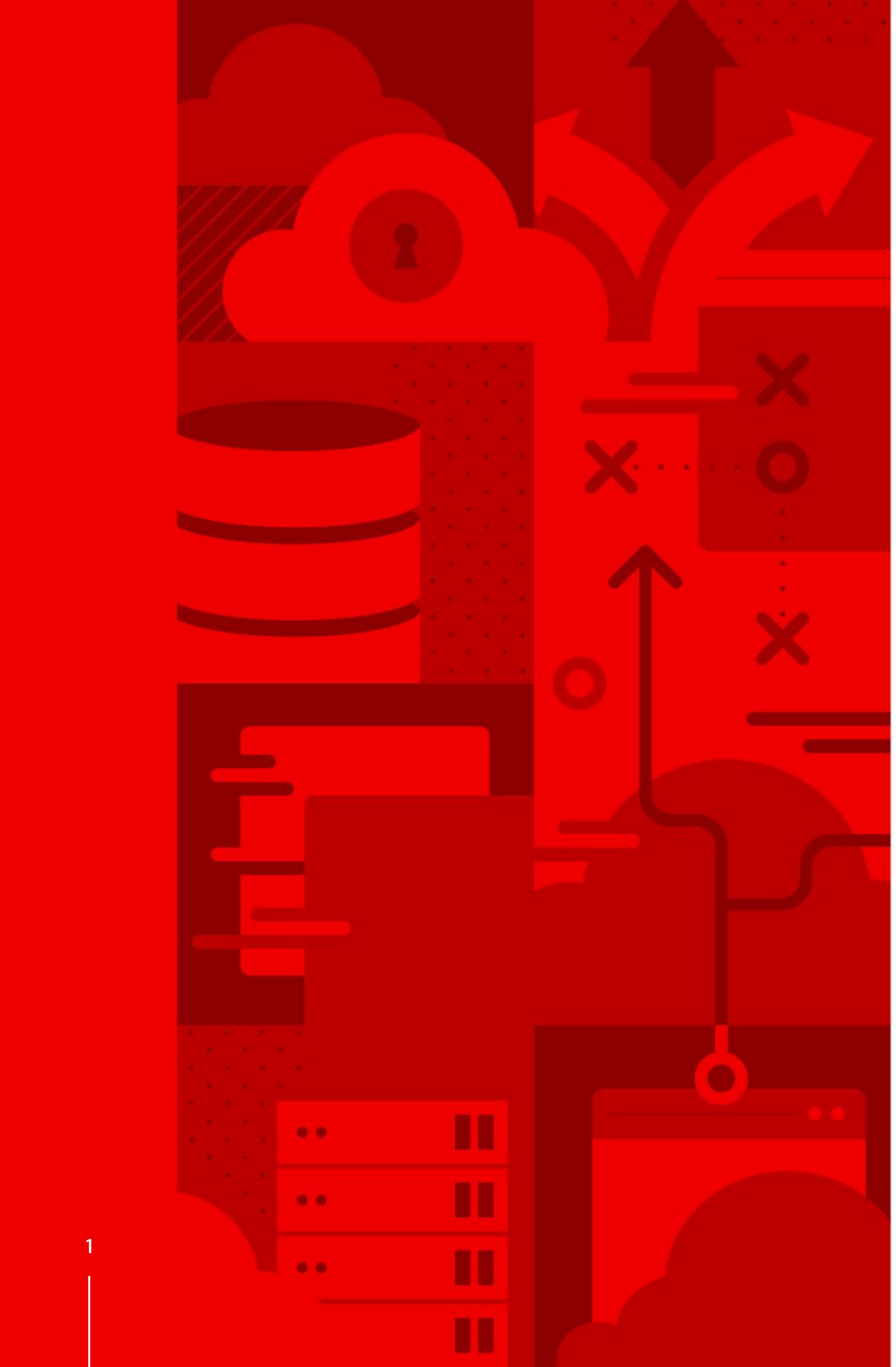


vfkit

A macOS hypervisor using Apple's virtualization framework

Christophe Fergeau
Senior Software Engineer

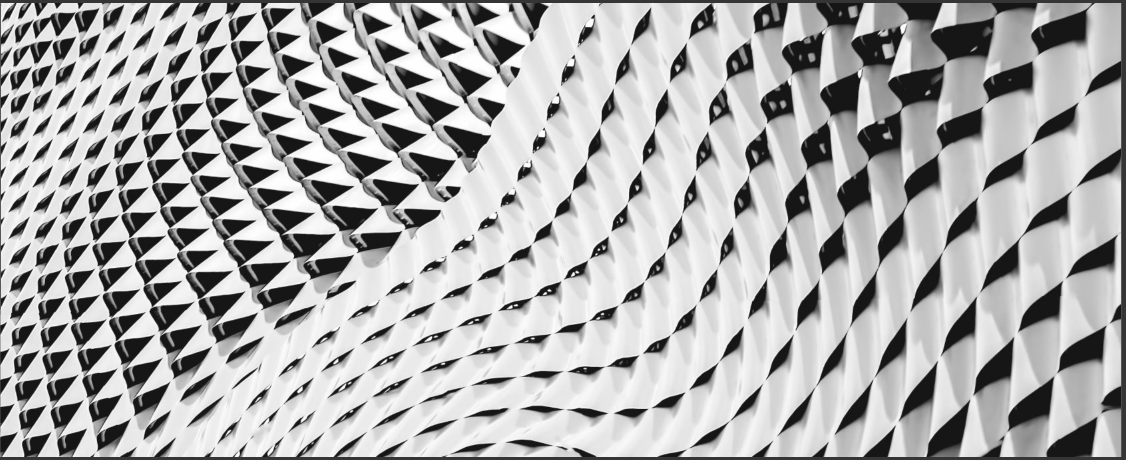
About me

- Christophe Fergeau
- Senior Software Engineer at Red Hat
- Working in the CRC/OpenShift Local team
- GitHub: @cfergeau

What we’ll discuss today

- Why is an hypervisor needed to run containers?
- Apple’s Virtualization Framework
- vfkit

Background



A hypervisor to run containers?

- Containers are images with linux binaries:

```
$ podman run registry.access.redhat.com/ubi9  file -L /bin/sh

/usr/bin/sh: ELF 64-bit LSB pie executable, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2,
BuildID[sha1]=5d82a44f2a4466ff21f763af86b004d1fcb3a8f1, for GNU/Linux
3.2.0, stripped
```

- Can be run natively on linux
- But containers can also be run on macOS and Windows

