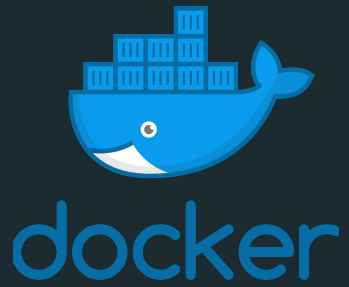
Making Immutable Infrastructure simpler with LinuxKit

Justin Cormack



Who am I?

Engineer at Docker in Cambridge, UK.

Work on security, operating systems, LinuxKit, containers

@justincormack

Some history

Config management

“The self-modifying behavior of both manual and automatic administration techniques helps explain the difficulty and expense of maintaining high availability and security in conventionally-administered infrastructures.

A concise and reliable way to describe any arbitrary state of a disk is to describe the procedure for creating that state.”

Steve Traugott, Why Order Matters: Turing Equivalence in Automated Systems Administration, 2002

docker

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

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

Application specific systems

"Use container-specific OSes instead of general-purpose ones to reduce attack surfaces. When using a container-specific OS, attack surfaces are typically much smaller than they would be with a general-purpose OS, so there are fewer opportunities to attack and compromise a container-specific OS."

NIST Application Container Security Guide, 2017

http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-190.pdf

Updates

Getting updates right is hard

- make sure update is tested first from exact current state
- update environment variable: restart process
- update config file: restart process, probably, unless supports reload
- update DNS entry: wait until cache expiry
- update shared library: restart all processes using it
- update container engine config: restart all containers
- update kernel: reboot machine

- almost all updates imply service downtime
- updates involve domain specific knowledge and complexity

Reprovision to update

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

The system remains in service

- individual components go out of service
- not "immutable", but "disposable" at the machine level
- the system of interest is the system as a whole, not the machine

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
- trying to 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

Immutability is already in use

- hardly anyone updates containers
- somehow we persuaded people not to do this
- it is easier with a new set of tooling, as you set principles

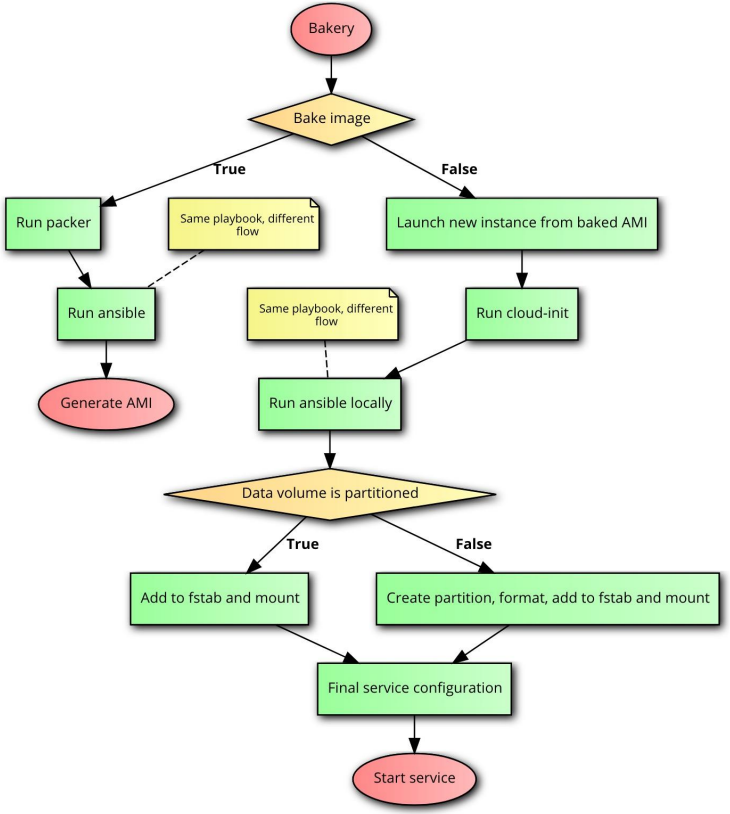
Change at a higher level

- infrastructure has grown from programs and computers to clusters
- need to understand at larger scales
- need to reduce complexity at the lower levels
- config management of distributed systems is the interesting problem
- you have a lot of really hard distribution problems to deal with
  - scaling
  - consistency
  - replication
  - failure
- don't complicate this by also having lots of mutability

Why no immutable infrastructure products?

Existing workflows

- most people use Packer
- plus other CM tools
- workflow is complex
- involves booting machine



https://blog.kintoandar.com/2017/06/Baking-delicious-cloud-instances.html

docker

Hard to package as an enterprise product

- charging model does not match
- no tooling running on production machines
- enterprises are very conservative

