The Open Container Initiative

Establishing standards for an open ecosystem

Jonathan Boulle
@baronboulle | jon@coreos.com

~~The Open Container Initiative~~

~~Establishing standards for an open ecosystem~~

Containers, standards, and the pianola

In search of a better metaphor

Jonathan Boulle

@baronboulle | jon@coreos.com

~~The Open Container Initiative~~

~~Establishing standards for an open ecosystem~~

Containers, standards, and the pianola

In search of a better metaphor

(...with some stuff about OCI too)

Jonathan Boulle

@baronboulle | jon@coreos.com

Containers, standards, and the pianola

... a tale of

- shipping containers
- infinite software
- cheesecake
- IKEA furniture
- mechanical pianos (aka the pianola)
- container standards

But first, a bit about OCI

OPEN CONTAINER INITIATIVE

AN OPEN GOVERNANCE STRUCTURE FOR THE EXPRESS PURPOSE OF CREATING OPEN INDUSTRY STANDARDS AROUND CONTAINER FORMATS AND RUNTIME

Containers and OCI

- OCI - the Open Container Initiative
  - Founded mid-2015 to end the "Container Wars"

Containers and OCI

- OCI - the Open Container Initiative
  - Founded mid-2015 to end the "Container Wars"
  - But really... to formalise the *de facto* Docker standard
  - Agreed-on starting point for future innovation

Containers and OCI

- OCI - the Open Container Initiative
  - Founded mid-2015 to end the "Container Wars"
  - But really... to formalise the *de facto* Docker standard
  - Agreed-on starting point for future innovation

- Two key projects
  - *Image specification* - what's in a container
  - *Runtime specification* - how a container runs

Containers and OCI

- What even is an application container?
  - cgroups? Namespaces? Union filesystems?

Containers and OCI

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

Containers and metaphors

CoreOS

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

CoreOS

What's inside containers?

- Shipping containers
  - Yoghurt?
  - Furniture?
  - Smaller containers? (containers all the way down)
  - Don't know how to start dealing with the contents

What's inside containers?

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

CoreOS

Shipping containers

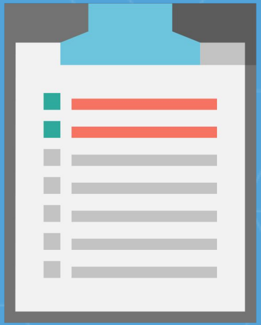
- ✕ Operators can ignore what's inside
    - ○ Because it's opaque and unstructured

CoreOS

Shipping containers

✓ Operators can ignore what's inside

- Add a shipping manifest!
- What's inside, how to process it



CoreOS

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

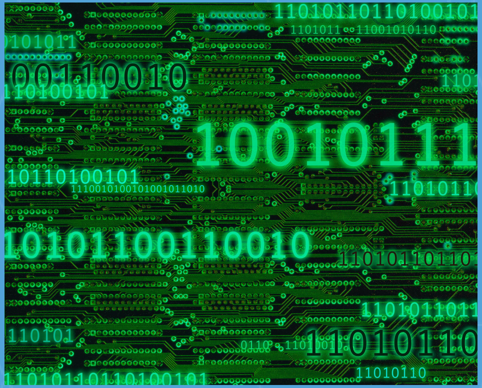
Software is

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

Physical metaphors are hard

Software is

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



CoreOS

Physical metaphors are hard

Software is

- ~~a stream of bits: ones 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

CoreOS

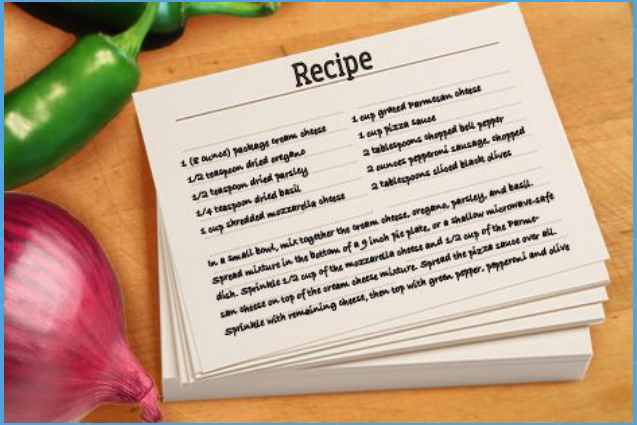
Physical metaphors - trying again

Sequence of instructions?

