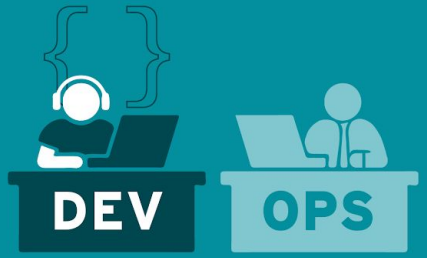


Build and run applications in a Dockerless Kubernetes world

Jorge Morales
OpenShift Developer Advocate
Riga Dev Days 2018

redhat.

Me (aka Jorge Morales)

- Spanish by nature and by language
- Work at Red Hat
- OpenShift Developer Advocate
- Mostly Java developer
- Obsessed with improving the developer experience

http://jorgemoral.es

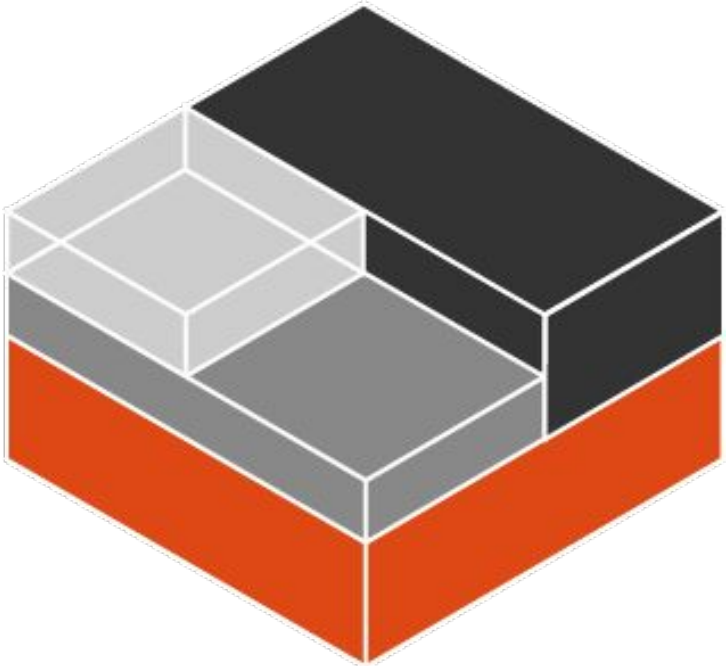
@jorgemoralespou

github.com/jorgemoralespou


Each day is a little
bit of history.
José Saramago
quotefancy


ONCE
UPON A TIME

Linux Containers

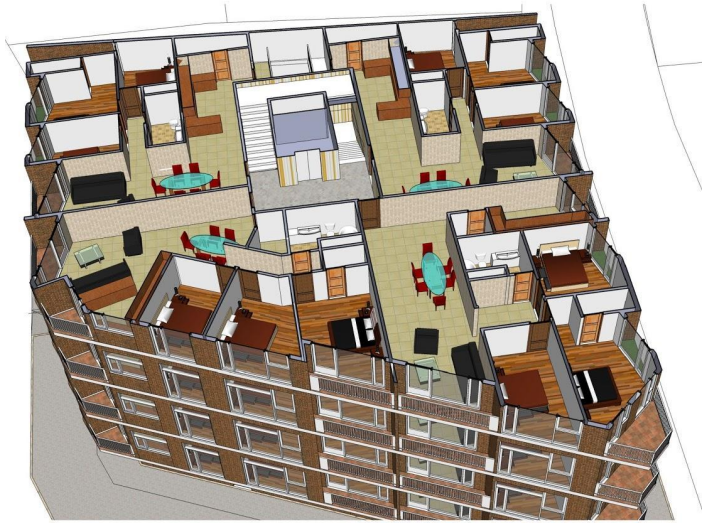


Kernel namespaces: sandboxing processes from one another

Control Groups (cgroups): control process resource allocations

Security: capabilities drop (seccomp), Mandatory access control (SELinux, Apparmor)

VMS vs Containers



VMs? Containers?
All I want to do is program!
Jeesh.


Making things easy is hard.
Ted Nelson
quotefancy

Because I'm Happy!

