KubeCon | CloudNativeCon
North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022

A Containerd and Friends Update: What's New in Runtimes?

Phil Estes
Principal Engineer
AWS

Maksym Pavlenko
Software Engineer
Apple

Michael Zappa
Technical Program Manager
Microsoft

Mike Brown
Software Engineer
IBM Cloud

October 26, 2022

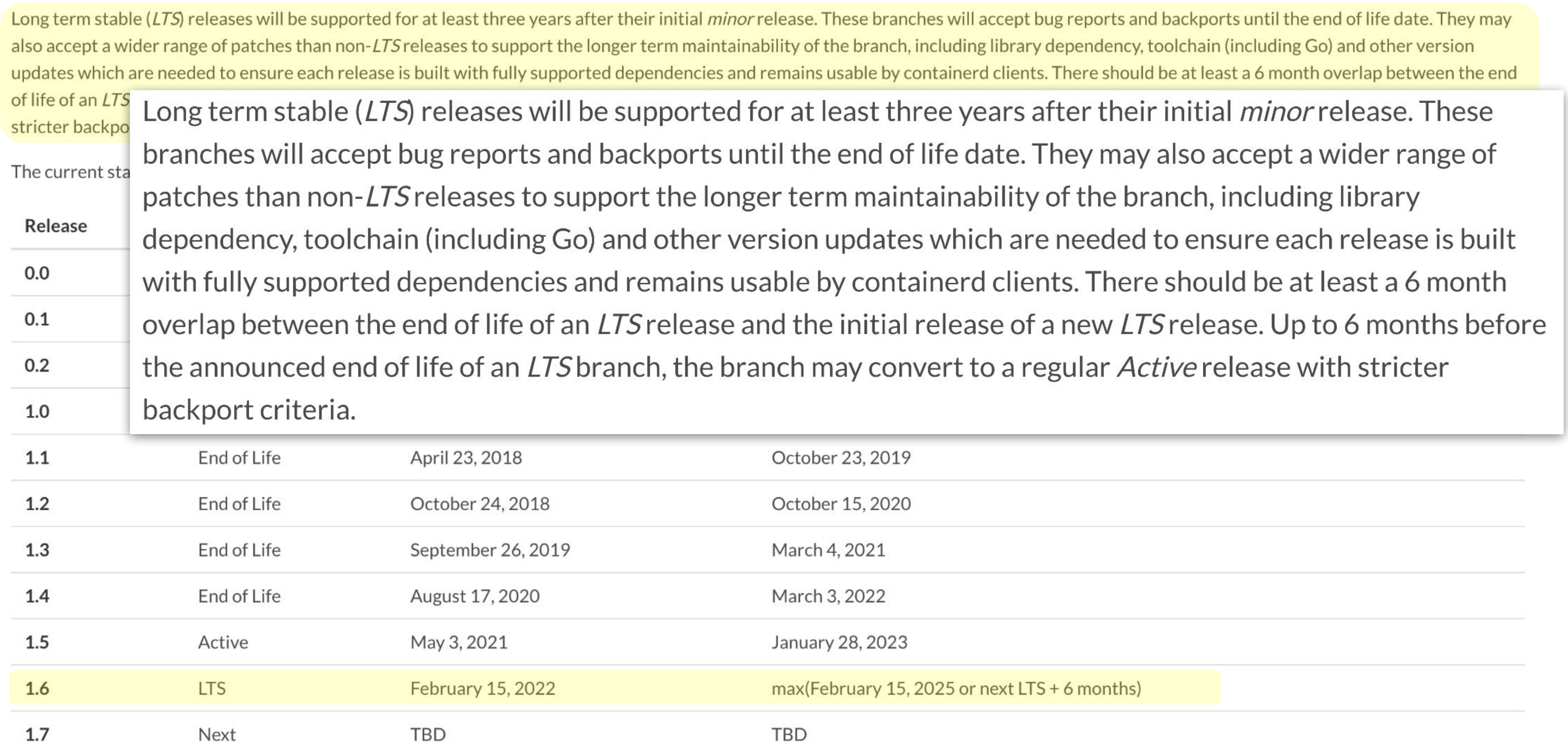
Community & Project Updates

Release updates
New LTS release
Community News

Releases

release/1.5	1.5.14 - Oct 2022; now extended support through end of January 2023
release/1.6	1.6.9 - Oct 2022; announcing as our LTS branch with 3 years of support
main	1.7-beta0: Available now! We’re starting our 1.7 pre-release series.