What about a recipe?

Recipes

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



Recipe

1 (8 ounce) package cream cheese
1/2 teaspoon dried oregano
1/2 teaspoon dried parsley
1/4 teaspoon dried basil
1 cup shredded mozzarella cheese

1 cup grated Parmesan cheese
1 cup pizza sauce
2 tablespoons chopped bell pepper
2 ounces pepperoni sausage, chopped
2 tablespoons sliced black olives

In a small bowl, mix together the cream cheese, oregano, parsley, and basil. Spread mixture in the bottom of a 9 inch pie plate, or a shallow microwave-safe dish. Sprinkle 1/2 cup of the mozzarella cheese and 1/2 cup of the Parme-san cheese on top of the cream cheese mixture. Spread the pizza sauce over all. Sprinkle with remaining cheese, then top with green pepper, pepperoni and olive

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

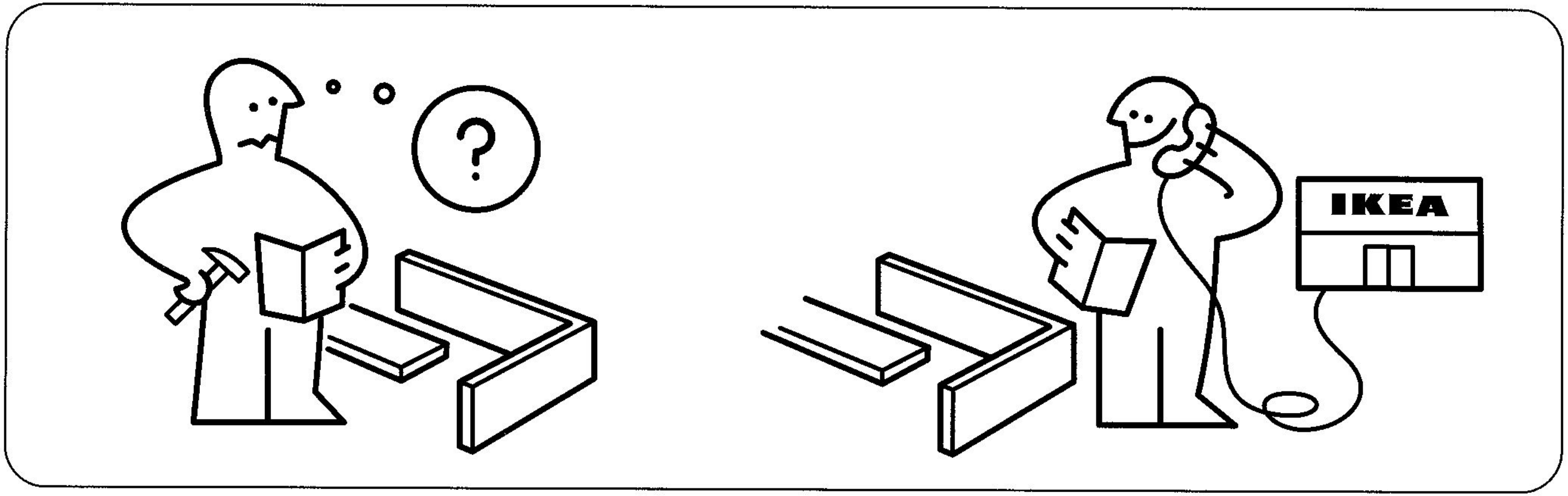
Recipes

- ✕ 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

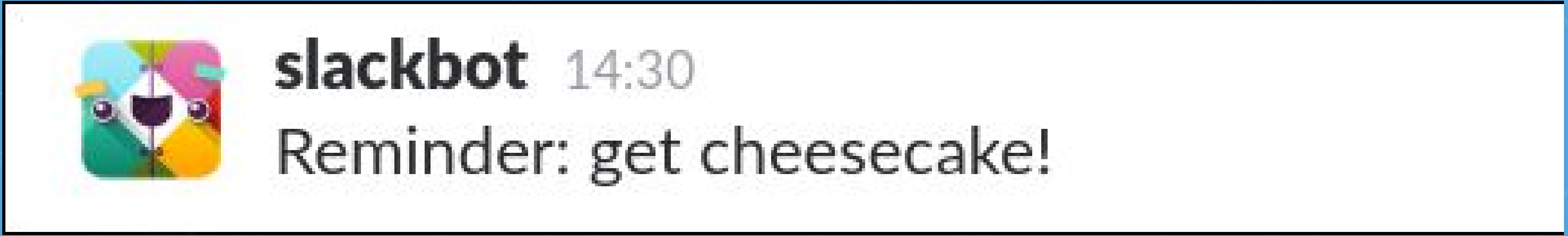
Software is

- 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)**

CoreOS

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
PIANOLA PIANO

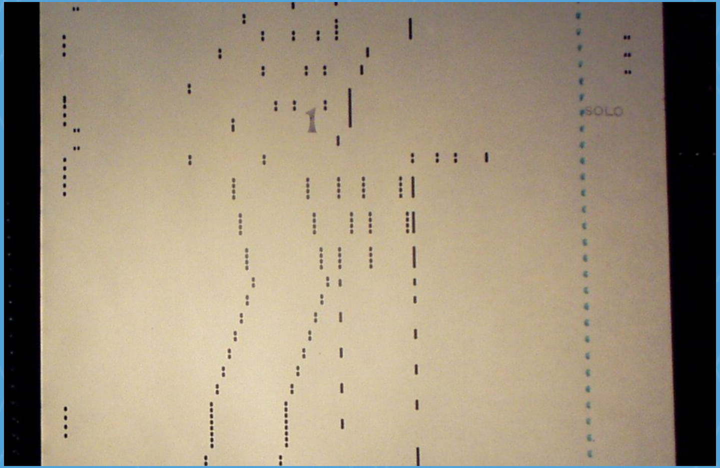
Containers and the pianola

- **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

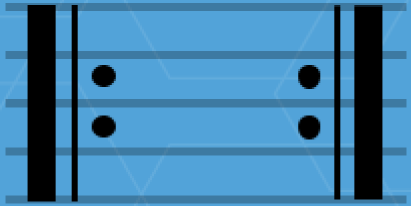
- Mechanical piano
- Dumbly consumes piano roll
- Different energy sources (e.g. pneumatic)

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

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*

Sheet music and container standards

- Standardise *entrypoint*
  - How can the pianola tell where in the sheet music it should start playing?
    - `"entrypoint": "19,2"`

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*

Sheet music and container standards

- Standardise *constraints*
  - How loud can this piece be played?
    - `"maxVolume": "123dB"`
    - Exceed the limit? Music stops

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

CoreOS

Sheet music and container standards

- Standardise *discovery*

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

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...

Time check-in

And maybe some more on OCI

OCI Today

Two separate but connected specifications

- **image-spec**: what's in a container
- **runtime-spec**: how to run a container

OCI Image Format Spec Project

- A serialized, distributable image format
  - Content-addressable
  - Platform-agnostic
- Optional extras:
  - Signatures based on image content address
  - Federated, delegatable naming based on DNS

