KubeCon | CloudNativeCon

Europe 2021

Virtual

Introduction and Deep Dive Into Containerd

Kohei Tokunaga & Akihiro Suda, NTT Corporation

Introduction to containerd

Kohei Tokunaga, NTT Corporation

Overview

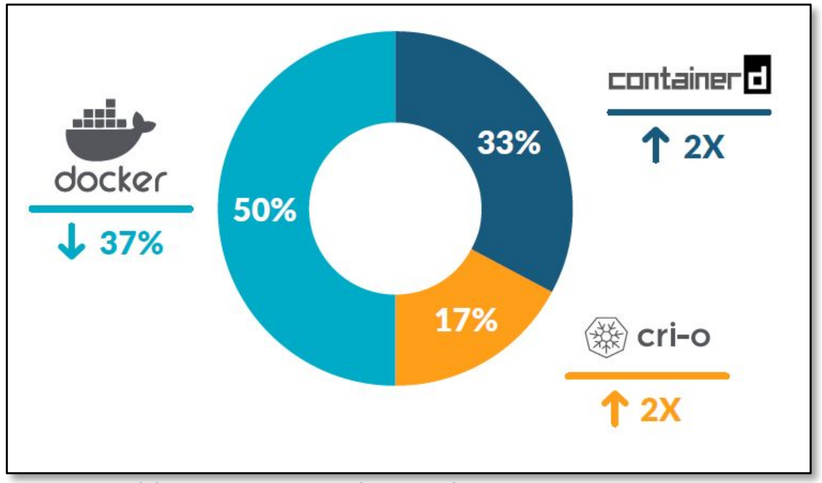
https://github.com/containerd/containerd

- CNCF graduated container runtime project
- Resource manager
  - Container process
  - Image artifacts
  - Filesystem snapshots
  - Metadata and dependencies management
- Tightly scoped (100% approval is required to stretch) but highly extensible
- Used by Kubernetes, Docker and various container-based projects

Usage in community

- Docker's use of containerd + pure use of containerd is 83% of container usage (Sysdig 2021 container security and usage report)