podman-machine

- https://github.com/containers/podman
- https://docs.podman.io/en/latest/markdown/podman-machine.1.html
- Used by podman to run linux containers on macOS (and Windows)
- The containers run in a virtual machine

podman-machine

```
$ podman machine init --now

…

Waiting for VM ...

…

$ podman run --rm hello-world
```

CRC - OpenShift Local

- https://github.com/crc-org/crc/
- Runs a local OpenShift cluster on a macOS/Windows/Linux machine
  - Red Hat OpenShift is a Kubernetes distribution
- The cluster runs in a virtual machine

Why the need for a new hypervisor?

macOS Hypervisors (free software / command line)

- HyperKit
- QEMU
- ??

Why the need for a new hypervisor?

macOS Hypervisors (free software / command line)

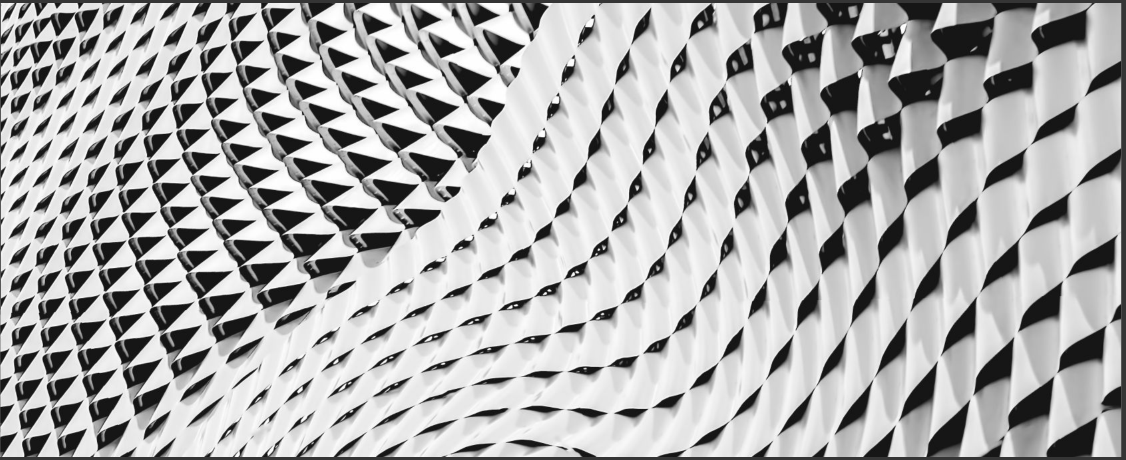
- ~~HyperKit~~ No support for Apple Silicon ARM CPUs
- QEMU
- ??

