


KubeCon | CloudNativeCon
North America 2024

What's Going On in the **containerd** Neighborhood?

Phil Estes, Principal Engineer, AWS
Mike Brown, Software Engineer/Architect, IBM
Samuel Karp, Staff Software Engineer, Google
Akihiro Suda, Software Engineer, NTT
Kirtana Ashok, Software Engineer, Microsoft

A conversation among maintainers

Who we are

- **Phil Estes**, Principal Engineer, AWS (moderator)
- **Mike Brown**, Software Engineer/Architect, IBM
- **Samuel Karp**, Staff Software Engineer, Google
- **Akihiro Suda**, Software Engineer, NTT
- **Kirtana Ashok**, Software Engineer, Microsoft

Why does containerd exist?

What value does it bring to the overall cloud native ecosystem?

How does the containerd project relate to Kubernetes?

How are other projects using/extending containerd in useful ways?

Runtime plugins

- **runc**: the regular runtime for Linux
- **runhcs**: Windows
- **runj**: FreeBSD jail
- **runwasi**: WASM
- **kata**: VM
- **runsc (gvisor)**: ptrace sandbox, etc.

Snapshotter plugins

Regular snapshotters: **overlayfs**, **btrfs**, **zfs**, **devmapper**, ...

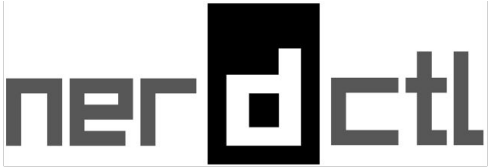
“Remote” snapshotters support pulling image contents on demand to shorten the container startup time

- **stargz**: Forward compatible with OCI v1 tar.gz images
- **nydus**: Uses an alternate image format
- **overlaybd**: Uses block devices as container images

Case study: nerdctl

nerdctl: contaiNERD CTL

- https://github.com/containerd/nerdctl
- Same UI/UX as the docker CLI (including Compose)
- Made for facilitating new experiments in the containerd platform (e.g., stargz, fast rootless with bypass4netns)
- Useful for debugging Kubernetes nodes too



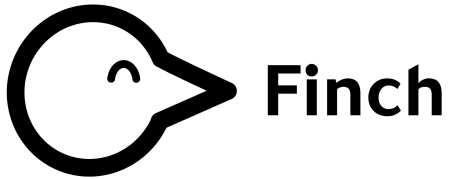

```
$ nerdctl run hello-world
$ nerdctl compose up
```

Lima & Finch

- Lima (https://lima-vm.io): “Linux Machine”
  - CNCF Sandbox Project
  - Similar to Docker Machine, but uses nerdctl as the default container engine


```
$ brew install lima
$ limactl start
$ nerdctl.lima run -p 80:80 nginx
```

- Finch (https://runfinch.com) : AWS’s container engine based on Lima and nerdctl



How did you get involved in containerd?

Why did we move to 2.0?

New features, newly stable features, defaults

- Transfer service NOW STABLE
- Sandbox service (and sandboxed CRI) NOW STABLE
- Faster image extraction with igzip NEW
- Improved OTEL configuration NEW
- NRI enabled by default NEW
- Image verifier plugins NEW
- Plugin introspection NEW
- CDI enabled by default NEW
- CRI support for user namespaces NEW

Highlight: Node Resource Interface (NRI)

- Akin to a mutating webhook, but for container configuration
  - Middleware between CRI and OCI
- Use cases
  - Injection (devices, network devices, OCI hooks)
  - Resource modification/management (ulimits, topology/NUMA, advanced QoS, SGX memory)
  - Policy enforcement
- Plugins can run in containers or as system services
- Enabled by default
- Community plugins
  - https://github.com/containerd/nri/tree/main/plugins
  - https://github.com/containers/nri-plugins

Highlight: Image verifier plugins

- Exec-based plugins containerd invokes during image pull
- Policy enforcement use-cases
  - Container image signature verification
  - Trust for particular signers
  - Allow only specific registries/repositories
- Integrated with the Transfer service (not supported for legacy pulls)

nerdctl v2.0

- Rootless networking is refactored to support “detach-netns”
  - Faster pull/push
    - Previously limited to less than 10Gbps due to user-mode TCP/IP
    - Now as fast as rootful
    - Containers can be accelerated too with bypass4netns (experimental)
  - Proper support for --net=host
  - Proper support for localhost registries

nerdctl v2.0

- Support running systemd in a container without --privileged
  - "Plain old VM"-like user experience
  - Often considered to be an anti-pattern
  - Useful for testing, etc.

```
$ nerdctl run --systemd=true ...
```

- Massive refactoring and testing

What does LTS mean for containerd?

How does a KEP get implemented in containerd?

KEP - Kubernetes Enhancement Process

- The KEP process usually starts with interested party discussions
  - SIG-Node or containerd slack channels..
  - sometimes in a container runtime community meeting
  - containerd issues/discussions tabs on the repo
- On agreement to move forward with a KEP, a discussion is added to one of the weekly Sig community calls. Usually SIG-Node sometimes we involve SIG-Auth/Docs/Test/Infra/Storage/....
- With consensus from Sig contributors/leadership - an issue is opened in github.com/kubernetes/enhancements
- The KEP process is well defined and they will help you out along the way
- Additionally, initial volunteers are sought out for the various roles, sometimes volunteers are known before the call sometimes that happens on the call, but we've always been inclusive..
- thus begins the Kubernetes KEP Process

KEP - Kubernetes Enhancement Process

- In the CDI device project case,
  - discussions were held at kubecon and the slack channels to form a WG between various device owners and runtimes.. CNCF WG for Container Orchestrated Devices … CDI Repo
- Here is an example of a CDI Device Update KEP for the CRI API
  - KEP tracking issue for adding a new field to the CRI API passing a list of CDI devices to inject in the container https://github.com/kubernetes/kubernetes/issues/114209 completed
  - Actual Approved KEP design details:
  - https://github.com/kubernetes/enhancements/tree/master/keps/sig-node/4009-add-cdi-devices-to-device-plugin-api approved
  - PRs are drafted, reviewed, tested, merged..
    https://github.com/kubernetes/kubernetes/pull/115891 merged
    https://github.com/containerd/containerd/pull/8252 merged
- Note: Prior to the “official” way of passing CDI Devices we did proof of concepts using annotations..

OCI Image Volume - KEP

OCI Image Volumes

- KEP Stage -Alpha:

https://kubernetes.io/blog/2024/08/16/kubernetes-1-31-image-volume-source/

- WIP - PR scheduled for the next point release of containerd

https://github.com/containerd/containerd/pull/10579

- Will also be adding support for restricting contents to a subpath of the image volume.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod
spec:
  containers:
    - name: test
      image: registry.k8s.io/e2e-test-images/echoserver:2.3
      volumeMounts:
        - name: volume
          mountPath: /volume
  volumes:
    - name: volume
      image:
        reference: quay.io/crio/artifact:v1
        pullPolicy: IfNotPresent
```

Follow the QR code to leave session feedback!


KubeCon
CloudNativeCon
North America 2024