- Used by several managed services as well as open source projects in community

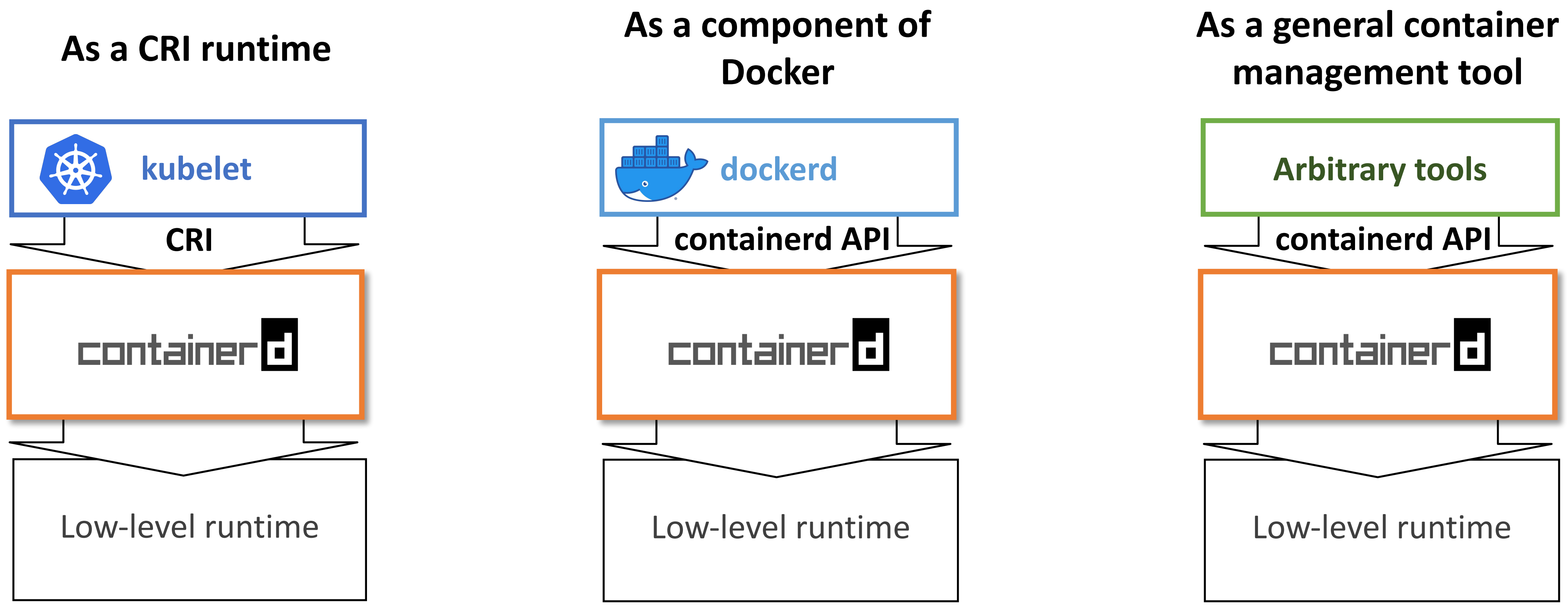


https://sysdig.com/blog/sysdig-2021-container-security-usage-report/

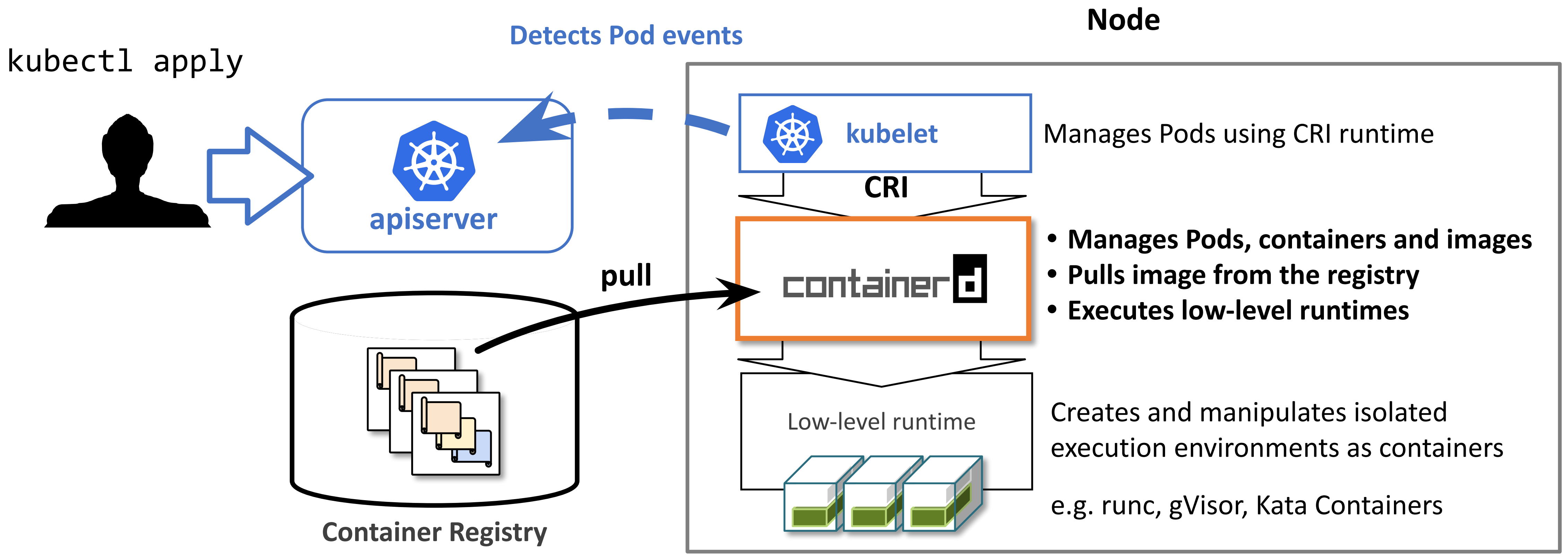
Adoption

- Managed: GKE, AWS Fargate, AKS, IKS
- Development: Docker/moby, BuildKit
- K8s distribution: k3s, kind、minikube, kubespray, microk8s, k0s
- FaaS: faasd

How containerd is used?



