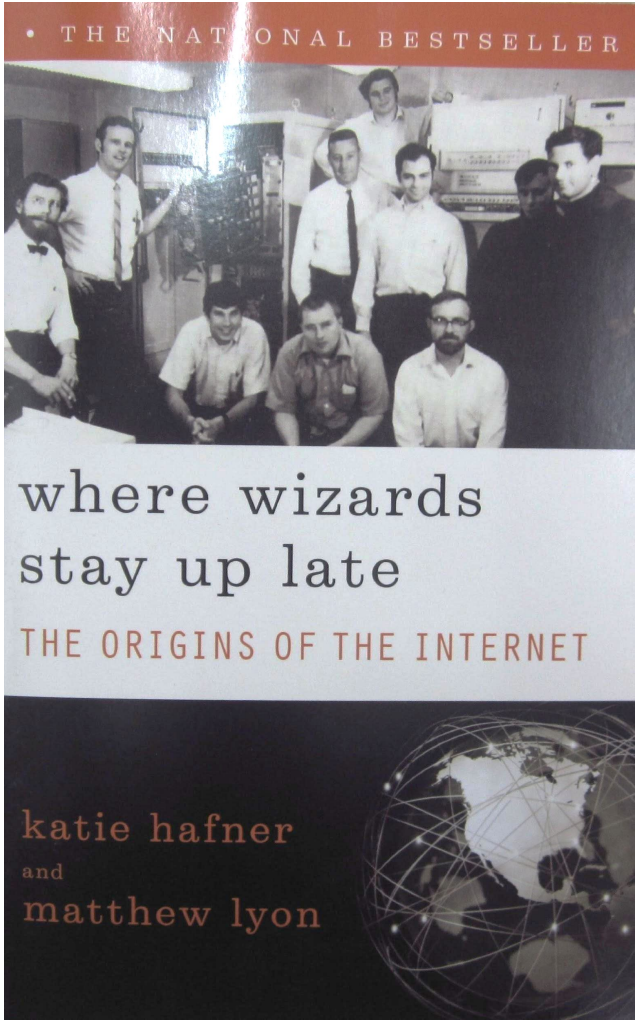
OPEN SOURCE
LEADERSHIP SUMMIT

What's Running My Containers?