Long Term Support



Long term stable (*LTS*) releases will be supported for at least three years after their initial *minor* release. These branches will accept bug reports and backports until the end of life date. They may also accept a wider range of patches than non-*LTS* releases to support the longer term maintainability of the branch, including library dependency, toolchain (including Go) and other version updates which are needed to ensure each release is built with fully supported dependencies and remains usable by containerd clients. There should be at least a 6 month overlap between the end of life of an *LTS* release and the initial release of a new *LTS* release. Up to 6 months before the announced end of life of an *LTS* branch, the branch may convert to a regular *Active* release with stricter backport criteria.

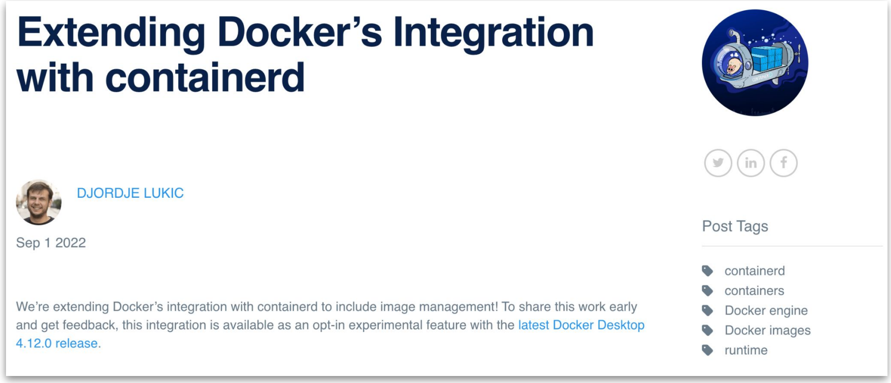
Long term stable (*LTS*) releases will be supported for at least three years after their initial *minor* release. These branches will accept bug reports and backports until the end of life date. They may also accept a wider range of patches than non-*LTS* releases to support the longer term maintainability of the branch, including library dependency, toolchain (including Go) and other version updates which are needed to ensure each release is built with fully supported dependencies and remains usable by containerd clients. There should be at least a 6 month overlap between the end of life of an *LTS*
stricter backpo

The current sta

Release			
0.0			
0.1			
0.2			
1.0			
1.1	End of Life	April 23, 2018	October 23, 2019
1.2	End of Life	October 24, 2018	October 15, 2020
1.3	End of Life	September 26, 2019	March 4, 2021
1.4	End of Life	August 17, 2020	March 3, 2022
1.5	Active	May 3, 2021	January 28, 2023
1.6	LTS	February 15, 2022	max(February 15, 2025 or next LTS + 6 months)
1.7	Next	TBD	TBD

Community Updates

Docker now has a roadmap to use image store/snapshotters directly from containerd rather than the existing graphdrivers



Extending Docker's Integration with containerd

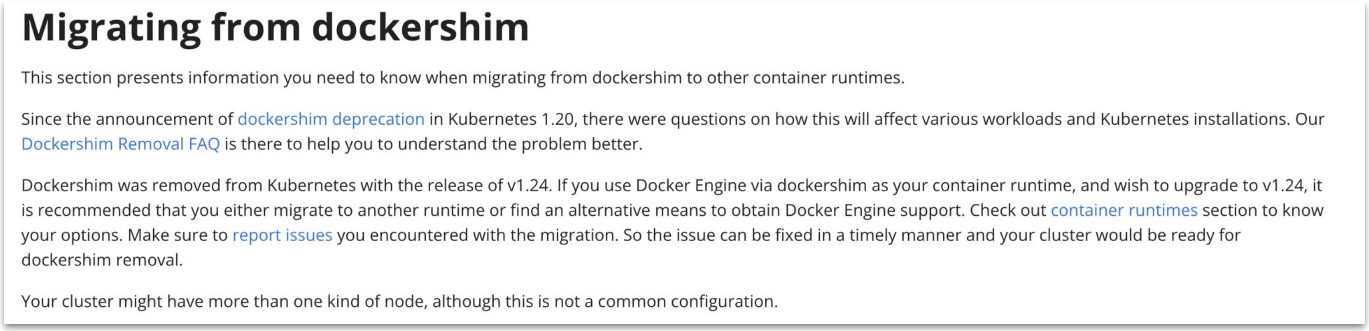
DJORDJE LUKIC

Sep 1 2022

We're extending Docker's integration with containerd to include image management! To share this work early and get feedback, this integration is available as an opt-in experimental feature with the latest Docker Desktop 4.12.0 release.