OCI Image Format Spec Project

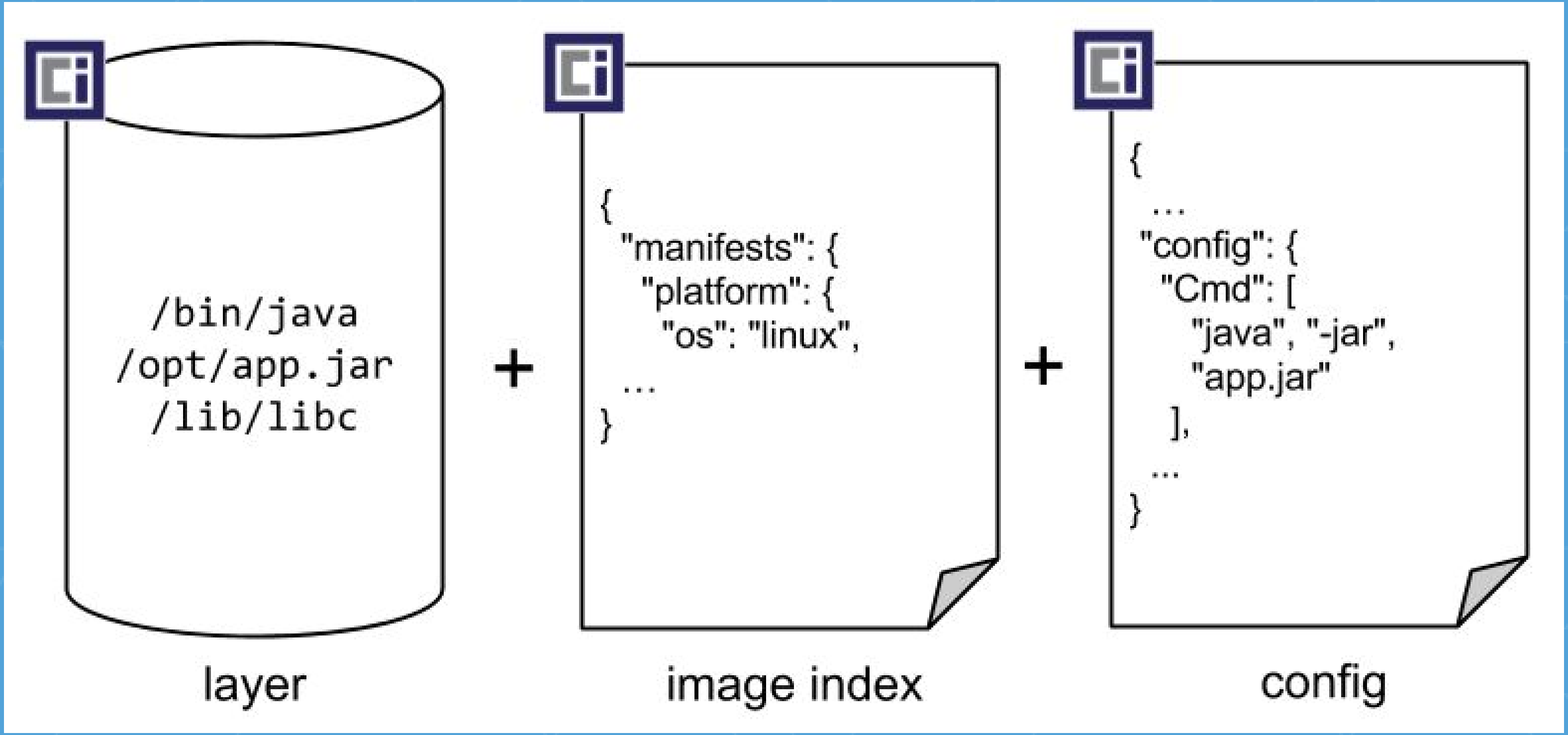
- Portable archive format
  - "The thing to distribute"
  - Structured tarball
- Image Manifest, Image Index, and Config
  - Metadata about the container image
  - References to *layers*, containing root filesystem
- Cryptographic addressability
  - How to securely reference images and parts of images

CoreOS

OCI Image Format Spec Project

- Backwards-compatible with Docker:
  - Taking the *de facto* standard Docker v2.2 format and writing it down for everyone to use
- Shared starting point for future innovation in container image format and distribution
- Intended to interoperate with Runtime Spec (similar to how appc defined both sections)

Anatomy of an OCI Image



```
{
  "manifests": {
    "platform": {
      "os": "linux",
  ...
}
```

```
{
  ...
  "config": {
    "Cmd": [
      "java", "-jar",
      "app.jar"
    ],
  ...
}
```

Inside the tarball

```
$ find busybox/
busybox/
busybox/refs
busybox/refs/latest
busybox/oci-layout
busybox/blobs
busybox/blobs/sha256
busybox/blobs/sha256/d09bddf0432...
busybox/blobs/sha256/56bec22e355...
busybox/blobs/sha256/e02e811dd08...
```

```
$ cat busybox/blobs/sha256/d09bddf043...
{
  "layers" : [
      {  "digest" : "sha256:56bec22e355981d...",
         "size" : 668151,
         "mediaType" : application/vnd.oci.image.layer.v1.tar+gzip"
      } ],
   "mediaType" : "application/vnd.oci.image.manifest.v1+json",
   "schemaVersion" : 2,
   "config" : {
      "digest" : "sha256:e02e811dd08fd49e7f6...",
      "mediaType" : "application/vnd.oci.image.config.v1+json",
      "size" : 1464
 }
```

