Containers and the pianola

Jonathan Boulle

@baronboulle

Containers and the pianola

... a tale of

- shipping containers
- infinite software
- cheesecake
- IKEA furniture
- VW factories (if we have time)
- mechanical pianos (aka the pianola)

Containers and CoreOS

- CoreOS mission: **"Secure the Internet"**
  - Make updates *seamless* and *automatic*
  - We started with the OS (Container Linux née CoreOS Linux)

Containers and CoreOS

- CoreOS mission: **"Secure the Internet"**
  - Make updates *seamless* and *automatic*
  - We started with the OS (Container Linux née CoreOS Linux)

- Application containers are key
  - Decouple application and OS update lifecycles (update at different cadences)
  - Application containers are **the** packaging system

Containers and OCI

- OCI - the Open Container Initiative
  - *Image specification* - standardise what's in a container
  - *Runtime specification* - standardise how a container runs

- What even is an application container?
  - Answer: not very interesting
  - A tarball and a bunch of JSON metadata

Containers and metaphors

Containers and (clichéd) metaphors

Shipping containers

- ✓ Agreed-on format (size and shape)
- ✓ Works with cranes, ships, trucks, trains, ...
- ✓ Transports can ignore what's inside
- ✓ Consistent experience

Application containers

- ✓ Agreed-on format
- ✓ Works with registries, build tools, runtimes...
- ✓ Transports can ignore what's inside
- ✓ Consistent experience

Shipping containers

- ✕ Operators can ignore what's inside
  - Because it's opaque and unstructured

What's inside containers?

- Shipping containers
  - Yoghurt?
  - Furniture?
  - Smaller containers?
  - Don't know how to start dealing with the contents

What's inside containers?

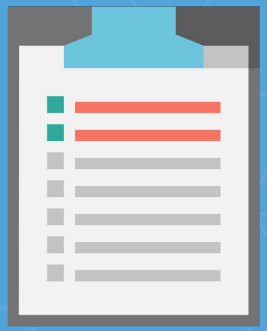
- Shipping containers
  - Yoghurt?
  - Furniture?
  - Smaller containers?
  - Don't know how to start dealing with the contents
- Application containers
  - A structured filesystem layout
  - An entrypoint: start with `/bin/httpd`

Shipping containers

- ✕ Operators can ignore what's inside
  - Because it's opaque and unstructured

Shipping containers

- ✓ Operators can ignore what's inside
  - Add a shipping manifest!
  - What's inside, how to process it



Shipping containers

- ✕ Monolithic size
  - Each container holds the same amount
  - Application containers can vary wildly

Shipping containers

- ✕ Monolithic size
  - Each container holds the same amount
  - Application containers can vary wildly
- ✕ Physically cumbersome
  - Difficult to build, difficult to move
  - Application containers can be copied in an instant

Shipping containers

- ✕ Monolithic size
  - Each container holds the same amount
  - Application containers can vary wildly
- ✕ Physically cumbersome
  - Difficult to build, difficult to move
  - Application containers can be copied in an instant
- ✕ Filled, emptied, re-used
  - Application containers are immutable, copied

Physical metaphors are hard

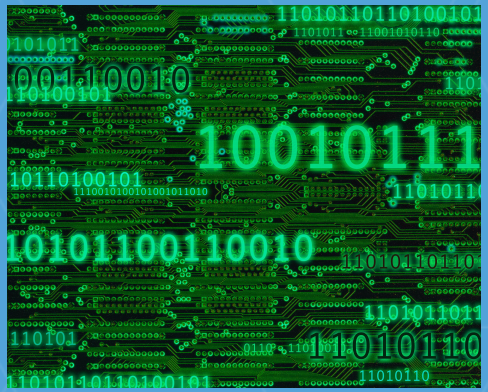
Software is

- instantly, immediately, cheaply copied
- instantly, immediately, cheaply transported
- only constrained by supply of electricity