Post Tags

- containerd
- containers
- Docker engine
- Docker images
- runtime



Migrating from dockershim

This section presents information you need to know when migrating from dockershim to other container runtimes.

Since the announcement of dockershim deprecation in Kubernetes 1.20, there were questions on how this will affect various workloads and Kubernetes installations. Our Dockershim Removal FAQ is there to help you to understand the problem better.

Dockershim was removed from Kubernetes with the release of v1.24. If you use Docker Engine via dockershim as your container runtime, and wish to upgrade to v1.24, it is recommended that you either migrate to another runtime or find an alternative means to obtain Docker Engine support. Check out container runtimes section to know your options. Make sure to report issues you encountered with the migration. So the issue can be fixed in a timely manner and your cluster would be ready for dockershim removal.

Your cluster might have more than one kind of node, although this is not a common configuration.

Kubernetes removal of dockershim has driven greater use of containerd; also has increased contributor activity

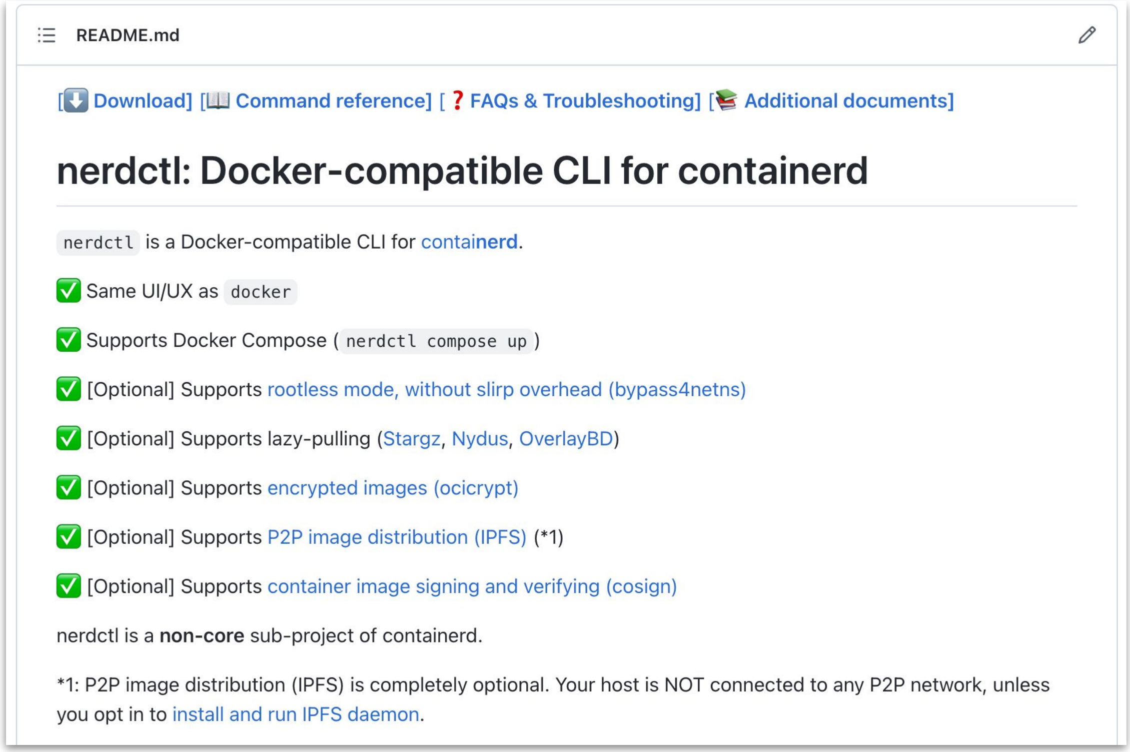
- **2020-2022**: More than 100% increase in regular contributors: from 30 to 70+
- Various surveys reflect anywhere from **50% - 200%** growth in containerd runtime market share in the past 12 months

Sub-project Updates

nerdctl client
Lazy-loading snapshotters
Rust integrations

nerdctl client

nerdctl is a Docker-compatible client for containerd with full rootless support and includes subproject features like lazy-loading, image encryption, and image signing



README.md

[⬇️ Download] [📖 Command reference] [❓FAQs & Troubleshooting] [📚 Additional documents]

nerdctl: Docker-compatible CLI for containerd

`nerdctl` is a Docker-compatible CLI for containerd.