Containerd as a CRI runtime

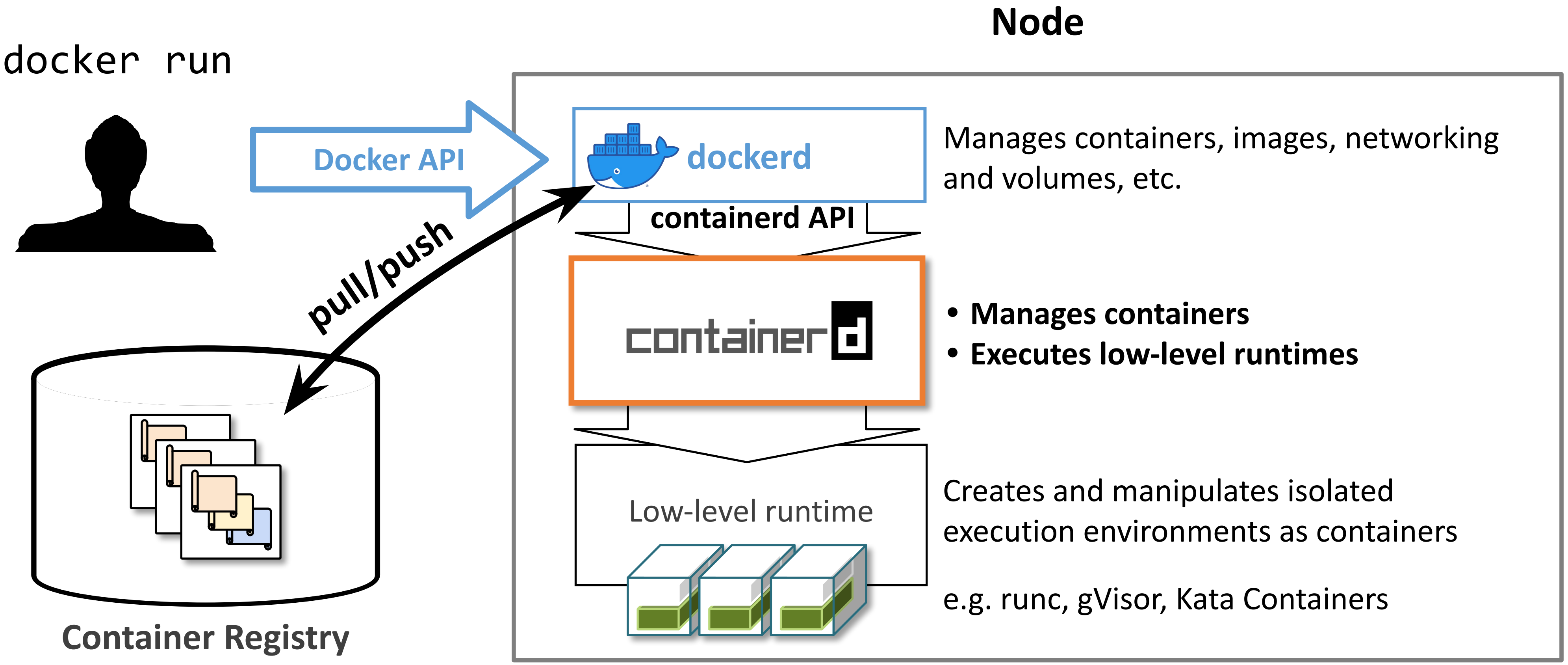


Node
Detects Pod events
kubectl apply
apiserver
kubelet
Manages Pods using CRI runtime
CRI
containerd
pull
• Manages Pods, containers and images
• Pulls image from the registry
• Executes low-level runtimes
Low-level runtime
Creates and manipulates isolated execution environments as containers
e.g. runc, gVisor, Kata Containers
Container Registry

The de facto standard CRI runtime for Kubernetes

- Managed Kubernetes: IKS, GKE, AKS, AWS Fargate, …
- Kubernetes distributions: K3s, kind, minikube, kubespray, microk8s, k0s, ...