Why the need for a new hypervisor?

macOS Hypervisors (free software / command line)

- ~~HyperKit~~ No support for Apple Silicon ARM CPUs
- ~~QEMU~~ Millions of lines of C code, and we would need to maintain our own builds, track CVEs, …
- ??

Apple’s Virtualization Framework



Apple’s Virtualization Framework

- https://developer.apple.com/documentation/virtualization
- Available in macOS 11 and newer
- High-level API to create Linux and macOS virtual machines
- Can be used from Swift or Objective-C
- Just an API/a framework (library), not an end-user application

Apple's Virtualization Framework

- Only supports devices needed in virtual machines (virtio)
- virtio-net for network communication
- virtio-blk for disk image handling
- virtio-serial, virtio-rng, virtio-balloon, …
- Supports graphical VMs (video, sound, keyboard, pointing devices, …)

Apple’s Virtualization Framework

- virtio-fs for file sharing between the host and the guest
- virtio-vsock for communication between the host and the guest using POSIX sockets

Can it run containers?

- Can run modern linux distributions (UEFI support)
- Networking support
- File sharing support for efficient `--volume` support

Can it run containers?

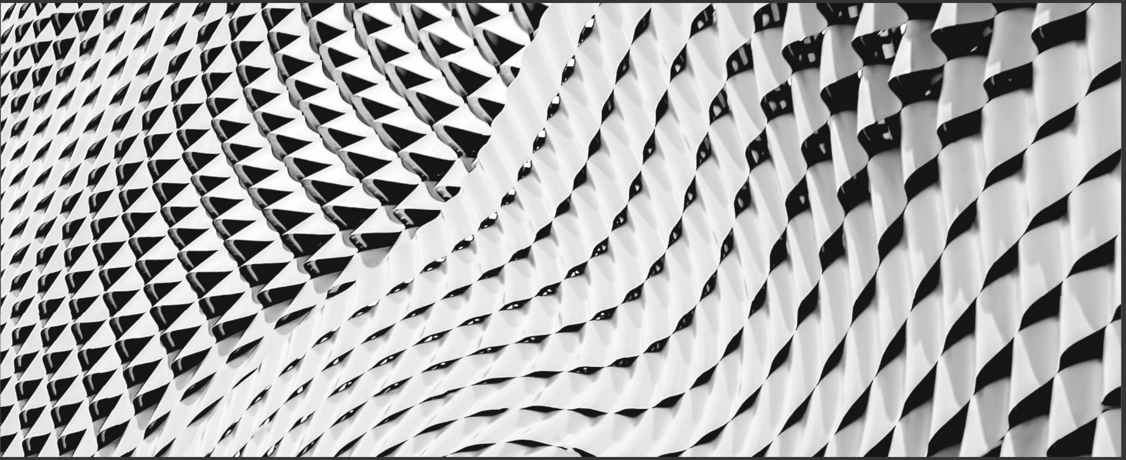
- Can run modern linux distributions (UEFI support)
- Networking support
- File sharing support for efficient `--volume` support
- But …
  - … just an API/library
  - … Objective-C / Swift

One more thing...

Rosetta

- Runs binaries built for Intel hardware on Apple Silicon machines
- Apple also provides linux binaries for Rosetta
- Can be used to run x86_64 binaries in linux virtual machines on Apple Silicon
- https://developer.apple.com/documentation/virtualization/vzlinuxrosettadirectoryshare?language=objc

vfkit



Code-Hex/vz

- https://github.com/Code-Hex/vz
- Written by Kei “Code-Hex” Kamikawa
- Go bindings for Apple’s Virtualization Framework
- MIT Licensing
- Follows closely new APIs added in macOS 12 and macOS 13