OCI Runtime Spec

- On-disk layout of a container
  - Extracted root filesystem and configuration, ready to run
- Lifecycle verbs
  - `create, start, kill, delete, state`
- Multi-platform support
  - Shared general configuration
  - Windows/Solaris/Linux-specific bits

CoreOS

OCI Runtime Spec

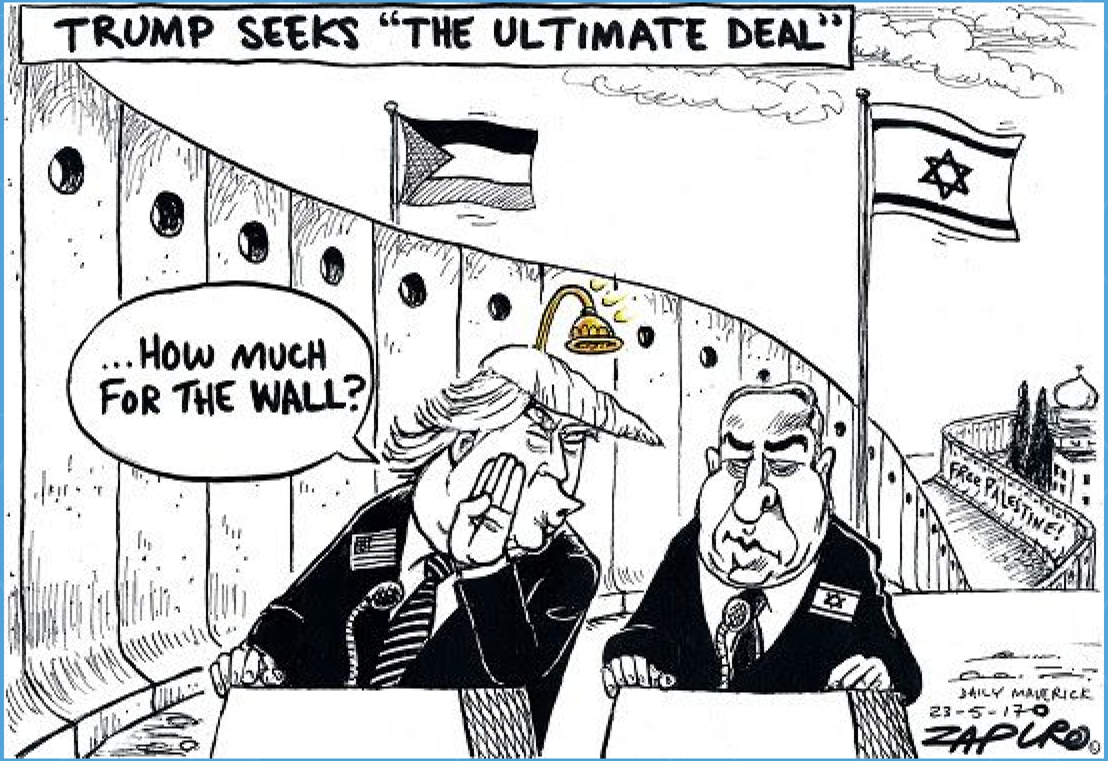
Example: container state

```
{
    "ociVersion": "v1.0.0-rc5",
    "id": "oci-container1",
    "status": "running",
    "pid": 4422,
    "bundlePath": "/containers/redis",
    "annotations": {
        "myKey": "myValue"
    }
}
```

Thank you!

All OCI work happens in the open - join us!

- GitHub:
  - https://github.com/opencontainers/image-spec
  - https://github.com/opencontainers/runtime-spec
- Email:
  - dev@opencontainers.org


TRUMP SEEKS "THE ULTIMATE DEAL"
...HOW MUCH FOR THE WALL?
FREE PALESTINE!
DAILY MAVERICK
23-5-17
ZAPIRO