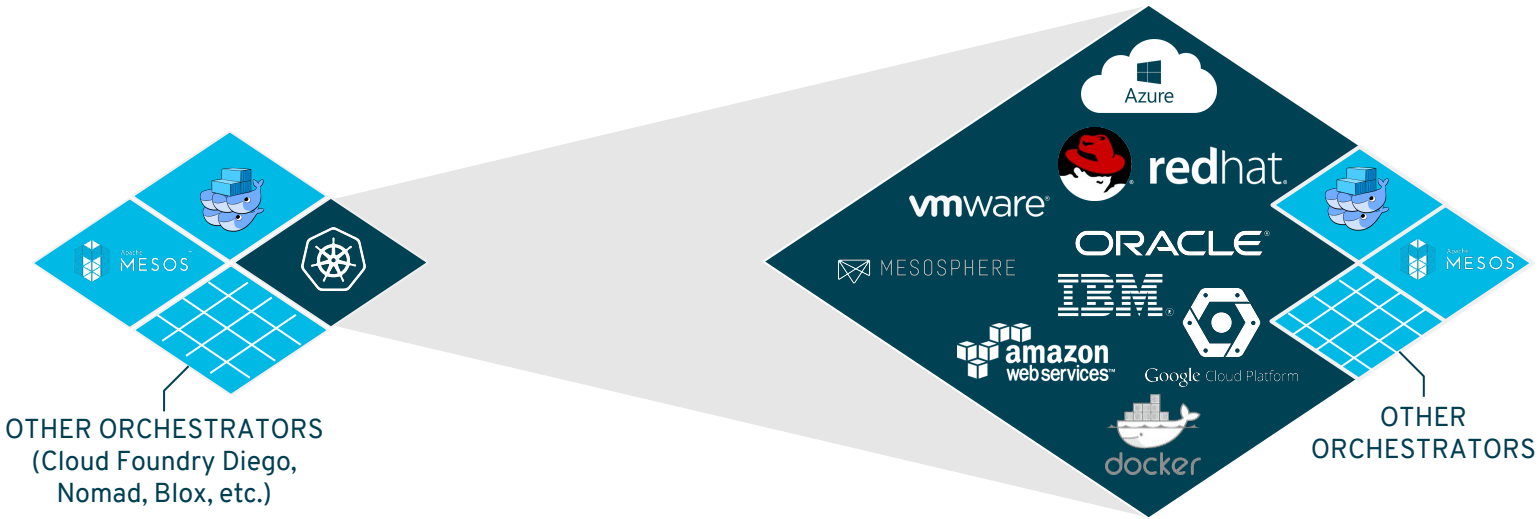

With scale came complexity

Orchestration

CONTAINER ORCHESTRATION LANDSCAPE

2 YEARS AGO
Fragmented landscape

TODAY
Kubernetes consolidation



Why kubernetes?