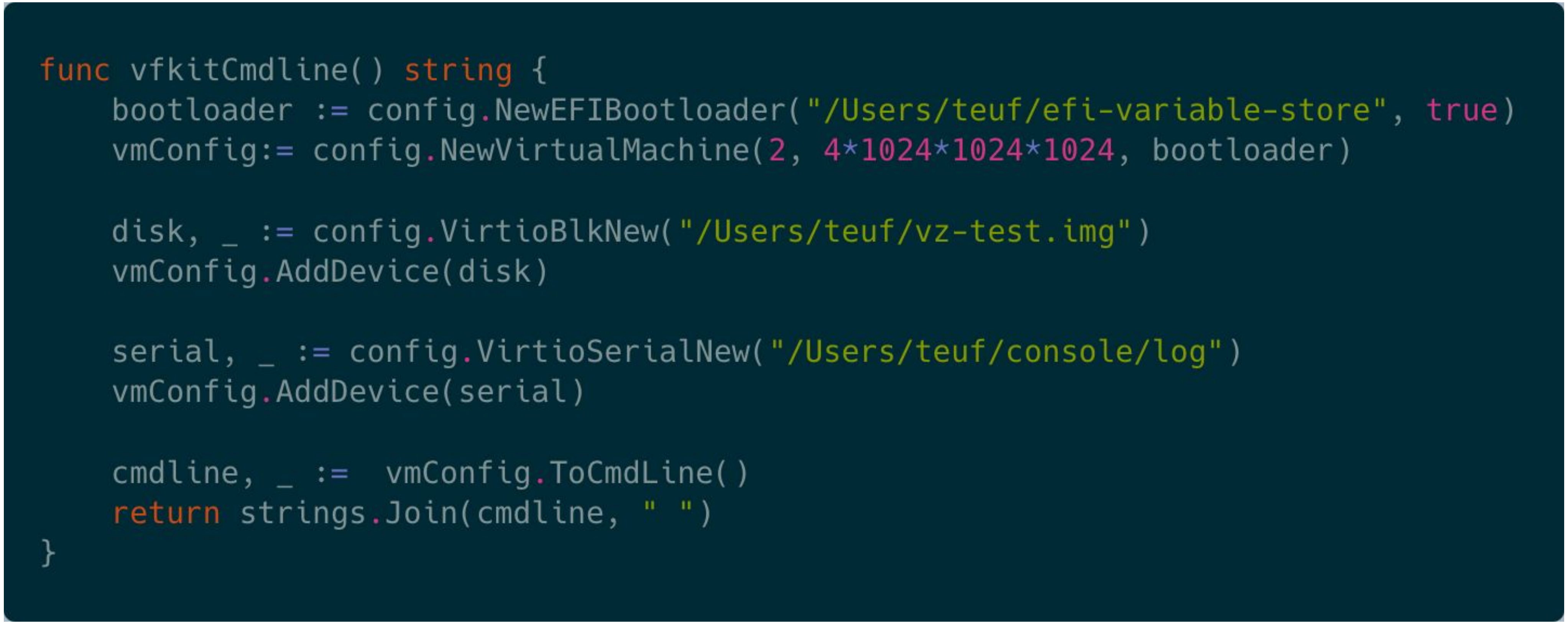
vfkit

- https://github.com/crc-org/vfkit
- Command-line tool which uses the Code-Hex/vz bindings
- Written in go
- Apache v2 Licensing

vfkit

```
./out/vfkit --cpus 2 --memory 2048 \
    --bootloader efi,variable-store=/Users/teuf/efi-variable-store,create \
    --device virtio-serial,stdio \
    --device virtio-fs,sharedDir=/Users/teuf,mountTag=dir0 \
    --device virtio-blk,path=/Users/teuf/vz-test.img \
    --device virtio-blk,path=/Users/teuf/Fedora-Server-x86_64-37-1.7.iso \
    --device virtio-net,nat,mac=72:20:43:d4:38:62
```

vfkit



```go
func vfkitCmdline() string {
    bootloader := config.NewEFIBootloader("/Users/teuf/efi-variable-store", true)
    vmConfig:= config.NewVirtualMachine(2, 4*1024*1024*1024, bootloader)

    disk, _ := config.VirtioBlkNew("/Users/teuf/vz-test.img")
    vmConfig.AddDevice(disk)

    serial, _ := config.VirtioSerialNew("/Users/teuf/console/log")
    vmConfig.AddDevice(serial)

    cmdline, _ :=  vmConfig.ToCmdLine()
    return strings.Join(cmdline, " ")
}
```

- `github.com/crc-org/vfkit/pkg/config` provides a go API to generate this command line

What's next?

- Add podman-machine driver using vfkit
  - Missing piece is ignition support
- Rosetta support
- Better integration with usermode networking stacks

Questions?

github.com/cfergeau

✉ cfergeau@redhat.com

Usermode networking

- Avoids networking issues on corporate machines with VPNs/…
- Uses github.com/containers/gvisor-tap-vsock