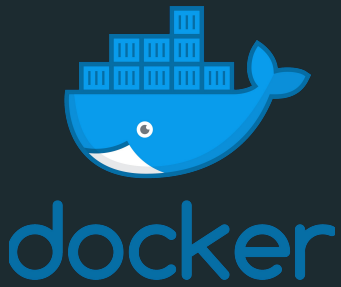
Immutability and state: making a functional OS

Justin Cormack



Who am I?

Engineer at Docker in Cambridge, UK.

Work mostly on Docker's open source projects

@justincormack

A bit of my story

Way back in the late 1990s

I started learning both Haskell and Linux sysadmin at the same time

For a long time I have done both dev and ops (DevOps is great!)

Why can't we have nice things in ops too? Why does it change so slowly?

I am both quite pragmatic about actually wanting to ship stuff and have people use it, but also think we can do much better. This talk is about working with that in industry.

The story of LinuxKit

LinuxKIT

Fast forward to 2015

Started working on Docker for Mac

Needed a simple embedded, maintainable, invisible Linux

first commit: "not required: self update: treated as immutable"

This project became LinuxKit, open sourced six months ago

Immutable delivery

LINUXKIT

The road to immutability

“As a system administrator, one of the scariest things I ever encounter is a server that’s been running for ages.

If you absolutely know a system has been created via automation and never changed since the moment of creation, most of the problems disappear.”

Chad Fowler,Trash Your Servers and Burn Your Code, 2013

Immutable delivery at Netflix

“In the cloud, we know exactly what we want a server to be, and if we want to change that we simply terminate it and launch a new server with a new AMI.”

Netflix Building with Legos, 2011

Immutable delivery always appealed to me

- So much simpler! Out with the old, in with the new
- Ops had been building layering of automated desired state management on top of systems designed for humans to manage
- Layers on layers

Design for mass production

LINUX KIT

Control state

State control for the whole system

Things we learnt from functional programming

- immutability wherever possible
- make state mutations controlled and understandable
- ad hoc state mutations from anywhere in the code are terrible!

How can we apply this to operating systems?

Diversion: Unikernels

Do you need an operating system?

The idea of unikernels is to remove the traditional OS

Why isn't my OS just a library of code I can use in my application?

- Libraries for traditional OS code eg TCP/IP
- (Use VMs not bare metal to avoid writing hardware drivers)
- Just run your application
- MirageOS (OCaml)
- HalVM (Haskell)
- IncludeOS (C++)
- Erlang on Xen (Beam)

Why isn't everyone using unikernels yet?

- Lots of library code to write and make robust for production
- Still fairly new
- Little support for stateful applications so far
- Being used successfully in areas such as network appliances

Because they run on VMs you tend to still need an OS underneath to manage multiple VMs and the physical hardware. Cloud providers don't tend to sell the right size VMs.

What do we need to run applications with Linux?

Containers for packaging applications

Containers showed that you could package up applications with less

- applications do not need a full Linux image to run
- just an application
- either statically linked or with just the needed libraries
- some certificates
- some secrets injected at runtime

For stateless applications that is all you need. For stateful applications add a storage volume for state.

Enough Linux to run containers is easy

- init binary
- configure devices
- mount pseudo filesystems in the standard locations
- set kernel config options eg sysctl
- format disk if blank
- mount persistent storage
- run a low level container manager eg containerd
- things like dhcp are optional, in containers

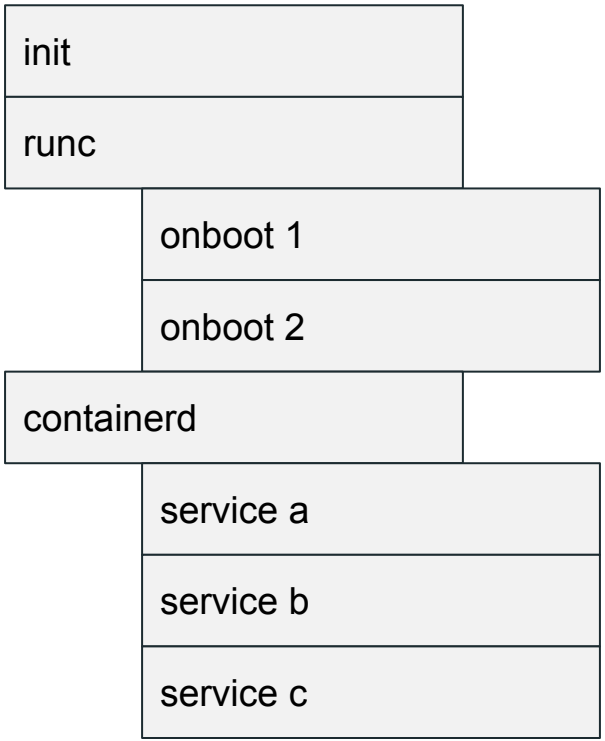
Simpler stack than `systemd`...

How does LinuxKit do this?