Physical metaphors are hard

Software is

- a stream of bits and zeroes
  - true, but not very helpful



Physical metaphors are hard

Software is

- ~~a stream of bits and zeroes~~
  - ~~true, but not very helpful~~
- a sequence of instructions, potentially endless
  - CPU dumbly follows these instructions (but really fast)
  - recreate the sequence, recreate the software
  - copy the sequence, copy the software

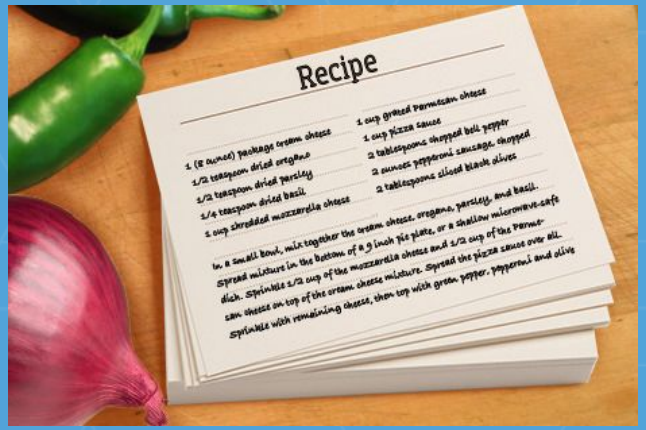
Physical metaphors - trying again

Sequence of instructions?

What about a recipe?

Recipes

- ✓ Sequence of instructions
- ✓ Easy to copy and redistribute
- ✓ Follow the instructions, get the same result



Recipes

- ✓ Sequence of instructions
- ✓ Easy to copy and redistribute
- ✓ Follow the instructions, get the same result

Application containers

- ✓ Sequence of instructions
- ✓ Easy to copy and redistribute
- ✓ Follow the instructions, get the same result

Application containers

- ✓ Sequence of instructions
- ✓ Easy to copy and redistribute
- ✓ Follow the instructions, get the same result