Containerd as a component of Docker



Manages containers, images, networking and volumes, etc.

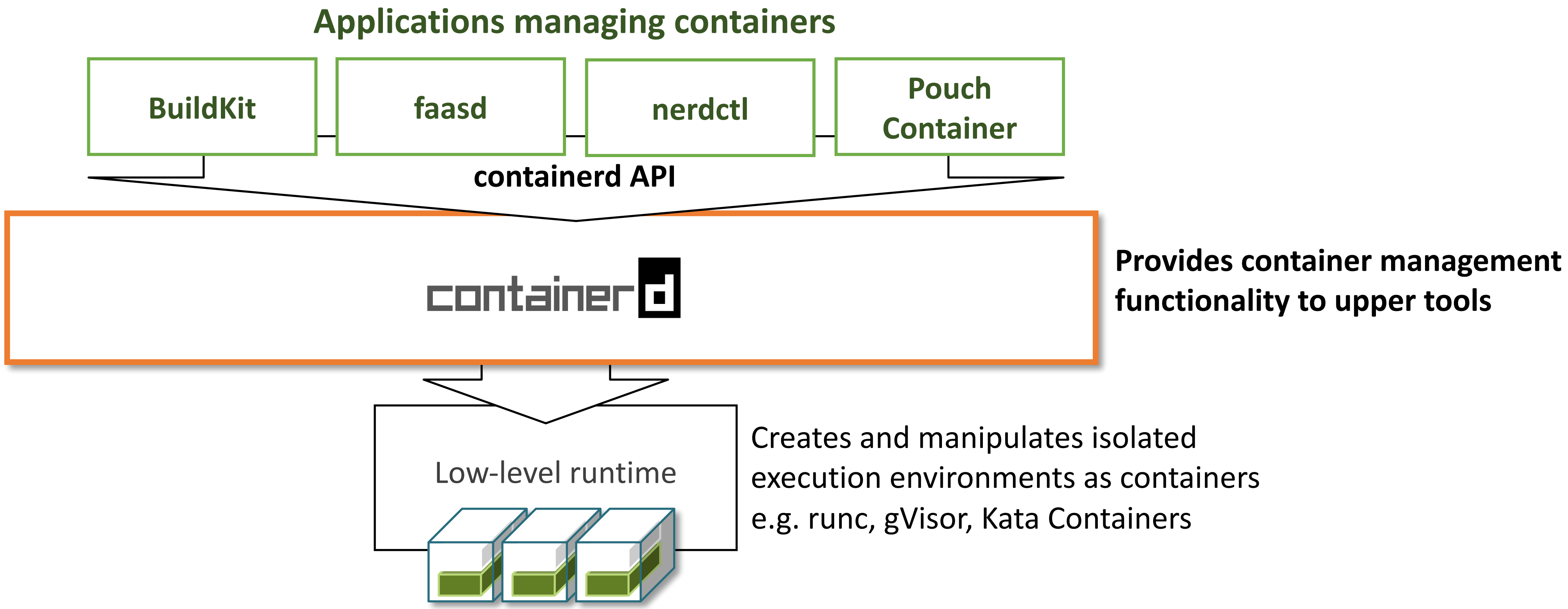
- **Manages containers**
- **Executes low-level runtimes**

Creates and manipulates isolated execution environments as containers

e.g. runc, gVisor, Kata Containers

Containerd as a general container management tool

- Several applications are developed based on containerd
- Containerd provides a Go client library (discussed later)
- Applications can extend containerd with plugins, *without recompilation* (discussed later)