✅ Same UI/UX as `docker`

✅ Supports Docker Compose (`nerdctl compose up`)

✅ [Optional] Supports rootless mode, without slirp overhead (bypass4netns)

✅ [Optional] Supports lazy-pulling (Stargz, Nydus, OverlayBD)

✅ [Optional] Supports encrypted images (ocicrypt)

✅ [Optional] Supports P2P image distribution (IPFS) (*1)

✅ [Optional] Supports container image signing and verifying (cosign)

nerdctl is a **non-core** sub-project of containerd.

*1: P2P image distribution (IPFS) is completely optional. Your host is NOT connected to any P2P network, unless you opt in to install and run IPFS daemon.

Latest Updates

- v1.0.0 released Oct 2022
- Supports lazy-loading snapshotters (added Nydus in addition to Stargz & OverlayBD support; SOCI coming soon?)
- New logging drivers: syslog, journald, fluentd (adding to json-file)
- New network drivers: macvlan, ipvlan (added to existing bridge driver)
- Experimental support for bypass4netns to accelerate rootless networking (up to 10x-100x faster, workload dependent)
- Lima joined the CNCF Sandbox 🎉 popularizing containerd and nerdctl for macOS users; Lima 0.13 release will include the new v1.0.0 nerdctl

Lazy-loading snapshotters

New innovations around container image lazy-loading have arrived in the past few years. The containerd snapshotter interface and proxy/remote snapshotter capability has allowed for innovation not tied directly to the containerd release lifecycle. A few examples are provided here.

stargz-snapshotter (contributed in Mar 2020 as a non-core sub-project)

- Uses the eStargz format; based on the stargz format created by Google's CRFS, but with additional features like runtime optimization and content verification
- Support now included in BuildKit and nerdctl as well as growing list of tools
- Experimental support for IPFS

SOCI-snapshotter (external)

- "Seekable-OCI" that uses the new OCI references WG capability to separately store a TOC/index as a reference to the main image (no image conversion/push step required)
- Created by AWS and open-sourced in summer 2022; built on core of stargz snapshotter; working together on potential refactor for shared codebase

nydus-snapshotter (contributed in Jan 2022 as a non-core sub-project)

- Works with overlaybd and accelerated-container-image components (also contributed as non-core subprojects) to provide implementation of block-based lazy loading image technology, used by Alibaba

Rust extensions

New additions to the Rust family in containerd

- GRPC client for containerd daemon
- Proxy snapshotters plugin
- Shim extensions
  - Protos and shim client
  - Binary logging
  - runc client
  - runc shim reference implementation

Rust extensions

Rust runc shim

- Offers same experience as Go version

```
fn main() {
    shim::run::<Service>("io.containerd.empty.v1", None)
}
```

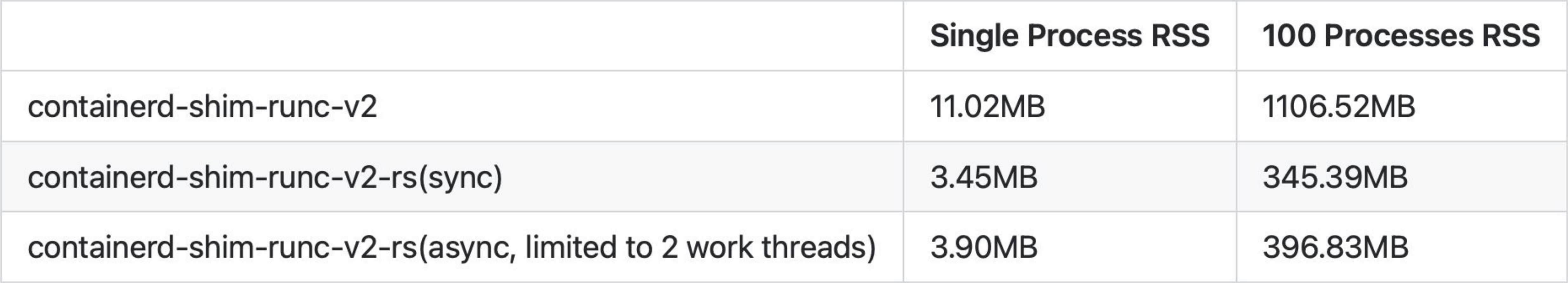
- Supports both sync and async

Rust extensions

Rust runc shim

- Reference implementation
- Contributed by Huawei Cloud
  - Running on 5000+ nodes in production
  - Decent resource utilization

16U32G Ubuntu 20.04, *containerd v1.6.8*, *runc v1.1.4*.