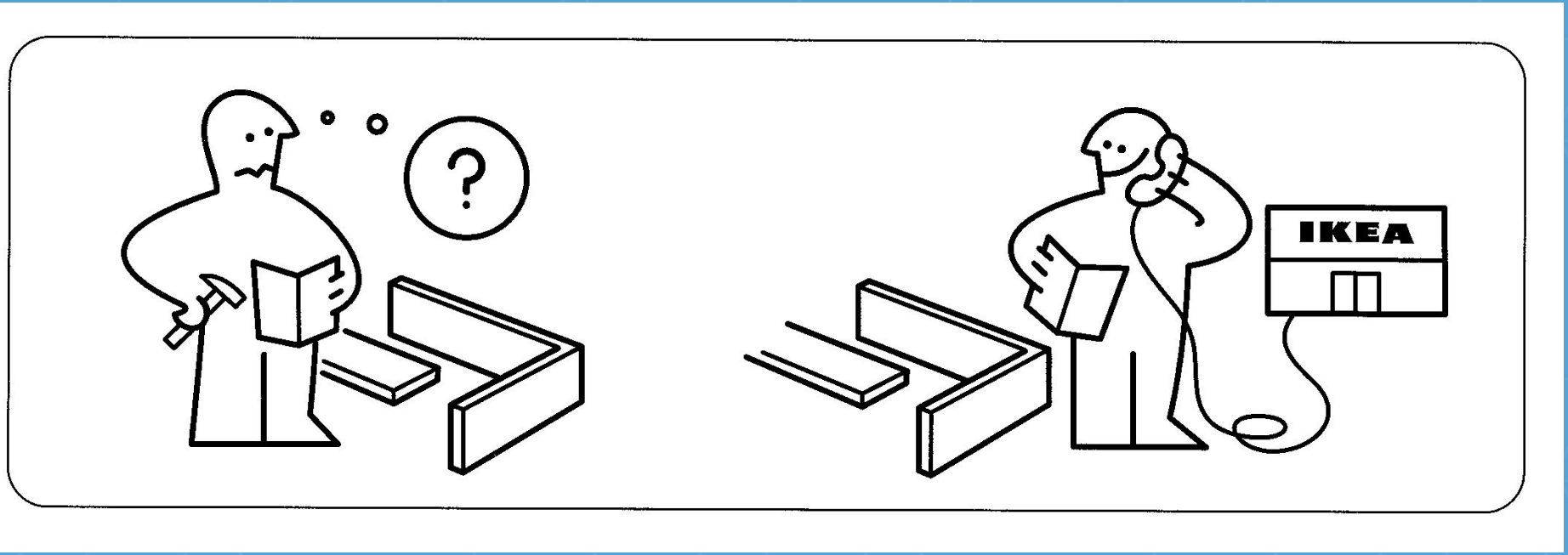
Recipes

- ✕ Needs an additional set of inputs (ingredients)
  - Inconsistency in results
  - Not self-contained like an application container

Recipes

- ✕ Needs an additional set of inputs (ingredients)
  - Inconsistency in results
  - Not self-contained like an application container

IKEA furniture



IKEA furniture

- ✓ Sequence of instructions
- ✓ Easy to copy and redistribute
- ✓ Follow the instructions, get the same result
- ✓ Ingredients (materials) included in the package
  - Self-contained, consistent result!

IKEA furniture

Physical metaphors - improving!

Physical metaphors are hard

Software is

- instantly, immediately, cheaply copied
- instantly, immediately, cheaply transported
- only constrained by supply of electricity

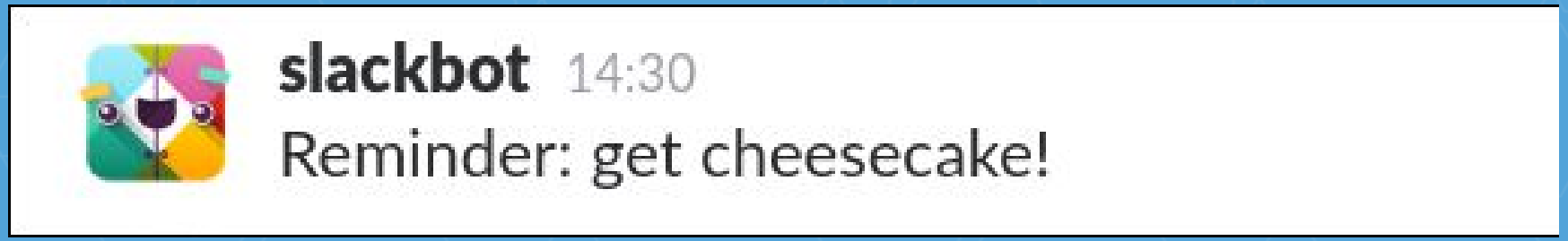
Physical metaphors are hard

Software is

- instantly, immediately, cheaply copied
- instantly, immediately, cheaply transported
- only constrained by supply of electricity
- **As long as you have electricity, software is *long-running* and *dynamic* (alive)**

Things that are not alive

- ✕ Shipping containers are (relatively) static
- ✕ So are IKEA bookshelves
- ✕ Cheesecakes get eaten
  - If not, let me know


slackbot 14:30
Reminder: get cheesecake!

Physical metaphors are hard

- ✗ Shipping containers are (relatively) static
- ✗ So are IKEA bookshelves
- ✗ Cheesecakes get eaten