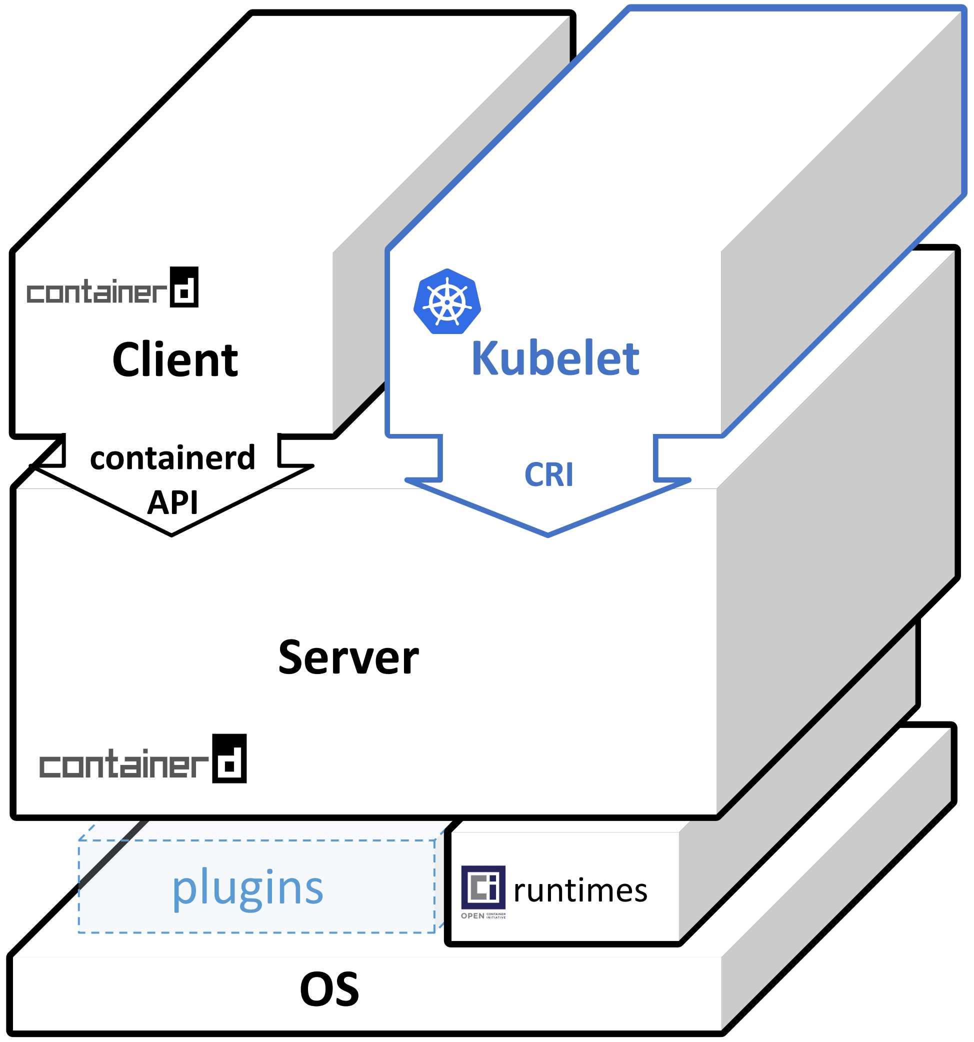

Applications managing containers
BuildKit
faasd
nerdctl
Pouch Container
containerd API
containerd
Provides container management functionality to upper tools
Low-level runtime
Creates and manipulates isolated execution environments as containers
e.g. runc, gVisor, Kata Containers