	Single Process RSS	100 Processes RSS
containerd-shim-runc-v2	11.02MB	1106.52MB
containerd-shim-runc-v2-rs(sync)	3.45MB	345.39MB
containerd-shim-runc-v2-rs(async, limited to 2 work threads)	3.90MB	396.83MB

New & Upcoming Features

Sandbox API (1.7)
Transfer Service (1.7)

Sandbox API

Sandbox API == New API for container groups

- **Controller interface to handle sandbox lifecycle**
  - pod-sandbox (extract from CRI)
  - microVM
  - VM
- **Shims provide Controller implementation**
- **CRI invokes Controller**

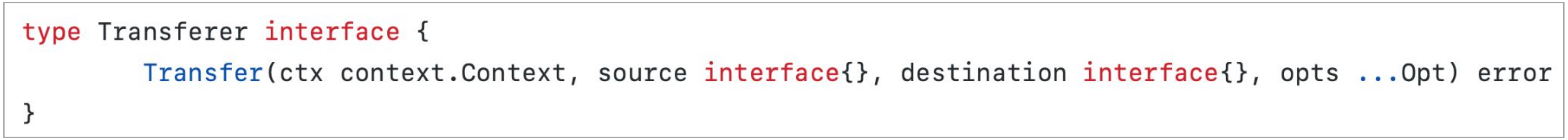
Sandbox API

Ongoing CRI integration:

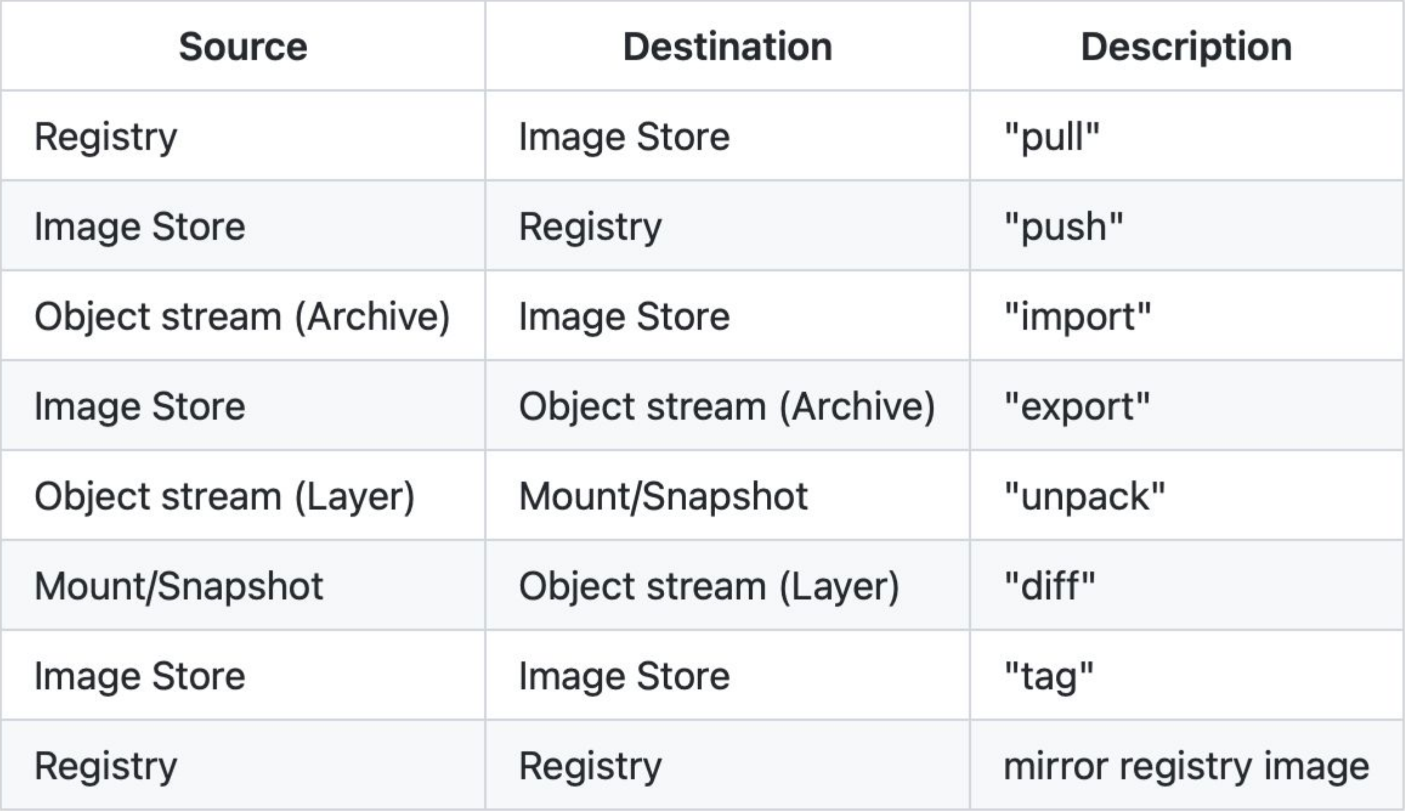
- CRI server fork to enable integration (`sbserver/` directory)
  - Calls Sandbox Controller interface instead of podsandbox
  - Adding **RemoteController** to call shims
- Containerd CRI will default to this implementation in v2.0
- Can currently try it out with `ENABLE_CRI_SANDBOXES` environment variable in v1.7

