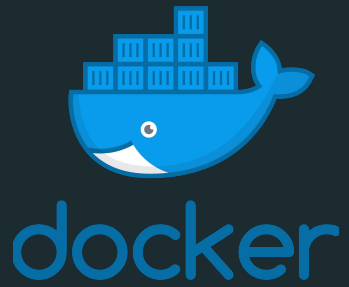
Simple custom Linux distributions with LinuxKit

Justin Cormack



Who am I?

Engineer at Docker in Cambridge, UK.

@justincormack

Tools for building custom Linux

Tools for building custom Linux

Existing tools are

- very complicated
- designed for embedded systems
- very opinionated
- hard to customise
- slow to build and test

LinuxKit

LinuxKit

- fast to build and test
- simple configuration
- totally customisable
- runs in many different environments
- not very opinionated
- uses containers for packaging
- suitable for embedded, mainframes, supercomputers

docker

Started in 2015

Originally built for Docker for Mac

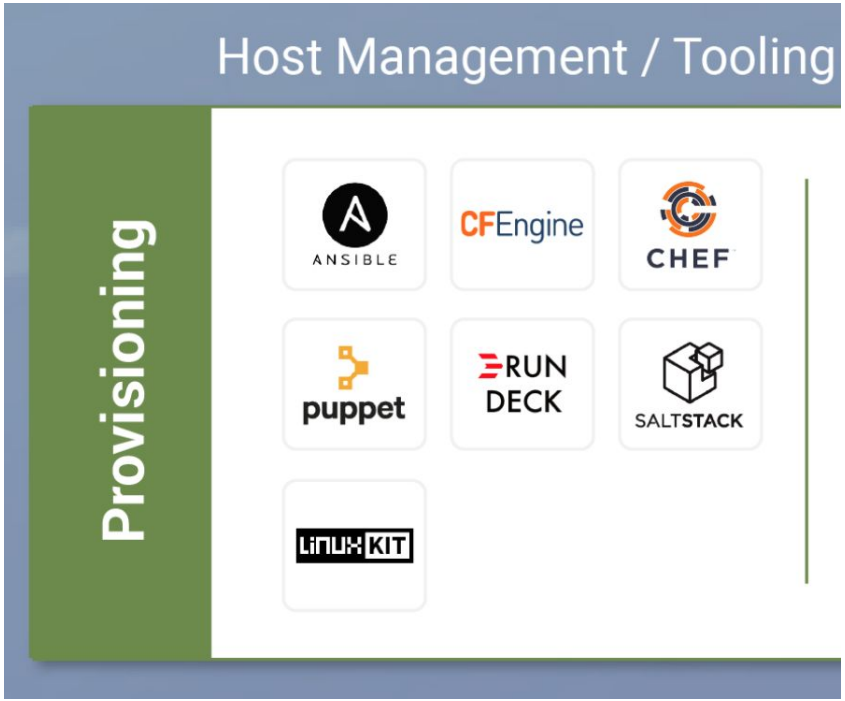
Needed a simple embedded, maintainable, invisible Linux

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

This project became LinuxKit, open sourced last year

LinuxKit is a config management tool

- defines your system configuration
- essentially just lists files
- uses containers to simplify this
- tooling to make bootable images
- built for automation



LinuxKit

- not a Linux distro!
- it is a kit, with enough pieces to get you started
- everything can easily be replaced if required
- components specified as containers
- containers run by containerd, small container runtime
- designed to be built and tested in a CI pipeline
- build times just a minute or so
- test locally then ship to production
- minimal so boots fast
- small so secure and does not need updating so much
- Apache licensed

Packages

- Packages are containers
- Containers are therefore the unit of defining the system
- A larger chunk that traditional systems
- Some less sharing
- Easier testability as it is a complete system
- Good service isolation
- VM isolation can be added
- Signing at the container level

Immutable

Sysadmins everywhere

“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

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

Updates

Reprovision to update

https://www.oreilly.com/ideas/an-introduction-to-immutable-infrastructure

State

immutability is a name

- naming is hard
- it does not mean there is no change!
- state in traditional Unix is not very well isolated
- enforce split between code (immutable) and application data (mutable)

Immutable does not mean stateless!