LinuxKit

- it is a kit, with some pieces to get you started
- everything can easily be replaced if required
- designed to be built in a CI pipeline
- build times just a few minutes
- test locally then ship to production
- minimal so boots fast too

LinuxKit startup



sequential startup
eg network configuration, disks

services start up in parallel after initialization

same design as pods in Kubernetes

LINUX KIT

Configure this from a yaml file

```yaml
kernel:
  image: linuxkit/kernel:4.9.60
  cmdline: "console=tty0 console=ttyS0 console=ttyAMA0"
init:
  - linuxkit/init:42a92119e1ca10380e0d33e26c0cbcf85b9b3558
  - linuxkit/runc:817fdc592eac6cb7804fa1721a43a7f6e23fb50f
  - linuxkit/containerd:82be2bbb7cf83bab161ffe2a64624ba1107725ff
onboot:
  - name: dhcpcd
    image: linuxkit/dhcpcd:48831507404049660b960e4055f544917d90378e
    command: ["/sbin/dhcpcd", "--nobackground", "-f", "/dhcpcd.conf", "-1"]
services:
  - name: getty
    image: linuxkit/getty:6af22c32c98536a79230eef000e9abd06b037faa
  - name: redis
    image: redis:4.0-alpine
    capabilities:
     - CAP_NET_BIND_SERVICE
     - CAP_CHOWN
     - CAP_SETUID
     - CAP_SETGID
     - CAP_DAC_OVERRIDE
```

LINUXKIT

Demo: how you should consume Linux

important differences

- root filesystem is immutable
- can run from ISO, initramfs, squashfs, ...
- no package manager
- no possibility to update at runtime
- replace with a new image to update software
- if you really want dynamic services at runtime you can use Docker or Kubernetes on top
- removes all complexity of install, update, reboot

Diversion: NixOS

Lots of people ask why not NixOS?

- Different use cases, solving different problems!
- NixOS is traditional package management done better
- LinuxKit has no package management in the conventional sense
- you can use Nix to build containers for LinuxKit
- Nix is great for building fully reproducible containers
- There is a LinuxKit based Nix container builder PR https://github.com/NixOS/nixpkgs/pull/29628
- NixOS is for better, more obedient pets
- LinuxKit is for managing herds of cattle

Practicalities

Simple tooling for lots of use cases

- Tooling can build most kinds of image needed to boot VMs or bare metal
  - ISO for EFI or BIOS
  - raw disk images
  - AWS AMIs
  - GCP disk format
  - QCOW2 for qemu and KVM
  - VHD
  - VMDK
  - raw kernel and initramfs
  - Raspberry Pi3 image

Simple tooling for lots of use cases

- Simple build, push, run workflow for many common use cases
  - AWS
  - GCP
  - Azure
  - OpenStack
  - Hyperkit for MacOS local
  - Hyper-V for Windows local
  - KVM for Linux local
  - Vcenter
  - Packet.net iPXE

Simple tooling for lots of use cases

Generally (example Google Cloud)

```
moby build file.yml
linuxkit push gcp filename
linuxkit run gcp filename
```

Some platforms have additional options

Managing state

Managing state cleanly

- root filesystem is immutable
- the containers are immutable
- you can pass in a config file as userdata
- for persistence you have to mount a disk
- most images with persistence use a single volume
- when you upgrade a machine with a new image re-attach the disk
- in cloud this is easy, as volumes can be re-attached anywhere
- for physical machines, reboot same instance
- looking at tools for snapshots and restore on different instances

Clean state

- A single state volume is easy to manage and test
- no state scattered around the filesystem
- use bind mounts to provide containers with persistent store where they expect it, usually in `/var` but sometimes in `/home` or `/etc`...

code + config + mutable state all clearly separated

Systems as functions

Patterns

- stateless machine images are simple, no arguments
- most stateful machines are best modelled as functions of a single state, a disk image
- a disk seems like a complex state, but the simpler and better structured your application is the more this makes sense
- eg the disk may just contain a single persistent data store
- snapshot the disk and you can replay your whole state

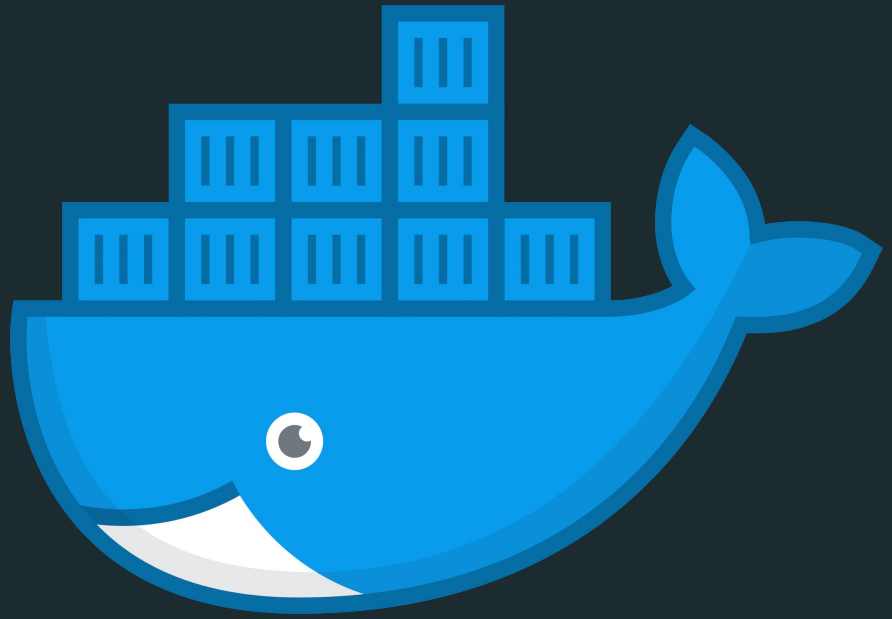
LinuxKit
make Linux fun again!

LINUXKIT

Things I wish for

- when things change, remove layers don't add them
- design for manufacturing
- with increasing automation, most computers never have a user log in
- easy trial adoption is really important to build community
- appliances are a great model for systems that "just work"
- an appliance is a function
- we have not lost the war against complexity yet
- if you have to use Linux, try consuming it differently

https://github.com/linuxkit/linuxkit



CODE MESH
LONDON 2017

THANK YOU