Containerd Internal

Kohei Tokunaga, NTT Corporation

Containerd Architecture

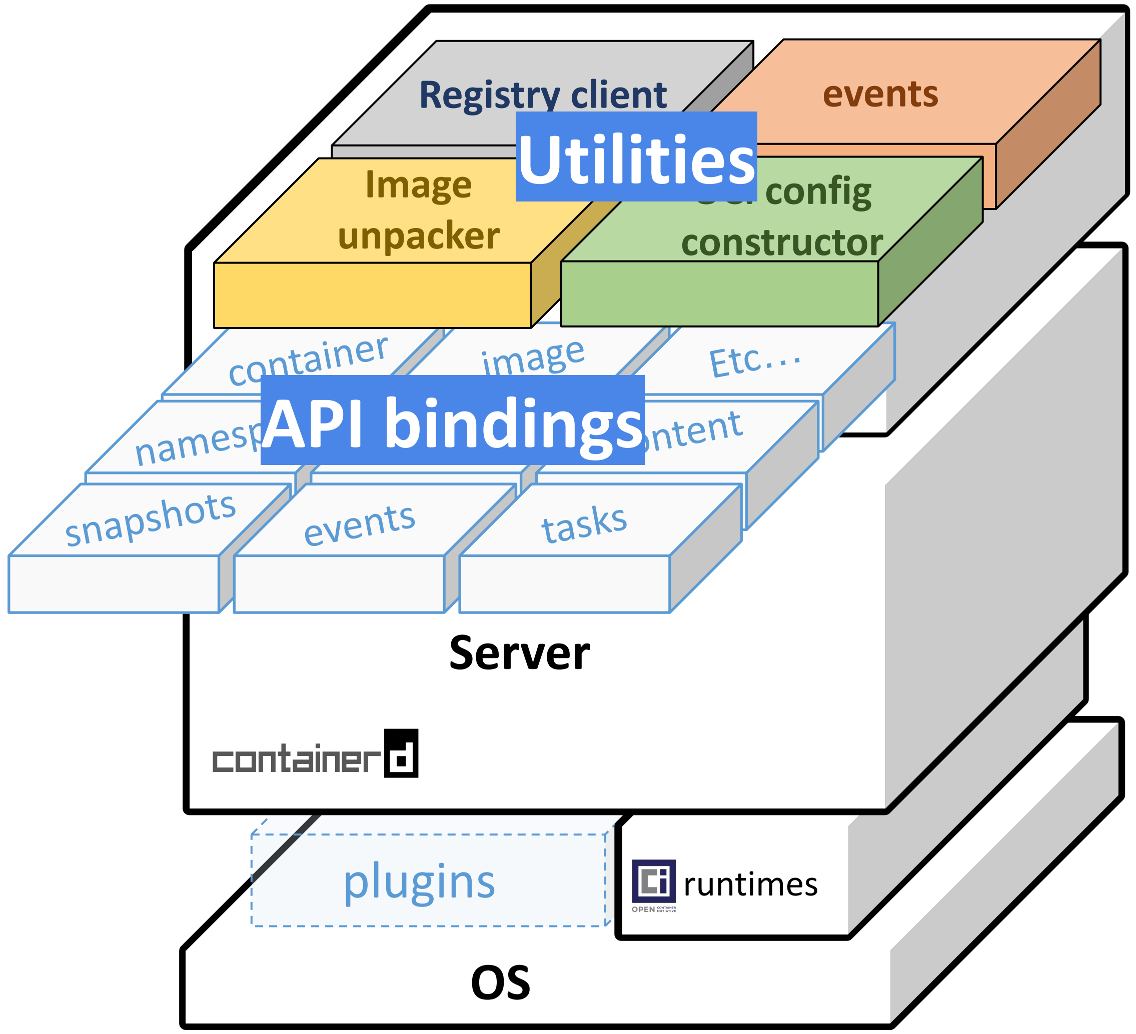
- Client-server architecture
  - Go client library (used by Docker, BuildKit, etc.)
- Client calls server via **containerd API**
  - Through /run/containerd/containerd.sock
- Various low-level runtimes are supported
  - OCI runtimes (runc, gVisor, Kata Container, etc)
  - Firecracker (firecracker-containerd)
- Extensibility
  - Low-level plugins
  - Extending containerd API with custom services
  - Client library is easy to customize