#1: Open source, backed by giants

#2: Vibrant and fast growing community

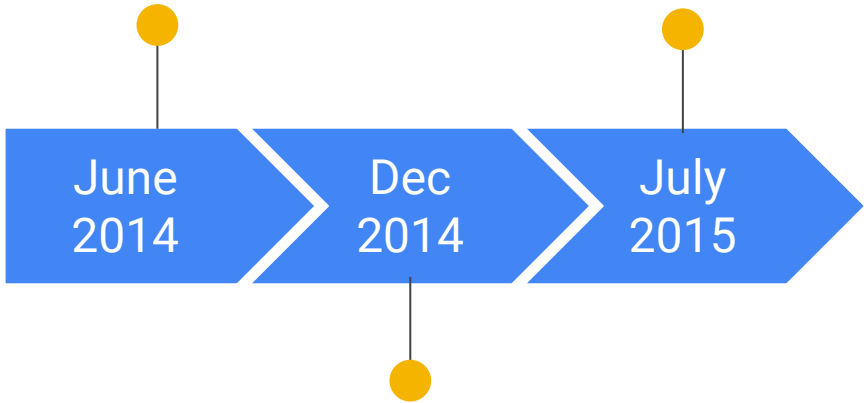
#3: Supported on all clouds

#4: Great partnerships

Started slow

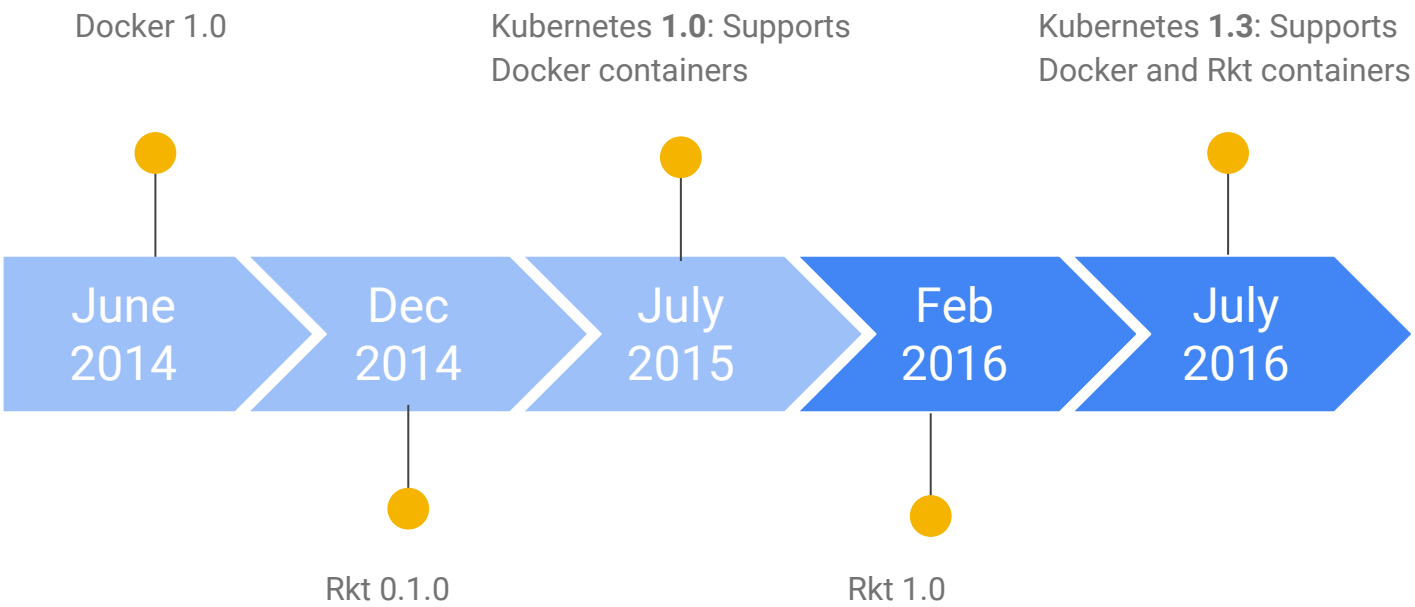
Docker 1.0

Kubernetes **1.0**: Supports Docker containers

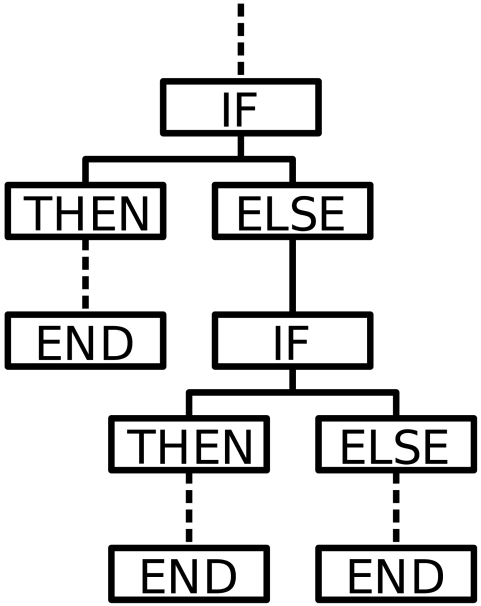


Rkt 0.1.0

then more runtimes showed up



and code got messy



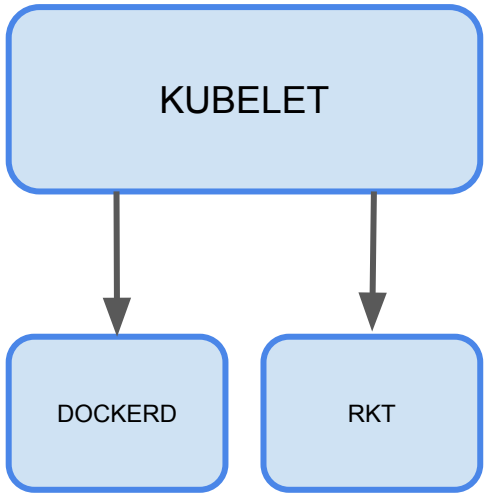
“Change is the essential process of all of existence.”