Transfer service

New API to transfer artifacts from source to destination



```go
type Transferer interface {
        Transfer(ctx context.Context, source interface{}, destination interface{}, opts ...Opt) error
}
```



Source	Destination	Description
Registry	Image Store	"pull"
Image Store	Registry	"push"
Object stream (Archive)	Image Store	"import"
Image Store	Object stream (Archive)	"export"
Object stream (Layer)	Mount/Snapshot	"unpack"
Mount/Snapshot	Object stream (Layer)	"diff"
Image Store	Image Store	"tag"
Registry	Registry	mirror registry image

Transfer service

- **New use cases and extension points**
  - Signing and image validation
  - Credential management
  - Custom pull logic
  - Image decryption
  - Pluggable sources / destinations
- **Sandbox API integration in future**
  - Confidential computing
  - Custom image handling (skip snapshotter)

Kubernetes Interfaces Update

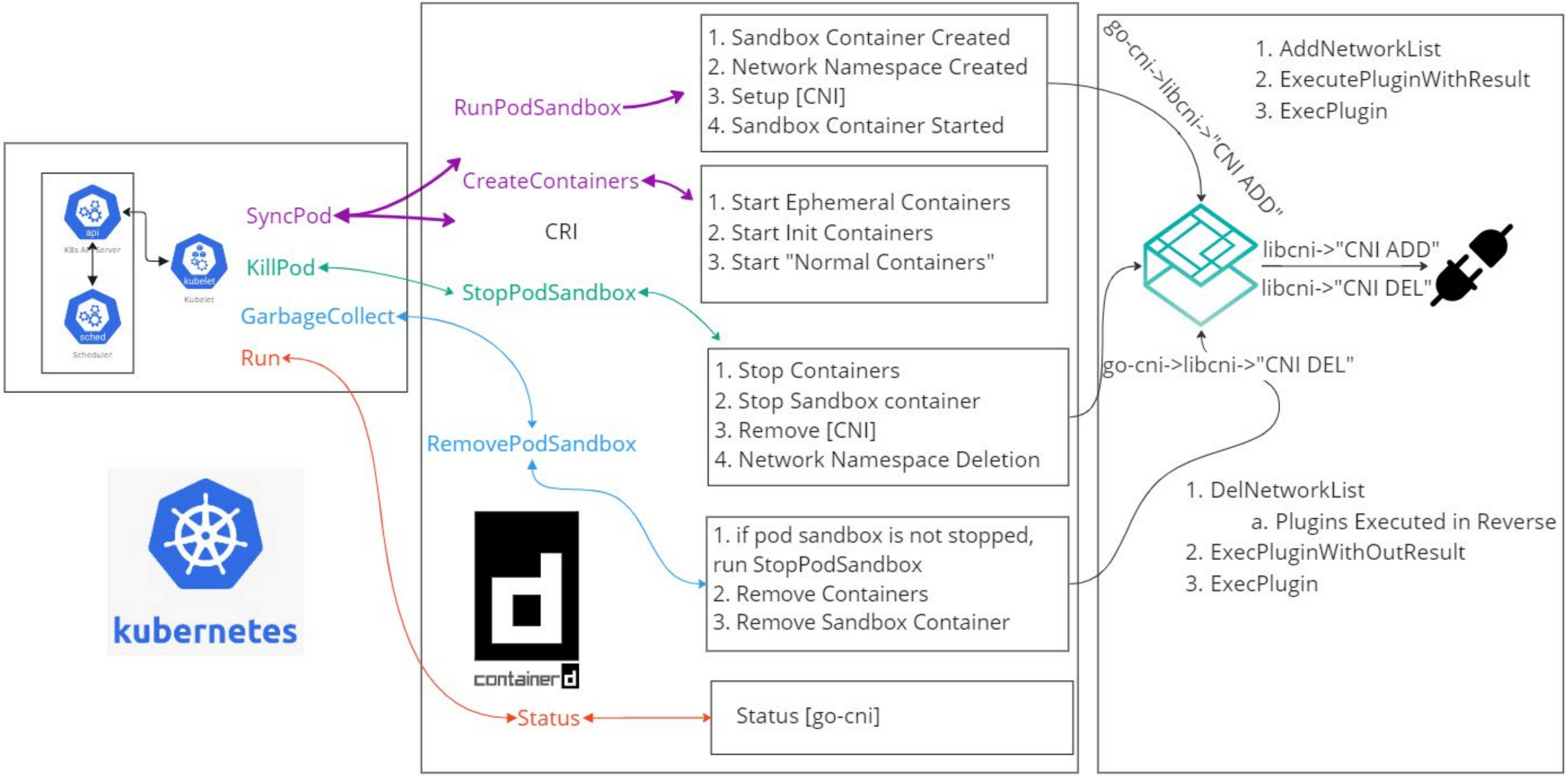
CNI Usage/Changes
CRI New APIs/Updates
NRI Architecture/Plugins