Containerd Client

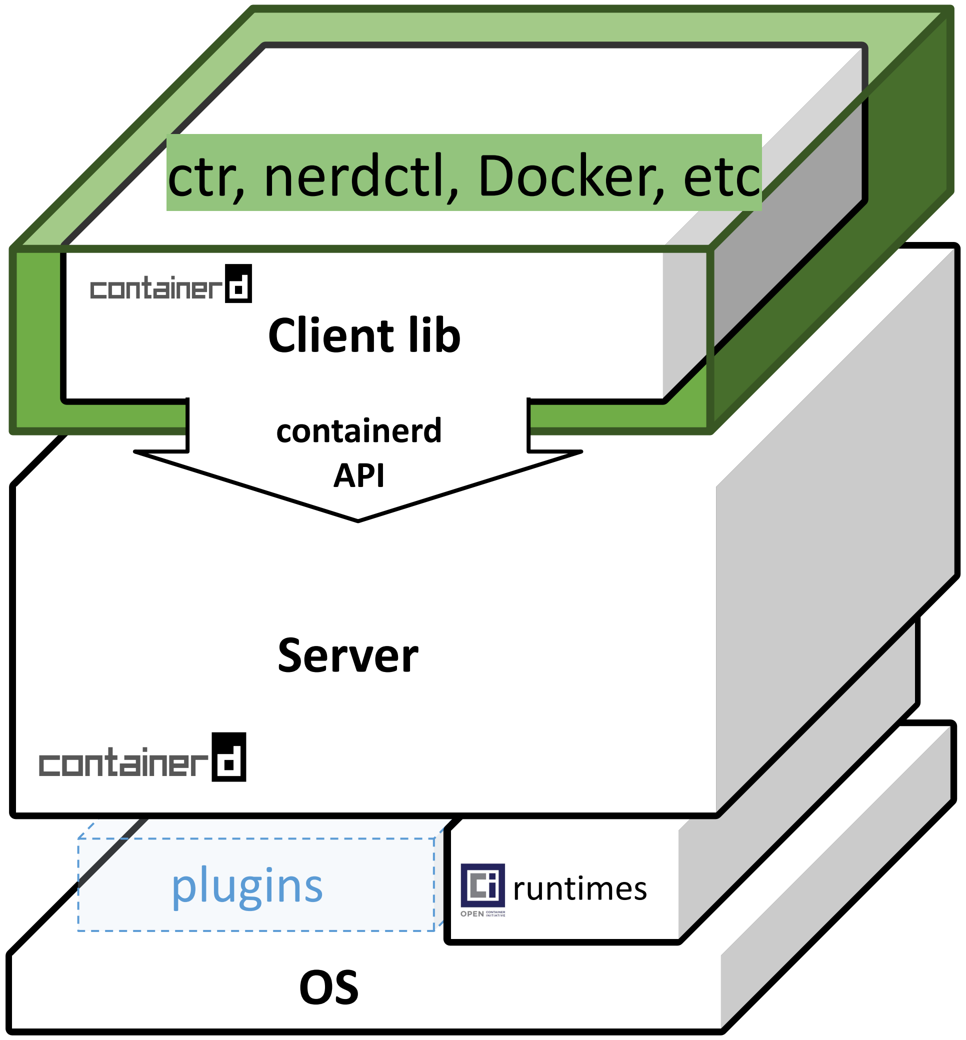
- “Smart” Client (Go library)
  - Containerd API bindings
  - Registry client
  - Pulling/Pushing images
  - Image unpacker
  - Creating OCI config for OCI runtimes