So we are going to have to build this as open source tooling

LinuxKit

Started in 2015

Originally built for Docker for Mac

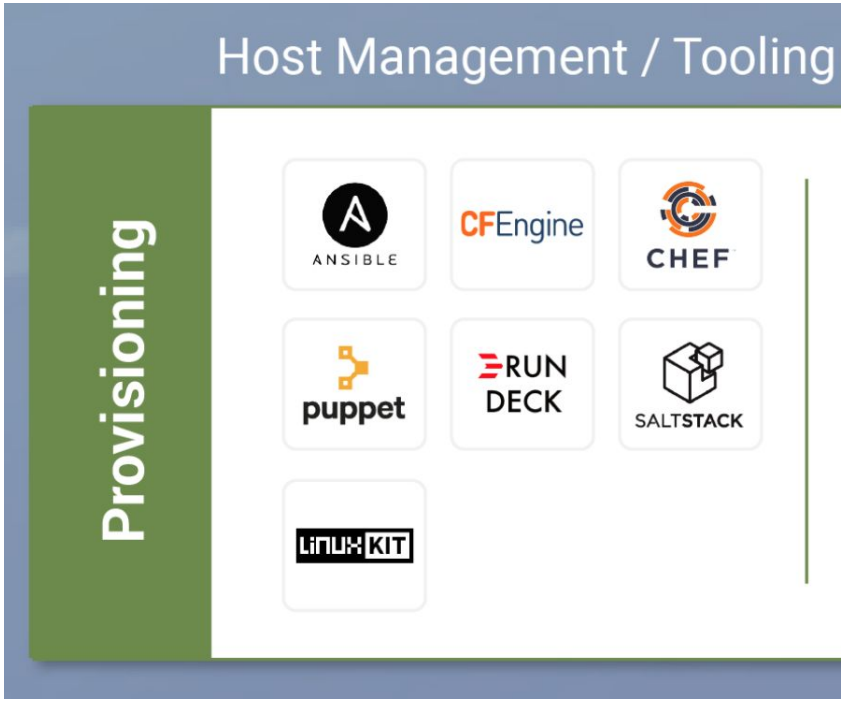
Needed a simple embedded, maintainable, invisible Linux

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

This project became LinuxKit, open sourced least year

LinuxKit is a config management tool

- defines your system configuration
- essentially just lists files
- uses containers to simplify this
- includes tools to make bootable etc
- built for automation



docker

LinuxKit

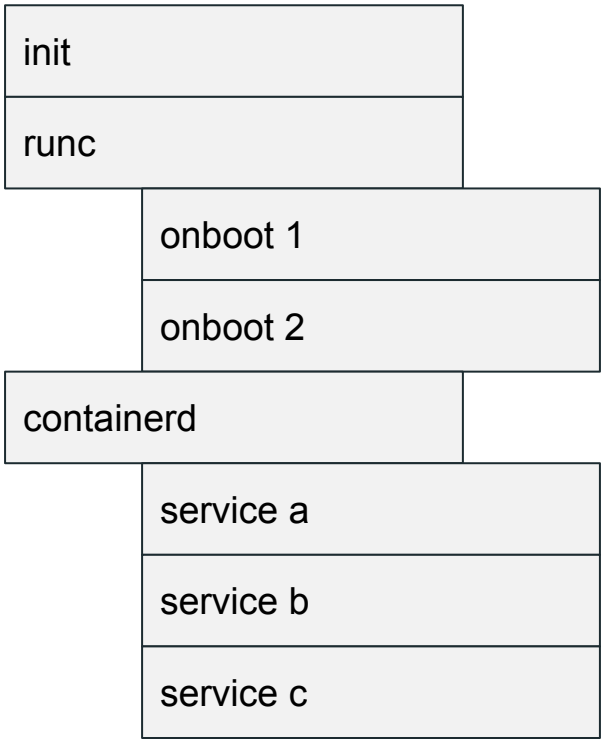
- not a Linux distro!
- it is a kit, with enough pieces to get you started
- everything can easily be replaced if required
- components specified as containers
- containers run by containerd, small container runtime
- designed to be built and tested in a CI pipeline
- build times just a minute or so
- 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 will be added later
- Signing at the container level

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, ...
- 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

docker

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

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

Why is no one using immutable infrastructure?

Lack of tooling

- other than LinuxKit and Packer, not many options
- some intermediate options like CoreOS
  - not entirely immutable, but close
  - very specialised and hard to customise

The stickiness of the traditional distro

- familiarity breeds contentment
- Linux distros seem so complex
- enterprises need Enterprise Linux?

Issues at scale

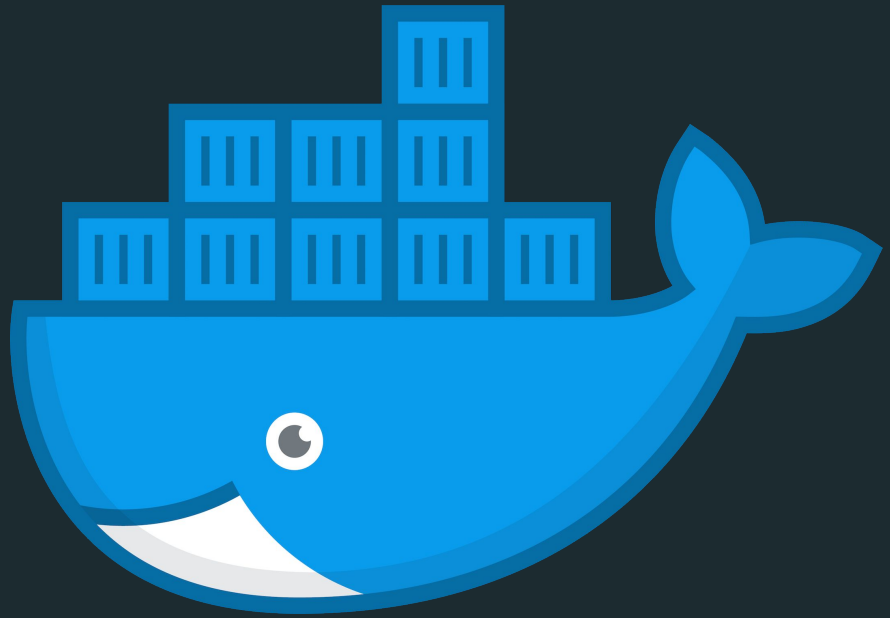
- cloud providers cannot reliably provide new instances at scale of thousands of nodes
- not designed for this rate of churn, disappointingly inelastic
- seems to be the reason Netflix no longer use immutable server model
- there are workarounds with update and kexec/reboot
  - LinuxKit does not have support for these yet
  - not very hard to add, can reuse CoreOS code

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