CNI

- **No More DockerShim!**
  - Where are the plugins executed now??
- **CNI Workflow**
  - Official Documentation of CNI/containerd Workflow is in progress!
- **Change of ordering in RunPodSandbox**
  - Improved cleanup on failure
    - IP Address Exhaustion
  - Future work captured!
- **Loopback plugin deprecration Support**
  - You need loopback regardless in current state
- **CNI Maintainer also a contained Maintainer** (Collaboration Time!)
- **Future updates to support CNI 2.0**

- **Network Configuration:**
  - /etc/cni/net.d
  - Configurable [config.toml]
- **CNI Plugins**
  - /opt/cni/bin
  - Configurable [config.toml]

CNI



1. Sandbox Container Created
2. Network Namespace Created
3. Setup [CNI]
4. Sandbox Container Started

1. Start Ephemeral Containers
2. Start Init Containers
3. Start "Normal Containers"

1. Stop Containers
2. Stop Sandbox container
3. Remove [CNI]
4. Network Namespace Deletion

1. if pod sandbox is not stopped, run StopPodSandbox
2. Remove Containers
3. Remove Sandbox Container

Status [go-cni]

1. AddNetworkList
2. ExecutePluginWithResult
3. ExecPlugin

1. DelNetworkList
   a. Plugins Executed in Reverse
2. ExecPluginWithOutResult
3. ExecPlugin

CRI updates

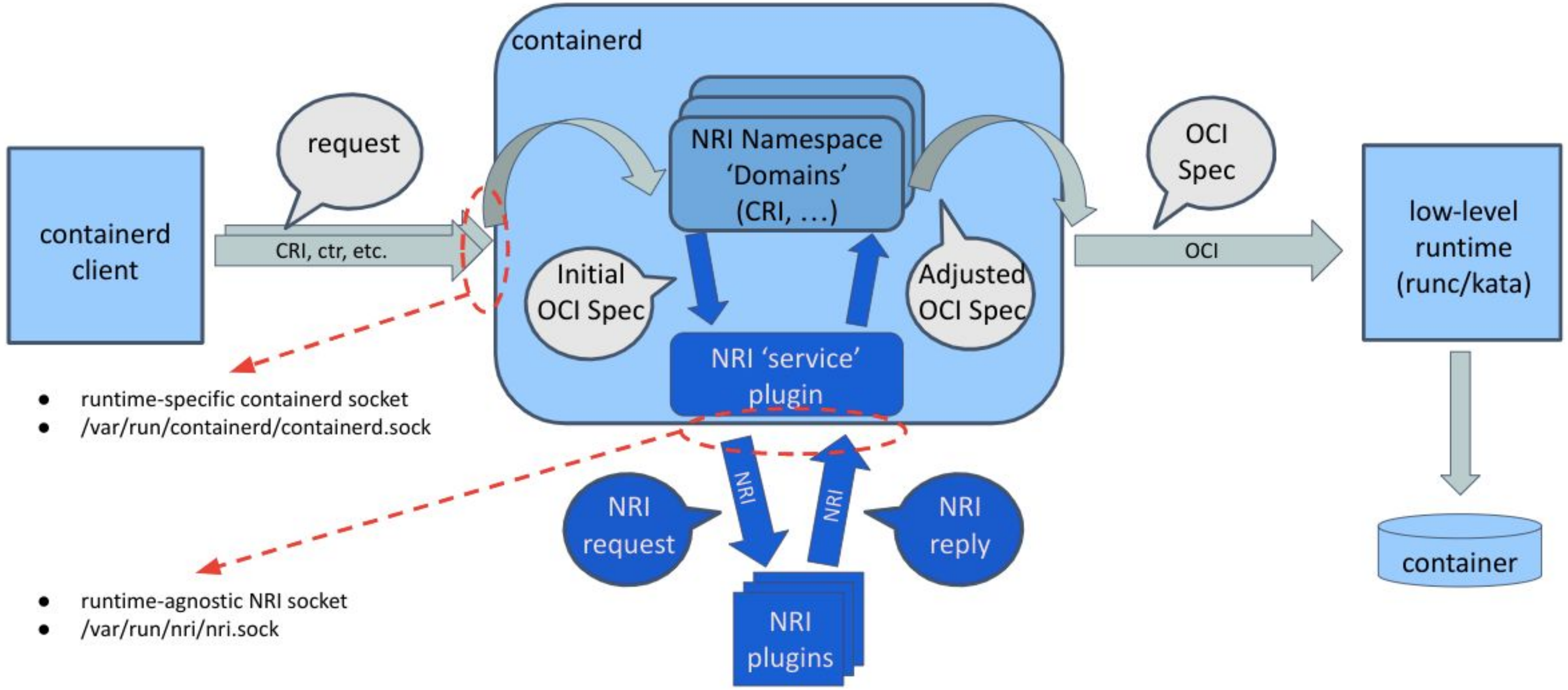
- **The Container Runtime Interface (CRI) is at v1(beta)**
  - New APIs and (non-breaking) API improvements still coming in based on CRI client needs
  - containerd currently vendors "k8s.io/cri-api **v0.25.0**" in **main** and **1.6 LTS** branches
  - API version v1alpha2 is being deprecated/removed from kubelet
    - containerd will keep v1alpha2 around for older clients at least until containerd v1.6 LTS EOL
- **User Namespace support**
  - Based on the user-namespace plan for Kubernetes userns support will appear in phases
  - Phase I contains alpha level support for pods without volumes (stateless pods) as of K8s v1.25
    - containerd issue #7063; needs a PR implementation
- **Checkpoint/restore forensic/debug usage (criu) enablement: ready for testing and feedback**