A review of runtimes & standards

Phil Estes, Distinguished Engineer, IBM Cloud, @estesp

THE LINUX FOUNDATION



OPEN SOURCE
LEADERSHIP SUMMIT

Where Wizards Stay Up Late

THE ORIGINS OF THE INTERNET

Katie Hafner and Matthew Lyon

THE LINUX FOUNDATION

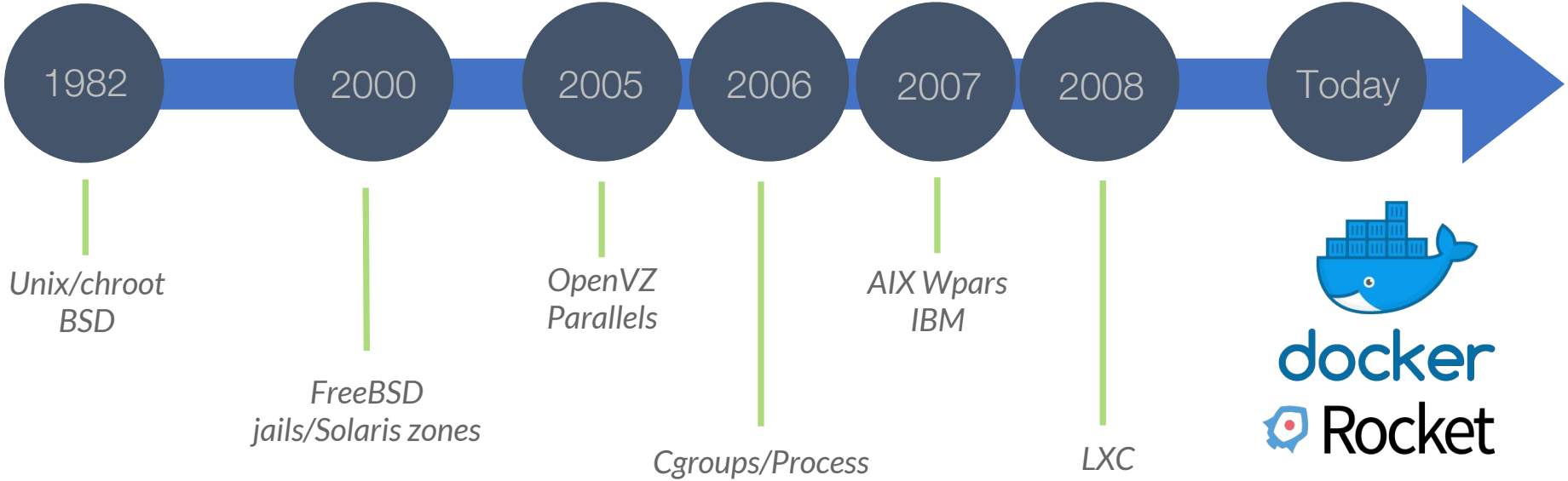
Standards

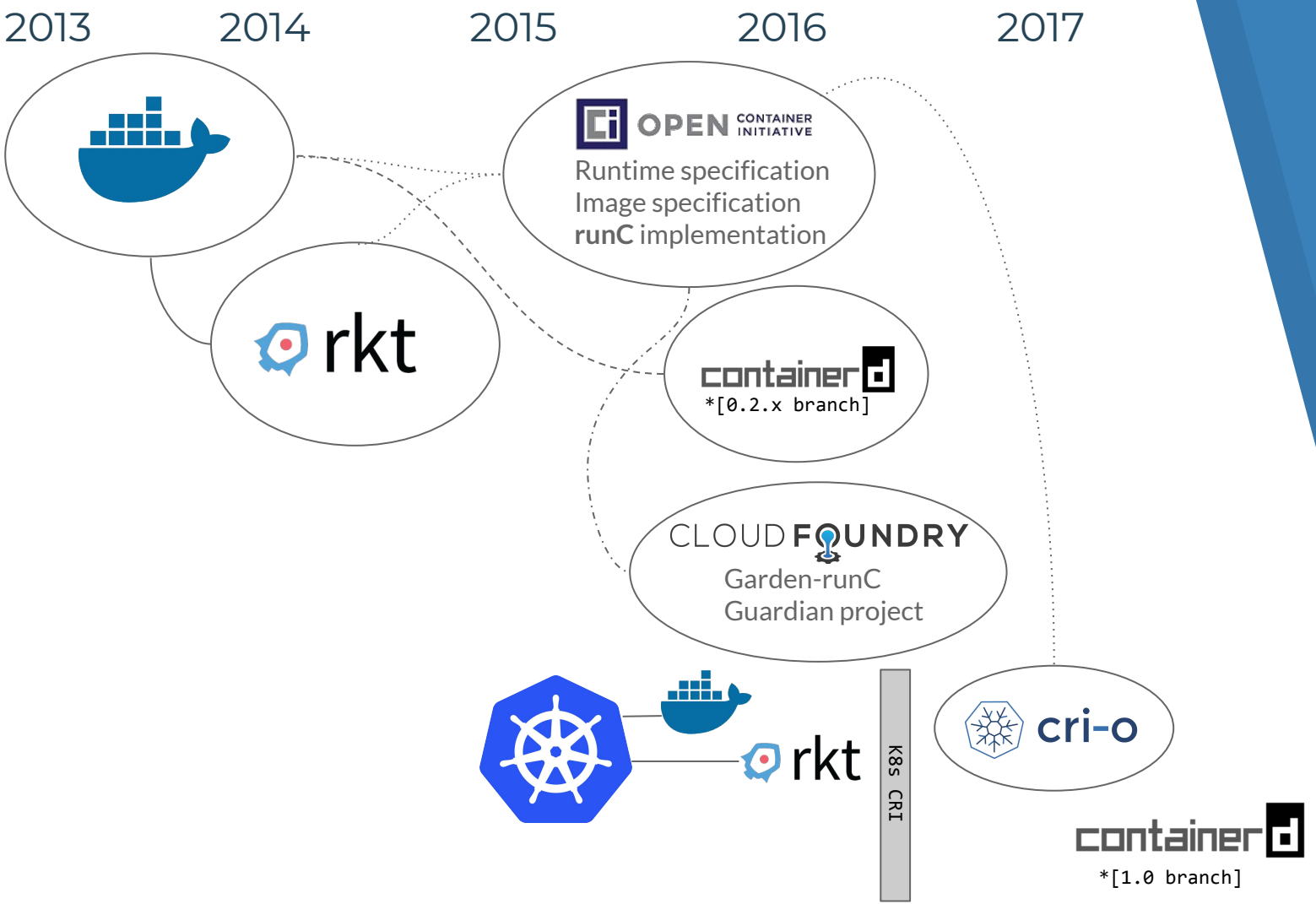
But the less formal meaning seemed even more fitting. “The other definition of protocol is that **it’s a handwritten agreement between parties, typically worked out on the back of a lunch bag**,” Cerf remarked, “which describes pretty accurately how most of the protocol designs were done.”