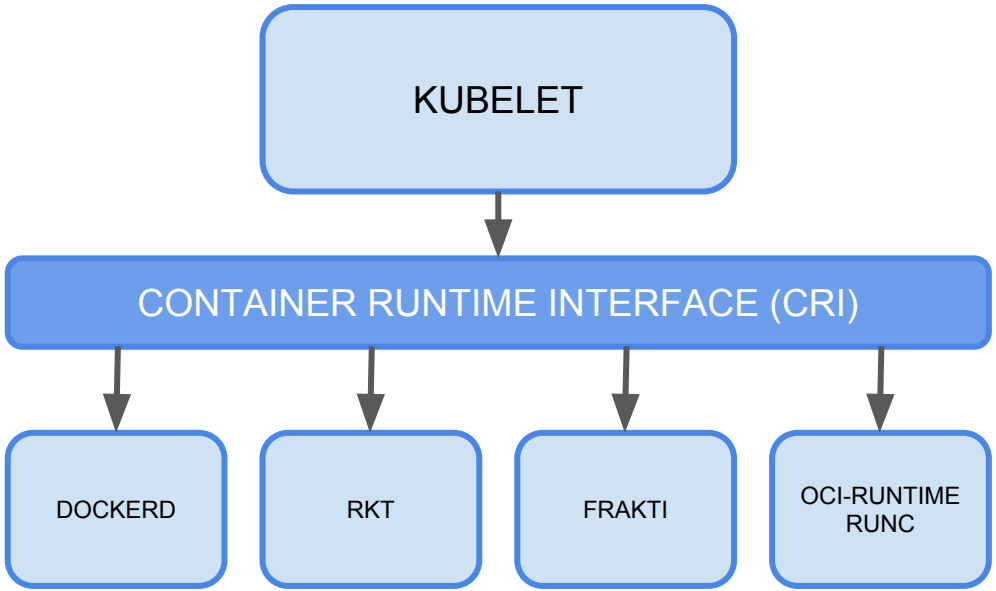
—Spock

Standardize containers

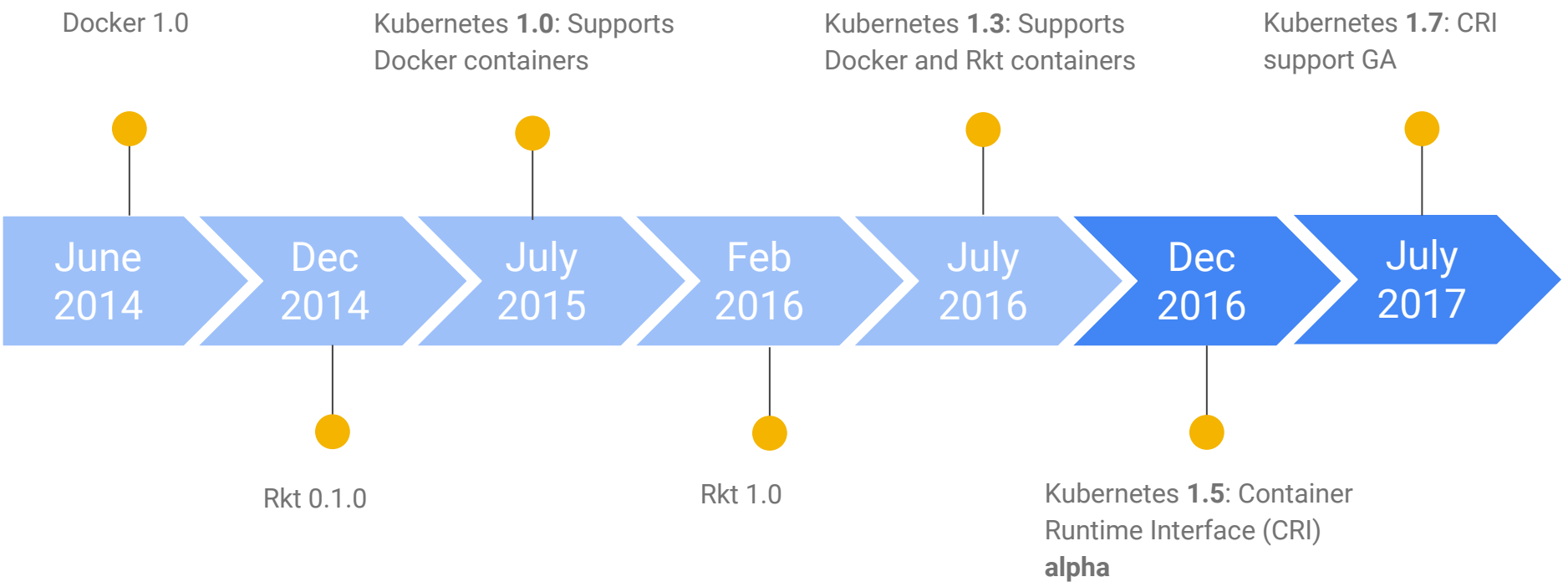
- **Runtime spec (runc = Reference implementation)**
- **Image spec**
- **Distribution spec (proposal)**

Use API/Interfaces to Container Runtimes



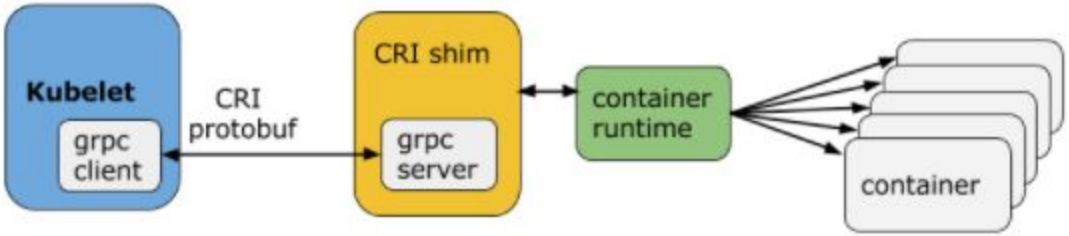


Standardization became a fact



What is Container Runtime Interface (CRI)?

- A gRPC interface and a group of libraries
- Enables Kubernetes to use a wide variety of container runtimes
- Introduced in Kubernetes 1.5
- GA in Kubernetes 1.7



CRI Implementations

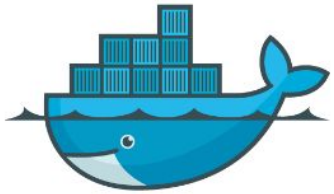
cri-o

cri-containerd

rktlet

frakti

virtlet



dockershim

CRI-O

- Open source & Open governance
- Lean, Stable, Secure and BORING!
  - Tied to the **CRI**
  - No features that can mine **stability** and **performance**
  - **Shaped** around Kubernetes
  - **Only supported user** is Kubernetes
  - **Versioning and Support** are tied to Kubernetes

Man, this guy is so boring!

When is the live demo?

@jorgemoralespou

Demo script

SHOW DOCKER AND STOP IT
$ docker images
$ systemctl stop docker
$ docker images

RUN A CONTAINER WITH OC/KUBECTL
$ kubectl get pods
$ kubectl run --image=nginx --port=80 nginx
$ kubectl expose deployment nginx --port=80
$ kubectl get svc
$ curl http://<CLUSTER-IP>
    $ oc expose svc/nginx-http
    $ oc get route
$ kubectl get pods
$ kubectl logs -f <POD_NAME>
$ kubectl exec -it <POD_NAME> sh
$ docker ps
$ runc list (show that containers are run by runc)


WOW!

What if I want to try it?

```
$ minikube start \
    --network-plugin=cni \
    --container-runtime=cri-o \
    --bootstrapper=kubeadm
```


KEEP
CALM
THERE'S NO
DOCKER
CLI

libpod/podman

Library (libpod) and CLI (podman) for managing OCI-based Pods, Containers, and Container Images

- Replacement for docker cli
  - known CLI
- Integrated with CRI-O (soon)
- **No daemon running**

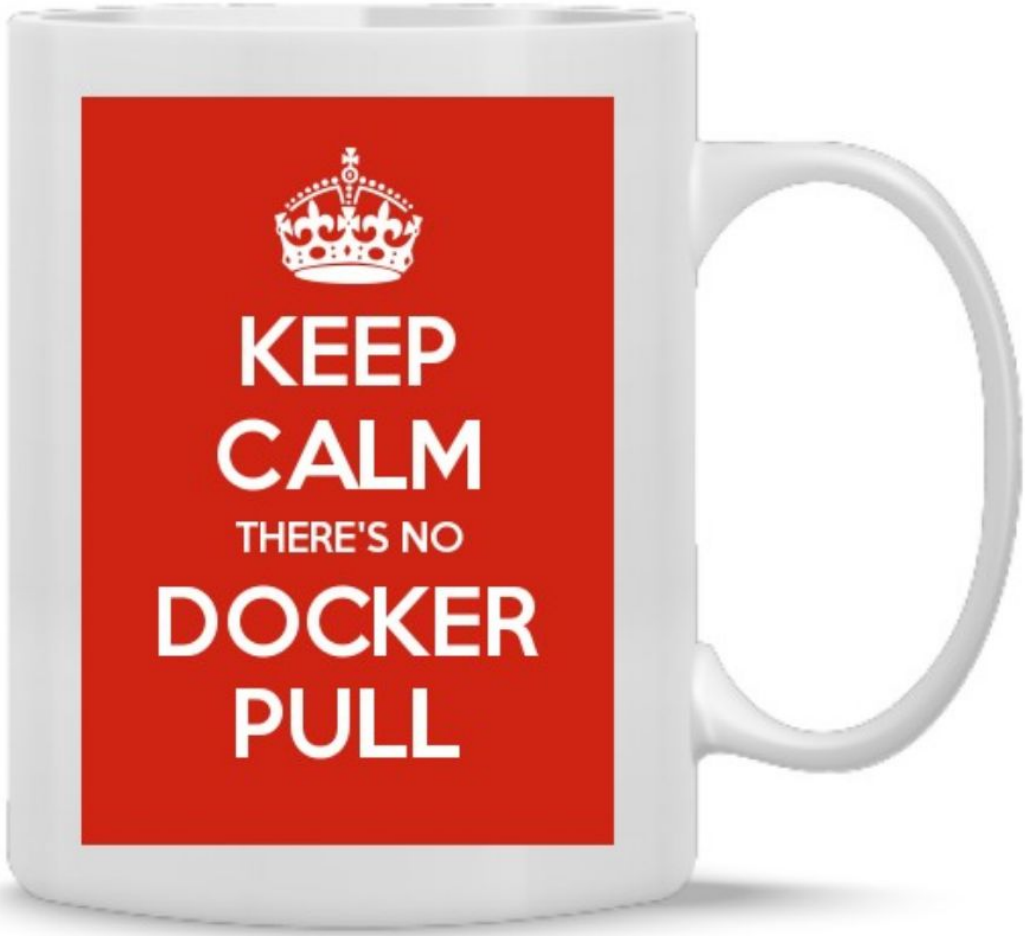

I wanna
SEE

Demo script

$ podman ps
$ podman images
$ podman run -it --rm -p 8080:80 nginx
$ <ANOTHER TERMINAL> curl http://localhost:8080
$ <ANOTHER TERMINAL> podman ps
$ <ANOTHER TERMINAL> podman logs <CONTAINER_ID>
$ <ANOTHER TERMINAL> podman exec -t <CONTAINER_ID> sh
$ <ANOTHER TERMINAL> ls /etc/nginx

$ podman images
$ podman run -t fedora echo "Hello Riga"
$ podman ps -a

$ cd Dockerfiles
$ podman build -t rigacontainer .
$ podman images
$ podman run -it rigacontainer cat /hello
$ podman ps -a
$ podman rm --all


KEEP
CALM
THERE'S NO
DOCKER
PULL

skopeo

- **Copy** images from/to (multiple transports/storages):
  - containers-storage:docker-reference
  - dir:path
  - docker://docker-reference
  - docker-archive:path[:docker-reference]
  - docker-daemon:docker-reference
  - oci:path:tag
  - ostree:image[@/absolute/repo/path]
- **Inspect** images
- **Delete an** image from a repository
- **Standalone binary / No daemon running**
- Perfect for pipelines (e.g. Jenkins)


I wanna
SEE

Demo script

$ systemctl start docker
$ docker images
$ podman images

COPY IMAGE
$ skopeo copy containers-storage:riga/example:latest docker-daemon:riga/example:latest
$ docker images
$ docker run -it --rm riga/example cat /hello
$ docker ps -a
$ docker rm $(docker ps -qa)
$ skopeo inspect docker-daemon:riga/example:latest
$ skopeo inspect docker:docker.io/library/fedora:latest


KEEP
CALM
THERE'S NO
DOCKER
BUILD

buildah

- Build images
- **No daemon running**
- shell-like syntax
- Build from Dockerfile(s)


I wanna
SEE

Demo script

```
$ cd ~/Dockerfiles
$ skopeo copy containers-storage:registry.fedoraproject.org/fedora:latest docker-daemon:fedora:latest
$ docker build -t riga/example-docker .

$ docker history riga/example
$ docker history riga/example-docker

$ buildah bud -t riga/buildah-dockerfile

$ container=(buildah from fedora)
$ echo $container
$ buildah containers
$ buildah config --author "Jorge" --label "METADATA=Built with buildah" $container
$ buildah inspect $container
$ buildah run $container bash
# echo “Hello Riga, built by Buildah” > /hello
# ls /
# cat /hello
# exit
$ buildah commit $container riga/example-buildah
$ podman run -it riga/example-buildah cat /hello
```


What else?

@jorgemoralespou

Daemon-less Dockerfile builds

- Consume a Dockerfile, but build image without a docker daemon
- Pros
  - Docker build-like experience (just write a Dockerfile)
  - Potentially more control over image layers (combine or shard)
  - Aim is for greater security
- Cons
  - Dockerfile fidelity might make difficult some use cases
  - Different approaches to image layer construction

Daemon-less Dockerfile builds

- Buildah
  - a tool that facilitates building OCI container images
- Img
  - Standalone, daemon-less, unprivileged Dockerfile and OCI compatible container image builder.
  - The commands/UX are the same as docker (drop-in replacement)
- Kaniko
  - kaniko is a tool to build OCI container images from a Dockerfile, inside a container or Kubernetes cluster
  - executes each command within a Dockerfile completely in userspace
- more...

Dockerfile-less builds

- User input is source / intent: “I want to run a Node.js web server”
- Pros:
  - Less configuration
  - Tools can intelligently build layers, better/safe layer caching
  - Docker image best practices can be codified into tools
- Cons:
  - Less flexible - Opinionated builds
  - Very fragmented across vendors, no real standard

Dockerfile-less builds

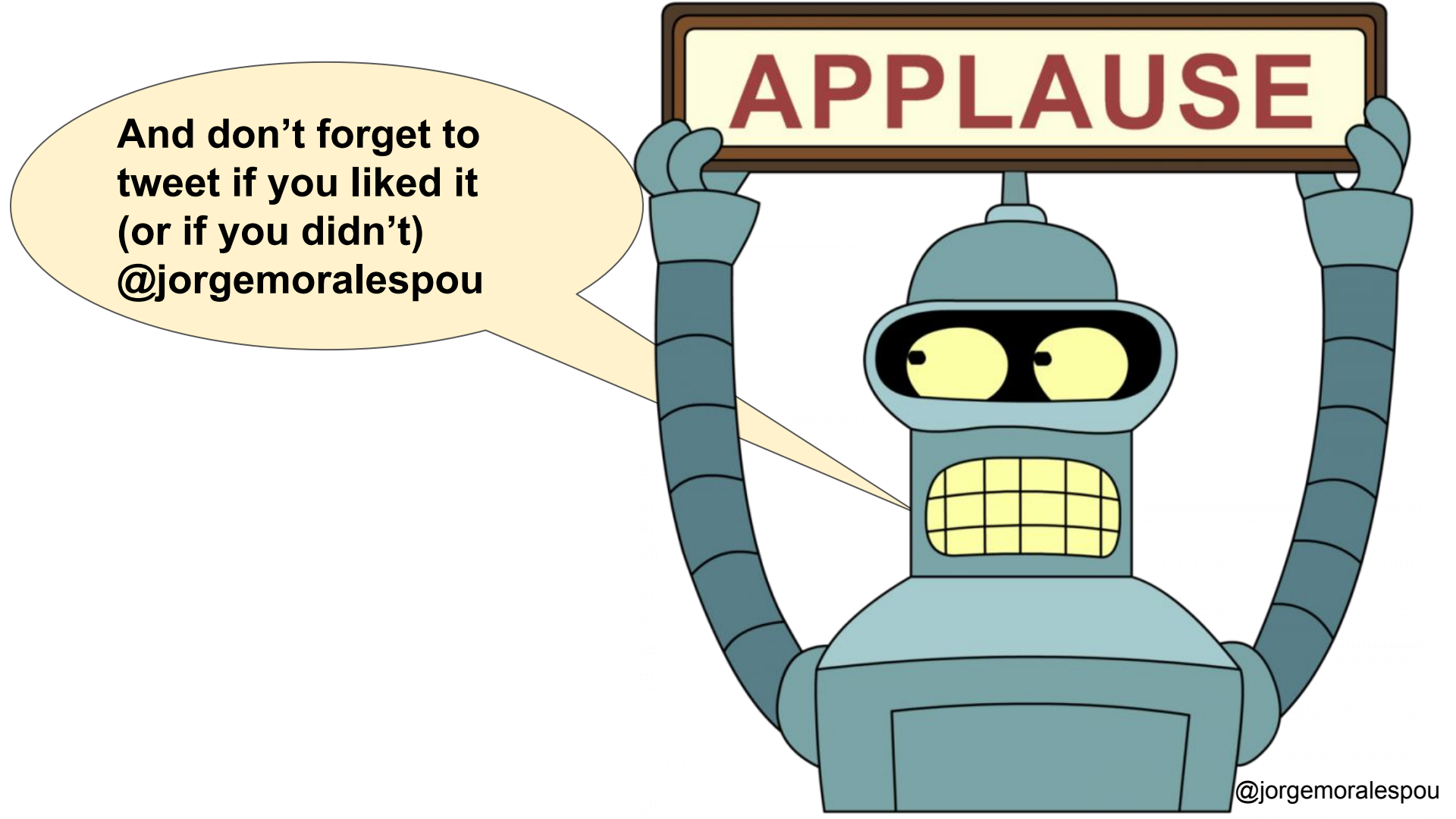
- Source to Image
  - User provides source, source gets built+layered into an application image
  - Dependent on ecosystem of framework/language builder images
- Buildpacks
  - Invented by Heroku, adopted by Cloud Foundry / Deis
  - User provides source, “build” produces “slug”, “export” produces container image
- FTL (Faster than light)
  - Purpose-built source to image builders per-language, goal is layer-per-dependency
  - Insight: turn build incrementality into deploy incrementality
- Bazel
  - Google’s OSS build system, supports declarative image builds
  - Used for user-mode Docker image builds for 3+ years

“Our ancestors called it magic, but you call it [computer] science.

I come from a land where they are one and the same.”

—Thor

@jorgemoralespou


And don’t forget to tweet if you liked it (or if you didn’t) @jorgemoralespou
APPLAUSE
@jorgemoralespou