  - checkpoint started through CRI via root service request to kubelet
  - docs for the "forensic" debug case are a WIP #37412
  - WIP regarding full checkpoint/restore for containers then pods
  - OCI image spec discussion #962
  - Shout out to Adrian Reber who has been driving criu support in the k8s stack for some time
- **Registry configuration** (ongoing discussions)
  - Balancing act between who is in charge of host registry config and authentication
  - Kubelet manages k8s image pull secrets, provider(cloud and other) keychain based authentication
  - Container runtimes own host config, redirect, and default authentications
  - Would like to move from current imperative model to declarative for CRI image services

CRI updates - NRI

Come to our Friday session to learn about the Node Resource Interface (NRI) project, a ttrpc service and framework for plugging in extensions into OCI-compatible container runtimes.

This work enables:

- tracking state changes of containers, pod-sandboxes, and other new sandbox types like micro VMs
- limited changes to container/pod-sandbox configuration
- Shout out to Krisztian Litkey for this chart and all the work he's done on the NRI projects, and to Michael Crosby for his initial NRI prototype.



- runtime-specific containerd socket
- /var/run/containerd/containerd.sock

- runtime-agnostic NRI socket
- /var/run/nri/nri.sock

Runtimes & OCI - Artifact Support

- The OCI focus for some time now has been adding support for new Artifact types and References for linking the new Artifact Manifests to other Artifact/Container Image Manifests.
    a. Big Shout-Out to the OCI reference-types WG for being the first official/completed OCI WG
- The artifact/OCI references work may have integration points with containerd image resolving/fetching in future releases; specifics TBD.
- OCI/container community is working through registry and client implementation updates to the new specs.

Please scan the QR Code above to leave feedback on this session

KubeCon | CloudNativeCon

North America 2022

BUILDING FOR THE ROAD AHEAD

DETROIT 2022