEMPLATE 00000000-FACB-11E6-BD58-64006A7986D3`

- [https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/user-guide/make-integration-service](https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/user-guide/make-integration-service)

- [https://github.com/tpn/winsdk-10/blob/9b69fd26ac0c7d0b83d378dba01080e93349c2ed/Include/10.0.16299.0/shared/hvsocket.h#L123-L147](https://github.com/tpn/winsdk-10/blob/9b69fd26ac0c7d0b83d378dba01080e93349c2ed/Include/10.0.16299.0/shared/hvsocket.h#L123-L147)
package main

import {
    "net"
    "github.com/linuxkit/virtsock/pkg/hvsock"
}

func Listen() (net.Listener, error) {
    svcid, err := hvsock.GUIDFromString("00000400-FA9C-11E6-BD58-64006A7986E3")
    if err != nil {
        return nil, err
    }
    return hvsock.Listen(hvsock.Addr{
        VMID: hvsock.GUIDWildcard,
        ServiceID: svcid,
    })
}
Going even further…

- Using systemd socket activation together with vsock
- Using vsock with a 4th hypervisor, macOS native virtualization framework
  - M1 support!
- Running Podman containers on macOS and Windows
Useful links

- CodeReady Containers: [https://github.com/code-ready/crc/](https://github.com/code-ready/crc/)
- gvisor-tap-vsock: [https://github.com/containers/gvisor-tap-vsock](https://github.com/containers/gvisor-tap-vsock)
- Contact information: [cfergeau@redhat.com](mailto:cfergeau@redhat.com)
Thank you

Red Hat is the world’s leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat