CNI, CRI, and OCI - Oh My!

Who are we ?

Elsie Phillips

Paul Burt

This talk is

standards + containers

What’s a standard?

Something those ISO folks make

“Whatever the country, whatever the language, we are always ISO”

A standard we know: Javascript

I hate javascript

(Why would you use that as your example?)

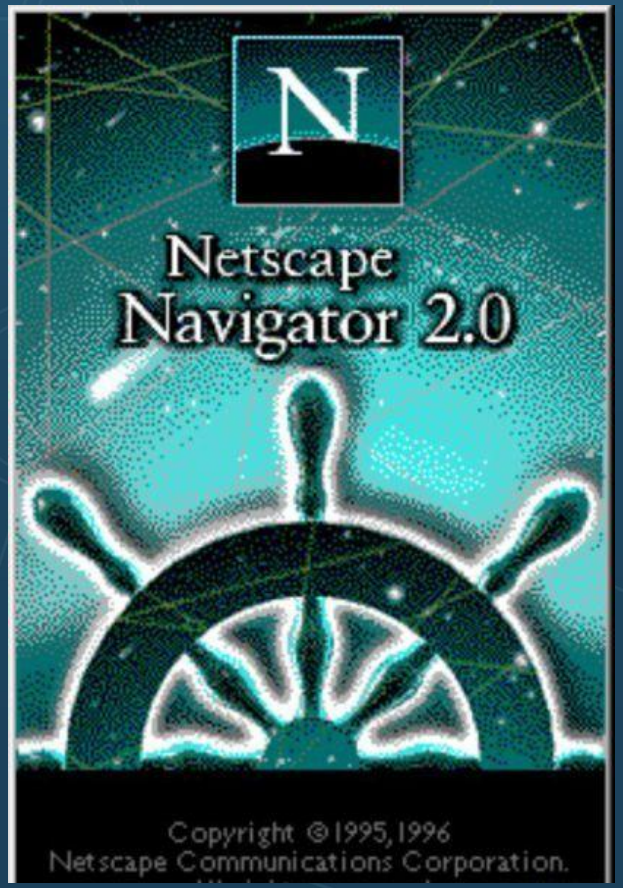
New Cool YAHOO! Today's News More Yahoos

Get Local YAHOO! CELEBRATE AND WIN. HOLIDAY EXTRAVAGANZA CLICK HERE. Weekly Picks

Search Options

Yellow Pages - People Search - City Maps -- Stock Quotes - Sports Scores

- **Arts and Humanities** - Architecture, Photography, Literature...
- **Business and Economy [Xtra!]** - Companies, Investments, Classifieds...
- **Computers and Internet [Xtra!]** - Internet, WWW, Software, Multimedia...
- **Education** - Universities, K-12, College Entrance...
- **Entertainment [Xtra!]** - Cool Links, Movies, Music, Humor...
- **Government** - 96 Elections, Politics **[Xtra!]**, Agencies, Law, Military...
- **Health [Xtra!]** - Medicine, Drugs, Diseases, Fitness...
- **News and Media [Xtra!]** - Current Events, Magazines, TV, Newspapers...
- **Recreation and Sports [Xtra!]** - Sports, Games, Travel, Autos, Outdoors...
- **Reference** - Libraries, Dictionaries, Phone Numbers...
- **Regional** - Countries, Regions, U.S. States...
- **Science** - CS, Biology, Astronomy, Engineering...
- **Social Science** - Anthropology, Sociology, Economics...


N
Netscape
Navigator 2.0
Copyright ©1995,1996
Netscape Communications Corporation.


About Internet Explorer
Microsoft
Internet
Explorer 4.0
Version 4.72.3110
Cipher Strength: 40-bit
Update versions: ;SP1;
Product ID:
Copyright Information
OK

You could hate JS so much more...

You uch

You would hate JS so much more...

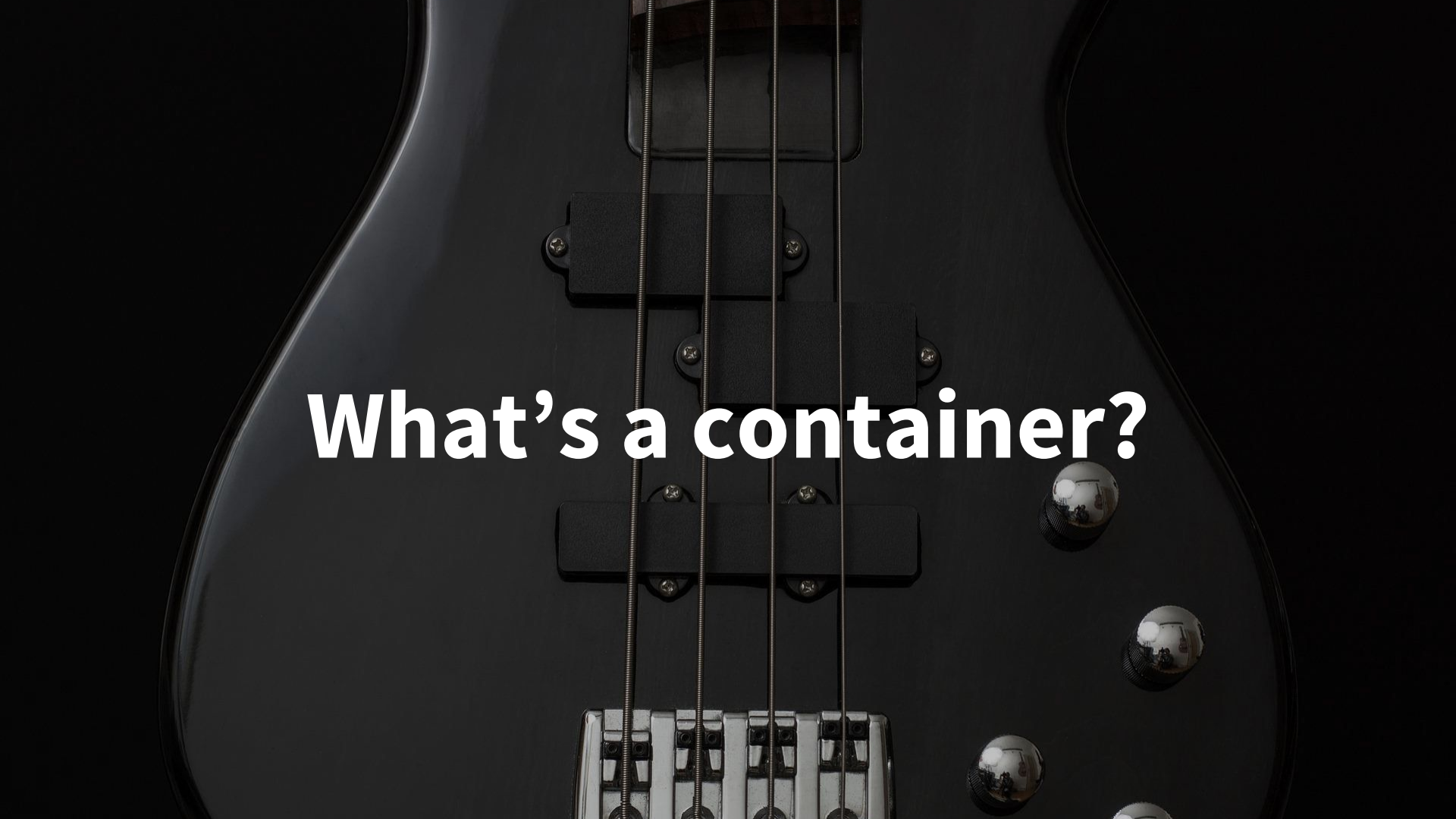



The same way JS enables multiple browsers to thrive


Frakti

The hypervisor-based container runtime for Kubernetes

Frakti lets Kubernetes run pods and containers directly inside hypervisors via runV. It is light weighted and portable, but can provide much stronger isolation with independent kernel than linux-namespace-based container runtimes.



What’s a container?

What's a container?

(Oh no, are these noobs really doing this at *Kubecon* ?)

What is a container?

It’s a TAR file.

AKA

Containers are . . .

TAR files

Cgroups
Chroot
Unshare
Nsenter
Bind mounts

Linux Magic

Core OS

For More on WHAT containers are

Containers From Scratch

By Eric Chiang

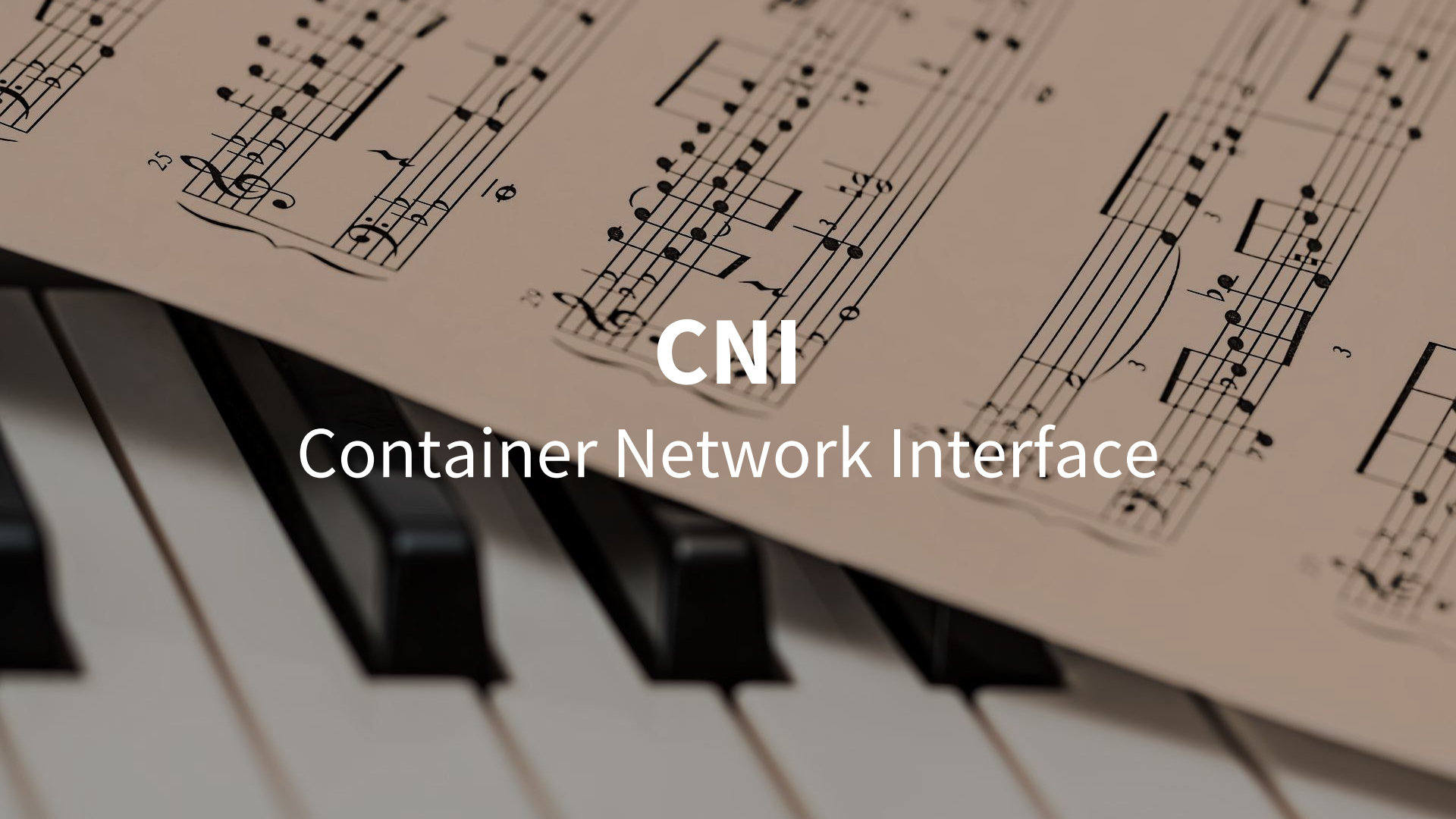
Best Practices for Containerized Environments

By Brian “Redbeard”

Why containers?

idk, why do ducks float?

Hell is other people's development environment



CNI

Container Network Interface

CNI

CNI - the Container Network Interface

What is CNI?

CNI (*Container Network Interface*), a Cloud Native Computing Foundation project, consists of a specification and libraries for writing plugins to configure network interfaces in Linux containers, along with a number of supported plugins. CNI concerns itself only with network connectivity of containers and removing allocated resources when the container is deleted. Because of this focus, CNI has a wide range of support and the specification is simple to implement.

As well as the specification, this repository contains the Go source code of a library for integrating CNI into applications and an example command-line tool for executing CNI plugins. A separate repository contains reference plugins and a template for making new plugins.

CNI

CNI - the Container Network Interface

What is CNI?

CNI (*Container Network Interface*), a Cloud Native Computing Foundation project, consists of a specification and libraries for writing plugins to configure network interfaces in Linux containers, along with a number of supported plugins. CNI concerns itself only with network connectivity of containers and removing allocated resources when the container is deleted. Because of this focus, CNI has a wide range of support and the specification is simple to implement.

As well as the specification, this repository contains the Go source code of a library for integrating CNI into applications and an example command-line tool for executing CNI plugins. A separate repository contains reference plugins and a template for making new plugins.

Zoom, Enhance!

CNI

CNI concerns itself only with network connectivity of containers and removing allocated resources when the container is deleted.

CNI vs CNM

Muhammad Ali vs Joe Frazier

CoreOS

THE CONTAINER NETWORKING LANDSCAPE: CNI FROM COREOS AND CNM FROM DOCKER

16 Sep 2016 11:47am, by Lee Calcote

THE CONTAINER NETWORKING LANDSCAPE: CNI FROM COREOS AND CNM FROM DOCKER

"...**both** are driver-based models, or plugin-based, for creating and managing network stacks for containers."

THE CONTAINER NETWORKING LANDSCAPE: CNI FROM COREOS AND CNM FROM DOCKER

"CNM is designed to support the Docker runtime engine **only**."

Thursday, January 14, 2016

Why Kubernetes doesn't use libnetwork

Kubernetes has had a very basic form of network plugins since before version 1.0 was released — around the same time as Docker's libnetwork and Container Network Model (CNM) was introduced. Unlike libnetwork, the Kubernetes plugin system still retains its "alpha" designation. Now that Docker's network plugin support is released and supported, an obvious question we get is why Kubernetes has not adopted it yet. After all, vendors will almost certainly be writing plugins for Docker — we would all be better off using the same drivers, right?

Before going further, it's important to remember that Kubernetes is a system that supports multiple container runtimes, of which Docker is just one. Configuring networking is a facet of each runtime, so when people ask "will Kubernetes support CNM?" what they really mean is "will kubernetes support CNM drivers with the Docker runtime?" It would be great if we could achieve common network support across runtimes, but that's not an explicit goal.

Indeed, Kubernetes has not adopted CNM/libnetwork for the Docker runtime. In fact, we've been investigating the alternative Container Network Interface (CNI) model put forth by CoreOS and part of the App Container (appc) specification. Why? There are a number of reasons, both technical and non-

kubernetes

An open source system for automating deployment, scaling, and operations of applications.

Learn about Kubernetes

“Kubernetes is a system that supports multiple container runtimes, of which Docker is just one.”

when people ask "will Kubernetes support CNM?" what they really mean is "will kubernetes support CNM drivers with the Docker runtime?" It would be great if we could achieve common network support across runtimes, but that's not an explicit goal.

Indeed, Kubernetes has not adopted CNM/libnetwork for the Docker runtime. In fact, we've been investigating the alternative Container Network Interface (CNI) model put forth by CoreOS and part of the App Container (appc) specification. Why? There are a number of reasons, both technical and non-

Download Kubernet

Blog Archive

▶ 2017 (49)

CNCF Hosts Container Networking Interface (CNI)

By cncf | May 23, 2017 | Blog

Today, the Cloud Native Computing Foundation (CNCF) Technical Oversight Committee (TOC) voted to accept CNI (Container Networking Interface) as the 10th hosted project alongside Kubernetes, Prometheus, OpenTracing, Fluentd, Linkerd, gRPC, CoreDNS, containerd, and rkt.

Container-based applications are rapidly moving into production. Just as Kubernetes allows enterprise developers to run containers en masse across thousands of machines, containers at scale also need to be networked.

The CNI project is a network interface created by multiple companies and projects; including CoreOS, Red Hat OpenShift, Apache Mesos, Cloud Foundry, Kubernetes, Kurma and rkt. First proposed by CoreOS to define a common interface between the network plugins and container execution, CNI is designed to be a minimal specification concerned only with the network connectivity of containers and removing allocated resources when the container is deleted.

"The CNCF TOC wanted to tackle the basic primitives of cloud native and formed a working group around cloud native networking,"

CLOUD NATIVE COMPUTING FOUNDATION

About Projects Certification People Community Newsroom

CNCF Hosts Container Networking Interface (CNI)

13

“ … voted to accept CNI (Container Networking Interface) as the 10th hosted project”

containers and removing allocated resources when the container is deleted.

“The CNCF TOC wanted to tackle the basic primitives of cloud native and formed a working group around cloud native networking,”

TL;DR of how it works

Ain’t nobody got time to read specs

The **runtime** creates a network namespace

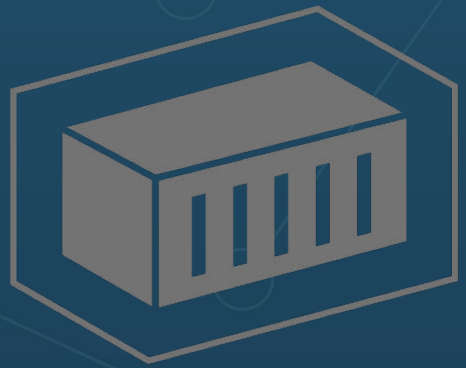
The **runtime** reads a JSON config

The **runtime** executes a plugin named by the config (with the ADD command)

The **plugin** finds out what to do from JSON streamed to stdin

The **plugin** does it’s thing

If there’s an error, the **runtime** tells the plugin to delete (DEL)



Otherwise, the **runtime** cleans up (DEL) at the end of the lifecycle

Example configuration

```
{
        "name": "mynet",
        "type": "bridge",
        "bridge": "mynet0",
        "isDefaultGateway": true,
        "forceAddress": false,
        "ipMasq": true,
        "hairpinMode": true,
        "ipam": {
                "type": "host-local",
                "subnet": "10.10.0.0/16"
        }
}
```

Notable developments this year

- IPv6 support
- plugin chaining
- port-forwarding

Amazon ECS CNI Plugins

CNI-Genie

CNI

All the cool plugins are doing it

CRI

Container Runtime Interface

kubernetes

An open source system for automating deployment, scaling, and operations of applications.

Learn about Kubernetes

Monday, December 19, 2016

Introducing Container Runtime Interface (CRI) in Kubernetes

Subscribe To Blog

Posts

Comments

@Kubernetesio

View on GitHub

#kubernetes-users

Stack Overflow

Download Kubernetes

Blog Archive

▶ 2017 (49)

kubernetes

An open source system for automating deployment, scaling, and operations of applications.

Learn about Kubernetes

Monday, December 19, 2016

Introducing Container Runtime Interface (CRI) in Kubernetes

Subscribe To Blog

Posts

Comments

"Docker and rkt were integrated directly and deeply into the kubelet source code through an internal and volatile interface."

▶ 2017 (49)

POWER CUBE 1910/USRU4P POWER CUBE USB by POWER C

Write a Review

Free shipping over $35 and Free Retu

~~$34.49~~

$15.95

Size

4 Outlets/2 USB

Buy 1	Buy 2	$1
$15.95/ea	**$15.15**/ea	

Add to Cart

CRI Timeline

Dec 12

2017

Mar 28

Jun 30

⋮

1.5 alpha out

1.6 Docker CRI gets beta + enabled by default

1.7 Docker CRI goes GA

CRI Timeline

Sep 29

1.8

CRI test suite + CLI tools.

Dec ??

1.9

CRI stats stats stats!

⋮

That’s exciting...

In a Mom & Dad got me socks for X-mas kind of way

README.md

CRI-O - OCI-based implementation of Kubernetes Container Runtime Interface

build passing | go report A+

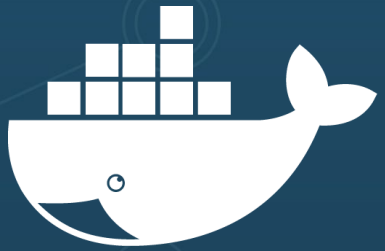
Status: Stable

Demo

CRI

We won’t need a different software ecosystem for every container format? Thank goodness.

OCI



docker

and the journey to standards

CoreOS

An image format
A container runtime
A log collection daemon
An init system and process babysitter
A container image build system
A remote management API

An image format

A container runtime

A log collection daemon

An init system and process babysitter

A container image build system

A remote management API

An image format

... the thing we want to standardise

Mid 2014

Docker Image Format Circa 2014

- Fluid format and evolution
  - No specification, just implementation
  - No guarantees of interoperability for other tool writers
- Not content-addressable
  - No way to verify integrity or leverage CAS
- No name delegation/discovery (e.g. MX records)
  - Centralised/siloed distribution
- No mechanism for signing
  - No way to attest content

Dec 2014

appc

App Container (appc)

CoreOS

appc image in a nutshell

- **Image Format (ACI)**
  - what does an application consist of?
- **Image Discovery**
  - how can an image be located?
- **Content-addressability**
  - what is the cryptographic id of an image?
- **Signing**
  - how is an image signed and verified?

April 2015

Docker v2.2 Image Format Circa 2015

- Versioned v2.0, v2.1, v2.2 schema
  - Still vendor-specific, but (mostly) documented!
- Content-addressable
- No name delegation/discovery
- Optional and separately-defined signing

Two separate worlds...

aka the "Container Wars"

OCI

June 2015-Present

OPEN CONTAINER INITIATIVE

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

POWER CUBE 1910/USRU4P POWER CUBE USB by POWER C

Write a Review

Free shipping over $35 and Free Retu

~~$34.49~~

$15.95

Size

4 Outlets/2 USB

Buy 1	Buy 2	$1
$15.95/ea	**$15.15**/ea	

Add to Cart

Why does OCI exist?

- Define **what a container is** in **an open way** so everyone can implement it
  - How to package, annotate, distribute, run, ...
  - Facilitate independent, interoperable tools

Why does OCI exist?

- Define **what a container is** in **an open way** so everyone can implement it
  - How to package, annotate, distribute, run, ...
  - Facilitate independent, interoperable tools
- Unify the best ideas from Docker, appc, etc
  - Content addressability, composability, signing
  - End the so-called "Container Wars"

OCI Members

amazon web services
anchore
APCERA
at&t
DATAMAN
CISCO
ContainerShip
CoreOS
CYCLE
DELL EMC
docker
EasyStack
facebook
FUJITSU
Goldman Sachs
Google
Hewlett Packard Enterprise
HUAWEI
IBM
Infoblox
InfoSiftr
intel
Joyent
KONTENA
MESOSPHERE
Microsoft
ORACLE
Pivotal
POLYVERSE
portworx
RANCHER
redhat
REPLICATED
resin.io
ROBIN
SUSE
sysdig
Twistlock
UNIVA
Verizon Labs
Virtuozzo
vmware
weaveworks
wercker

What makes up the standard?

The OCI standards

Two separate but connected specifications

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

OCI Image Spec

- Portable archive format
- Composed of:
  - image manifest
  - image index (optional)
  - filesystem layers
  - configuration

```
busybox/blobs/sha256/03b1be98f3f9b05cb57782a3a71a44aaf6ec695de5f4f8e6c1058cd42f
04953e
/tmp/asg # jq < busybox/index.json
{
  "schemaVersion": 2,
  "manifests": [
    {
      "mediaType": "application/vnd.oci.image.manifest.v1+json",
      "digest": "sha256:57b4b433672b7d0ee3c3ea274bda013bf090610cbf5c68455839f7c
1a94673fa",
      "size": 347,
      "annotations": {
        "org.opencontainers.image.ref.name": "latest"
      },
      "platform": {
        "architecture": "amd64",
        "os": "linux"
      }
    }
  ]
}
/tmp/asg #
```


2017 What's in a container? The OCI Answer

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

What happened to appc?

Image formats: a summarised history

	Docker v1	appc	Docker v2.2	OCI
Introduced	2013	December 2014	April 2015	April 2016
Content-addressable	No	Yes	Yes	Yes
Signable	No	Yes, optional	Yes, optional	Yes, optional
Federated namespace	Yes	Yes	Yes	Yes
Delegatable DNS namespace	No	Yes	No	Yes

Open Container Initiative (OCI) Releases v1.0 of Container Standards

By Open Container Initiative | July 19, 2017 | Announcement

Open, portable, vendor-neutral container specifications now available

SAN FRANCISCO, Calif. - 19 July, 2017 - The Open Container Initiative (OCI), an open source community for creating open industry standards around containers, today announced the debut release of its container runtime and image format specifications, comprised of Runtime Specification v1.0 (a specification for defining the lifecycle of a container) and Image Format Specification v1.0 (a specification for the container image format). Combined with efforts to create a formal certification program later this year, OCI is bringing a set of common, minimal, open standards and specifications around container technology to a reality.

OCI v1.0 specifications lay the foundation for container portability across different implementations to make it easier for customers to support portable container solutions. The OCI will launch a certification program shortly such that different implementations can demonstrate conformance to the specifications.

"The v1.0 release of the OCI specifications is a huge milestone for both the container community and the industry at large," said Chris Aniszczyk, Executive Director, OCI. "By creating these open, accessible specifications, along with early deployments, we are bringing the industry closer to portability and standardization. This is no small feat, and I am incredibly proud of the OCI community for all the hard work that went into this release."

The initial release comes following an integrated and collaborative effort among a diverse community made up of individual contributors and disparate organizations, including the project's over 40 member organizations. Formed in June of 2015, the OCI was launched with the express purpose of developing vendor neutral container standards that provide the industry the ability to fully commit to container technologies today without the fear of lock-in. OCI began with a specification describing container runtime behavior and expanded a year

So, that’s it?

Lol, nope
Lots to be done still

README.md

Container Storage Interface (CSI) Specification build passing

This project contains the CSI specification and protobuf files.

Standards will continue to evolve

Standards naturally lead to more options for users

And that’s something worth being passionate about.

Check out Open Cloud Services on CoreOS.com/blog