Proclamations of officialness didn’t further the Net nearly so much as throwing technology out onto the Net to see what worked. And **when something worked, it was adopted**.

“Standards **should be discovered, not decreed**,” said one computer scientist in the TCP/IP faction. Seldom has it worked any other way.

A Brief Container History




2013
2014
2015
2016
2017
OPEN CONTAINER INITIATIVE
Runtime specification
Image specification
runC implementation
rkt
containerd
*[0.2.x branch]
CLOUD FOUNDRY
Garden-runC
Guardian project
rkt
K8s CRI
cri-o
containerd
*[1.0 branch]

- A Linux Foundation **Collaborative Project**
- **Free** from specific vendor control / an open ecosystem
- Includes:
  - a **runtime** specification
  - reference runtime* (**runc**)
  - an **image format** specification
  - an image **distribution** spec (2019)

seeded with **runc + **libcontainer** by Docker*

> Announced June 20th, 2015
> Charter signed on December 8th, 2015
> 37 member companies
> Initial specifications reached 1.0 in June 2017

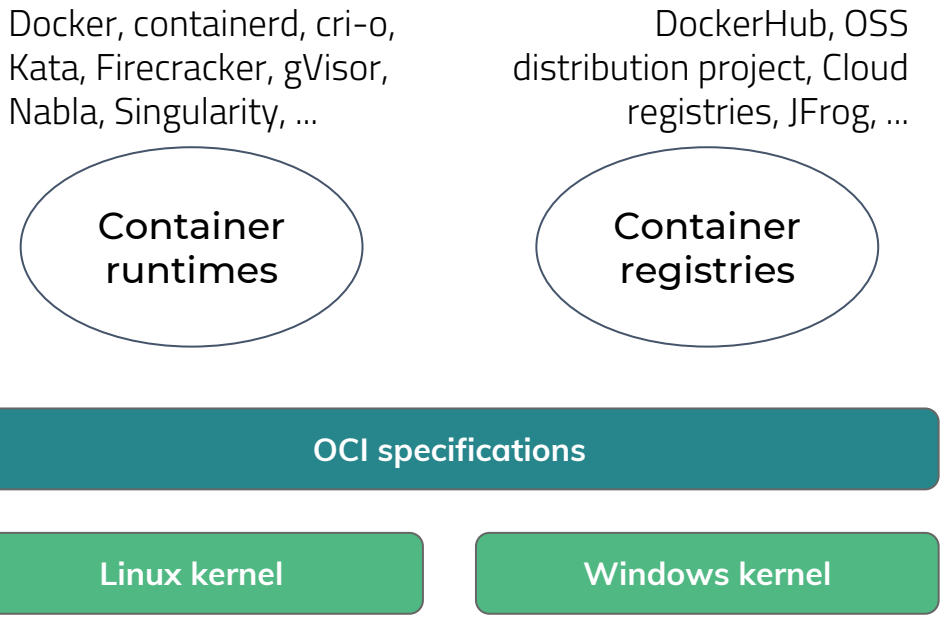
https://opencontainers.org
https://github.com/opencontainers

runc

Created in June 2015
> 18 releases (1.0.0-rc6 : Nov 2018)
> 246 contributors
> OCI maintained/governance
> Used by Docker, containerd, cri-o, garden-runc, cycle.io, among others

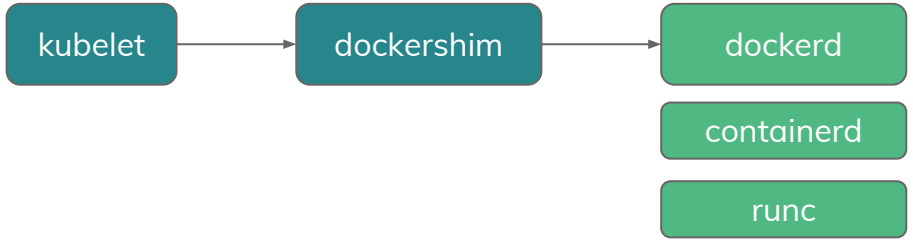
- Runc is a client wrapper around the pre-existing **libcontainer** library project
- Runc is one **implementation** of the OCI runtime specification
- **Scope** of runc is clearly limited by OCI charter: no networking, image handling/resolution, storage support
- Enablement of **low-level OS features** happen here: ambient caps, rootless containers, new cgroup support, and so on
- **Daemon-less** operation; wrapping code must handle any broader node and cluster level container management

A Standard Container Substrate



Kubernetes doesn't run containers

https://github.com/kubernetes/kubernetes/tree/release-1.4/pkg/kubelet/dockershim



CLOUD NATIVE COMPUTING FOUNDATION

OPEN CONTAINER INITIATIVE

The Kubernetes CRI

Monday, December 19, 2016

Introducing Container Runtime Interface (CRI) in Kubernetes

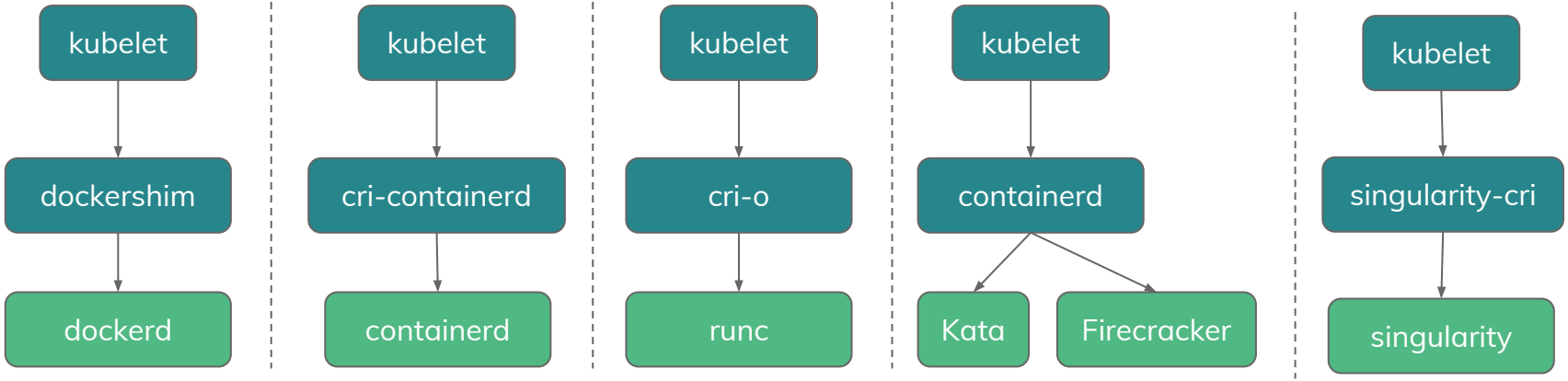
Editor's note: this post is part of a series of in-depth articles on what's new in Kubernetes 1.5

At the lowest layers of a Kubernetes node is the software that, among other things, starts and stops containers. We call this the "Container Runtime". The most widely known container runtime is Docker, but it is not alone in this space. In fact, the container runtime space has been rapidly evolving. As part of the effort to make Kubernetes more extensible, we've been working on a new plugin API for container runtimes in Kubernetes, called "CRI".

What is the CRI and why does Kubernetes need it?

Each container runtime has it own strengths, and many users have asked for Kubernetes to support more runtimes. In the Kubernetes 1.5 release, we are proud to introduce the Container Runtime Interface (CRI) -- a plugin interface which enables kubelet to use a wide variety of container runtimes, without the need to recompile. CRI consists of a protocol buffers and gRPC API, and libraries, with additional specifications and tools under active development. CRI is being released as Alpha in Kubernetes 1.5.

What CRI Runtimes Exist?




```
kubelet --container-runtime {string}
        --container-runtime-endpoint {string}
```

CRI Implementations

- A stable, core, performant core container runtime for the cloud
- Has a CRI implementation, and is a CNCF graduated project

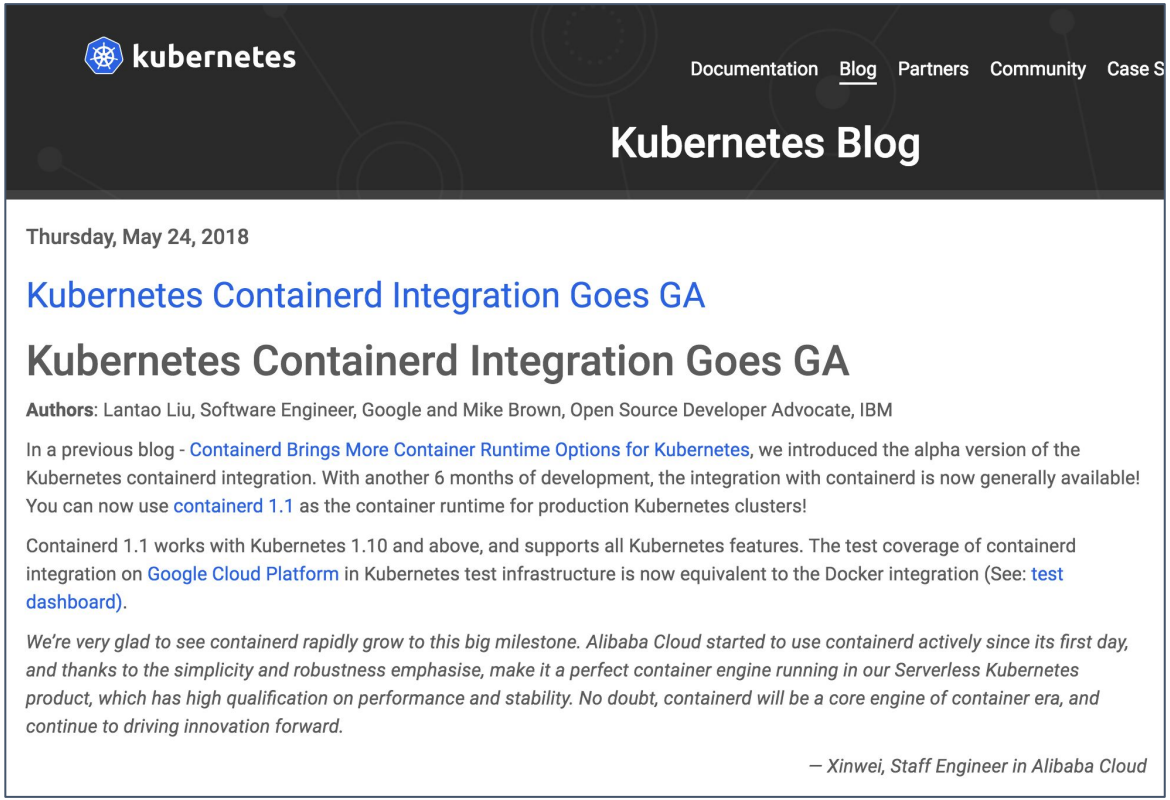
- "all the runtime Kubernetes needs and nothing more"; RH created
- CRI implementation over runc and 2 open libraries; K8s incubator

- Intel Clear Containers + Hyper.sh combined project
- Lightweight virtualization (KVM/qemu) under cri-o and containerd

- Amazon open source project announced Nov 2018; lightweight virt.
- Uses Rust-based VMM instead of qemu; plugs into containerd

- CRI implementation over Sylabs Singularity runtime project
- Userbase traditionally from academia/HPC use cases

Containerd + CRI



kubernetes

Documentation Blog Partners Community Case S

Kubernetes Blog

Thursday, May 24, 2018

Kubernetes Containerd Integration Goes GA

Kubernetes Containerd Integration Goes GA

Authors: Lantao Liu, Software Engineer, Google and Mike Brown, Open Source Developer Advocate, IBM

In a previous blog - Containerd Brings More Container Runtime Options for Kubernetes, we introduced the alpha version of the Kubernetes containerd integration. With another 6 months of development, the integration with containerd is now generally available! You can now use containerd 1.1 as the container runtime for production Kubernetes clusters!

Containerd 1.1 works with Kubernetes 1.10 and above, and supports all Kubernetes features. The test coverage of containerd integration on Google Cloud Platform in Kubernetes test infrastructure is now equivalent to the Docker integration (See: test dashboard).

We're very glad to see containerd rapidly grow to this big milestone. Alibaba Cloud started to use containerd actively since its first day, and thanks to the simplicity and robustness emphasise, make it a perfect container engine running in our Serverless Kubernetes product, which has high qualification on performance and stability. No doubt, containerd will be a core engine of container era, and continue to driving innovation forward.

— Xinwei, Staff Engineer in Alibaba Cloud

https://kubernetes.io/blog/2018/05/24/kubernetes-containerd-integration-goes-ga/

CRI Product Landscape

- **GKE**: containerd-based K8s clusters in **beta**/selectable; default is **Docker**
- **IBM Cloud**: containerd-based clusters in **production** (all versions)
- **Azure**: OSS acs-engine includes containerd; AKS uses **Docker**; (but CRI-O for OpenShift deployment)
- **Amazon**: EKS uses **Docker by default**; Firecracker using **containerd**
- **CloudFoundry**: Eirini project (CF on K8s) using **containerd**; pre-Eirini (non-K8s-based) used **runc**, now **containerd**
- **OpenShift**: prior versions used RHEL-Docker (1.12/13); **cri-o** GA in OpenShift during 2018
- **ICP**: IBM private cloud offering **defaults to Docker**; **containerd** in tech preview

OCI Network Effect

Singularity

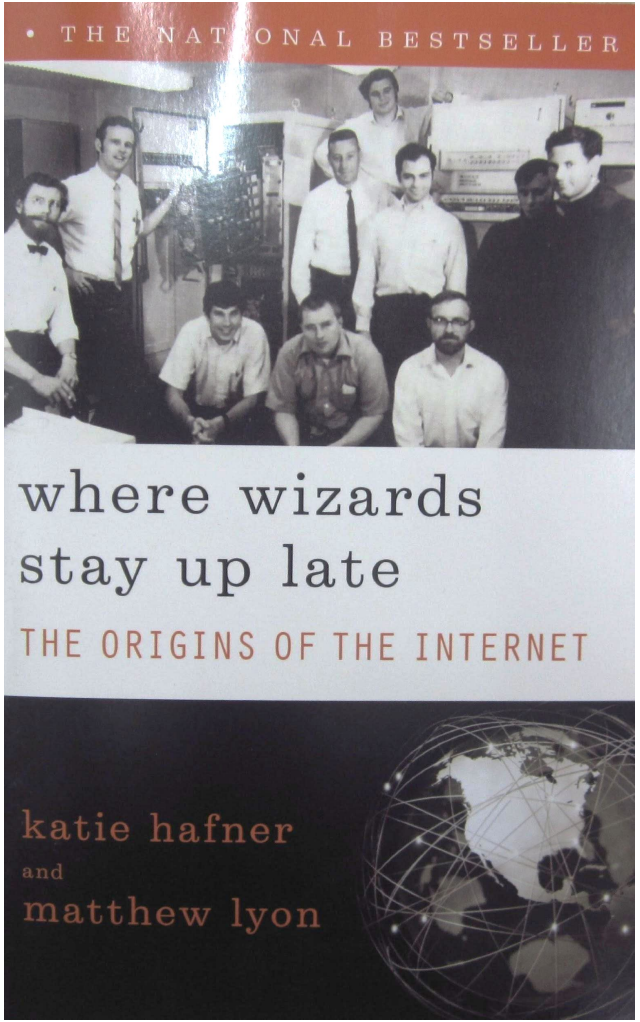
- Added full OCI support in v3.1.0 of Singularity (Feb 2019)
- Can import/run OCI images/specs using singularity project

LXC

- Added OCI support in v3.0.0 of LXC (May 2018)
- Can download/run OCI format containers with LXC runtime

Summary

- **Positive outcomes**
  - OCI created a **level playing field** whereby implementers of runtimes and higher-layer stacks could have **complete interoperability** via OCI standards
  - Good cross-industry **collaboration** has delivered on stable, "boring" container runtime technology; higher layers can provide choice in implementations
  - Network effects driving OCI **interoperability** to "non-Docker" use cases
  - CRI makes runtime choice a reality for Kubernetes as a common substrate
- **Work in progress**
  - Choice can be **confusing** to those outside our bubble
  - **Common** tooling choices/strategies
  - Keeping the **momentum**; OCI now standardizing image distribution (registry API)



OPEN SOURCE
LEADERSHIP SUMMIT

"The process of technological development is like building a cathedral," remarked Baran years later. "Over the course of several hundred years new people come along and each lays down a block on top of the old foundations, each saying, 'I built a cathedral.'Next month another block is placed atop the previous one. Then comes along an historian who asks, 'Well, who built the cathedral?' Peter added some stones here, and Paul added a few more. If you are not careful, you can con yourself into believing that you did the most important part. But the reality is that each contribution has to follow onto previous work. Everything is tied to everything else."

THE LINUX FOUNDATION


OPEN SOURCE
LEADERSHIP SUMMIT
THE LINUX FOUNDATION