✓ Software goes on,
and on,
and on

Physical metaphors - one more try


STECK

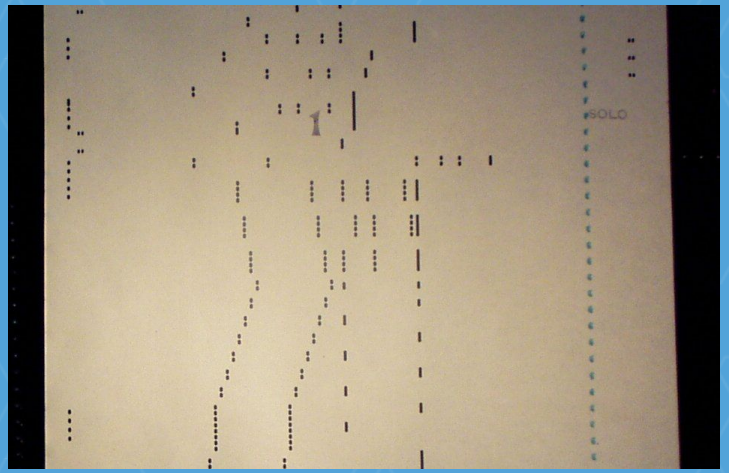
Containers and the pianola

- **Piano roll** as **application containers**
  - Set of instructions to follow
- **Pianola** as **computer** (specifically, CPU)
  - Dumbly follows instructions, only input is energy
- **Musical performance** as **software execution**
  - Dynamic, alive, long-lived

The piano roll

Packaging format

- Machine-understandable sheet music
- Various lengths, sizes
- Easy to copy, transport
- Can reference each other



The pianola

- Mechanical piano
- Dumbly consumes piano roll
- Different energy sources (e.g. pneumatic)

Software is alive

- Long running, dynamic, self-referential
- Limitless (with enough electricity)

Software is alive

- Long running, dynamic, self-referential
- Limitless (with enough electricity)

So is musical performance!

- Repeat a code block, repeat a bar of music
- Limitless (with enough mechanical energy)

How far does this go?

Sheet music and container layers

- Software containers have *layers*
  - One application container derives from another
  - No need to copy all of the files you need; simply reference the parent layer
- Piano rolls can, too!
  - Reference a bar or piece from another roll
  - Share common melodies, etc

Sheet music and container standards

- Standardise entrypoint
  - How can the pianola tell where in the sheet music it should start playing?
    - `"entrypoint": "19,23"`
  - How can the container runtime tell which binary it should start executing?
    - `"entrypoint": "/bin/httpd"`

Sheet music and container standards

- Standardise constraints
  - How loud can this piece be played?
    - `"maxVolume": "123dB"`
    - Exceed the limit? Music stops
  - How much memory can this container use?
    - `"maxMemory": "123MB"`
    - Exceed the limit? Software stops

Sheet music and container standards

- Standardise discovery
  - How can I find this referenced piece by Bono?
    - Look up Bono in the telephone book
    - Call the phone number
    - Ask for his cool piece
  - How can I find this referenced container image layer?
    - Look up `bono.com` in DNS
    - Connect to port 80
    - `GET /songs/cool_piece JPTP/1.1`

What else?

Multiple clients / listeners
Remote access / listening
Container orchestration / pianola orchestras
So much more...

Thanks!

QUESTIONS?

jonathan.boulle@coreos.com

@baronboulle

linkedin.com/jonboulle

LONGER CHAT?

Let's talk!

IRC: #coreos

More events: coreos.com/community

We're hiring: coreos.com/careers
careers-berlin@coreos.com

CoreOS is running the world's ~~pianolas~~ containers

We're hiring: careers-berlin@coreos.com

OPEN SOURCE

90+ Projects on GitHub, 1,000+ Contributors

CoreOS

rkt etcd

coreos.com

ENTERPRISE

Support plans, training and more

TECTONIC

QUAY

sales@coreos.com


CoreOS

Containers and car factories

VW factories

- ✓ Multiple instances (copies) in different locations
  - Same template, same behaviour
- ✓ "Black boxes" of processing
  - Take input (orders), produce output (cars)
- ✓ Redundant, highly available
  - Load-balance orders across factories

Containers and car factories

Containers as microservices

- ✓ Multiple instances (copies) in different locations
  - Same template, same behaviour
- ✓ "Black boxes" of processing
  - Take input (requests), produce output (responses)
- ✓ Redundant, highly available
  - Load-balance traffic across instances

Containers and me

- **appapp** - the application application (RIP)
- **appc** - App Container Specification
- **OCI** - Open Container Initiative
- **CNCF** - Cloud Native Computing Foundation


CoreOS