- functional programming is the analogy
  - no mutable global state
  - state change is explicit
  - state changes only made by specific parts of the code
  - aim is understandability
- state is managed and controlled, and in chosen locations
- LinuxKit has an immutable root filesystem, add writable drives for data
- controlled state mutation, not scattered all over
- persistent state changes are for data, not code or configuration

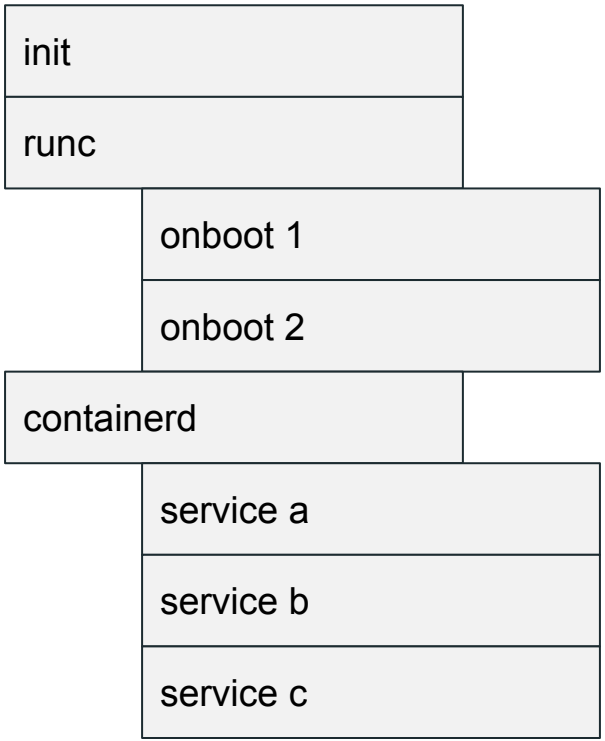
Manage your data independently

Independent data lifecycle from code

- network storage
- cloud storage
- filesystem snapshotting (Dotmesh integration)
- replication

LinuxKit architecture

LinuxKit startup



sequential startup
eg network configuration, disks

services start up in parallel after initialization

same design as pods in Kubernetes

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
```

note

- root filesystem is immutable
- can run from ISO, initramfs, squashfs, ...
- onboot stage is very flexible, can mount service containers from network or other arrangements if required.
- no package manager
- no possibility to update at runtime
- replace with a new image to update software
- for dynamic services can use Docker or Kubernetes on top
- removes all complexity of install, update, reboot

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
  - VMware Vcenter
  - Packet.net iPXE
  - Hyperkit for MacOS
  - Hyper-V for Windows
  - KVM for Linux
  - VMware Fusion
  - Virtualbox

Simple tooling for lots of use cases

Generally (example Google Cloud)

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

Some platforms have additional options. You should always use other tooling to run in production, `linuxkit run` is a development tool.

docker

Total control of how code runs

- Yaml config specifies the OCI configuration
- specify exactly users, capabilities, namespaces
- images can have their configuration in labels to simplify config
- modularity at the function and application level
- runs exactly what you need

Demo

Roadmap

- reworking build to not require Docker
  - easier to run in CI, eg in a container
  - will not require root access
  - Go code for building disk images
  - a lot of work to do here, but really useful
- more detailed application blueprints
  - a lot of work in linuxkit/kubernetes ongoing now
- remove remaining shell scripting from configuration!
  - most gone already
  - some shelling out rather than native code still
- more users and use cases!

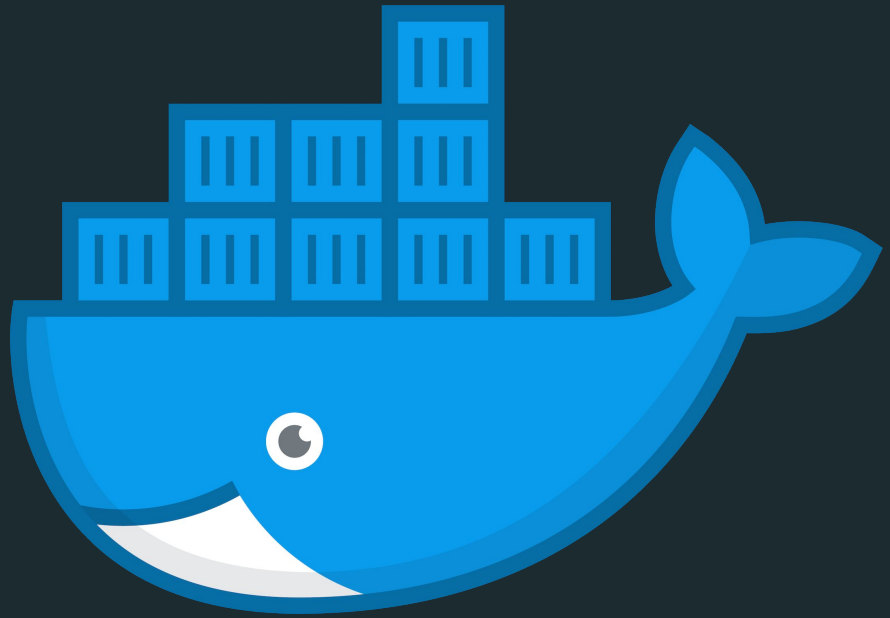
docker

Summary

Summary

- manage the configuration of a distributed system
- use a simpler, immutable OS, just focus on the overall system
- try our tools, they are simple but fun
- fast tooling is way better!
- build for automation, not for interactive use

- "time to do some crazy bullshit with operating systems" Adam Jacob



THANK YOU