- Go application can integrate with containerd using client library



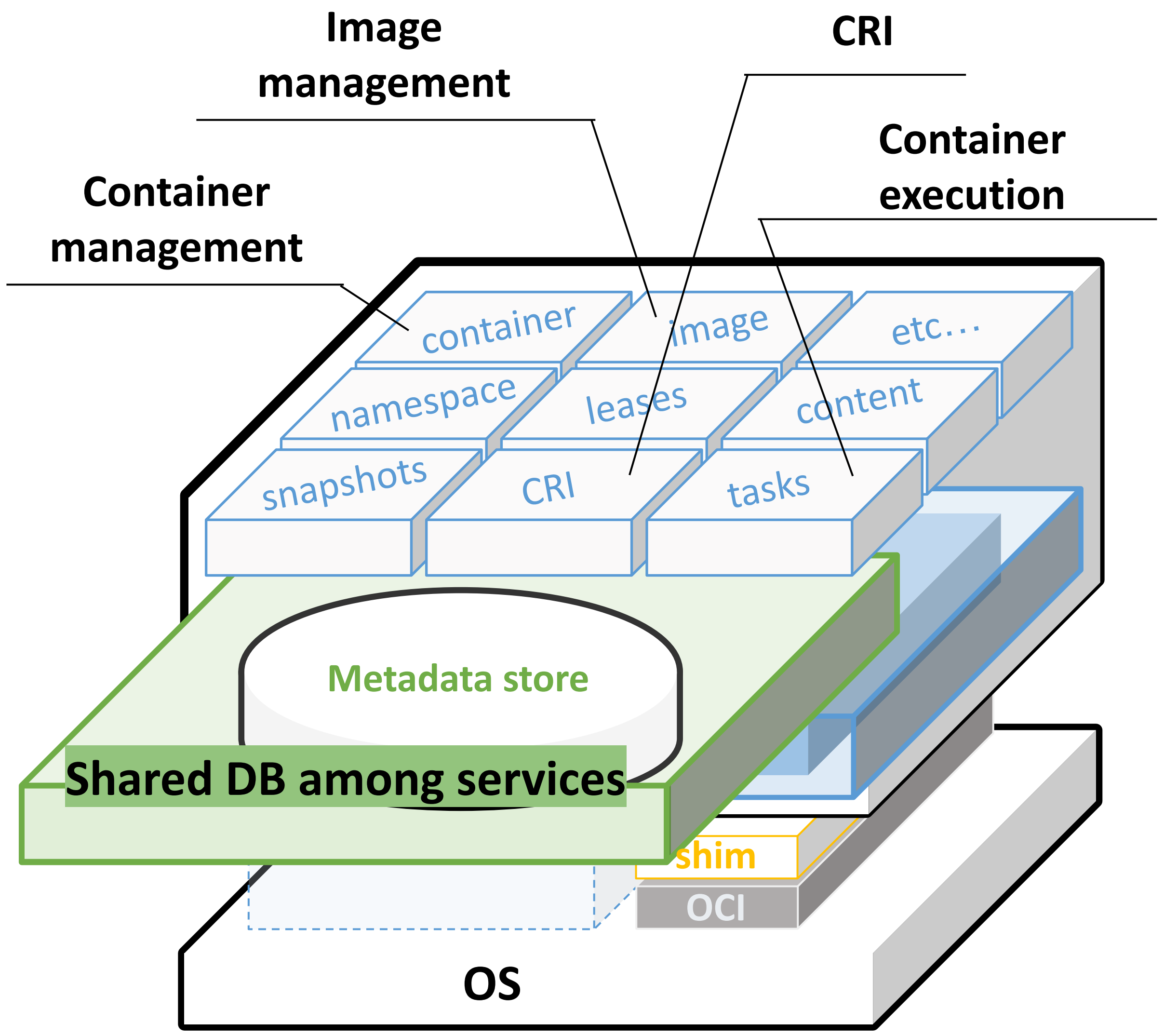
Containerd Client Implementations

- **ctr**: https://github.com/containerd/containerd
  - CLI client for containerd
  - Mainly for debugging or trying new features
- **nerdctl**: https://github.com/containerd/nerdctl
  - Docker-compatible CLI for containerd
  - Easy to use for Docker users
  - Supports containerd's cutting-edge features (e.g. lazy pulling, image encryption)
- **containerd-based tools**
  - Arbitrary tools can integrate to containerd using client library
  - e.g. Docker, BuildKit, faasd



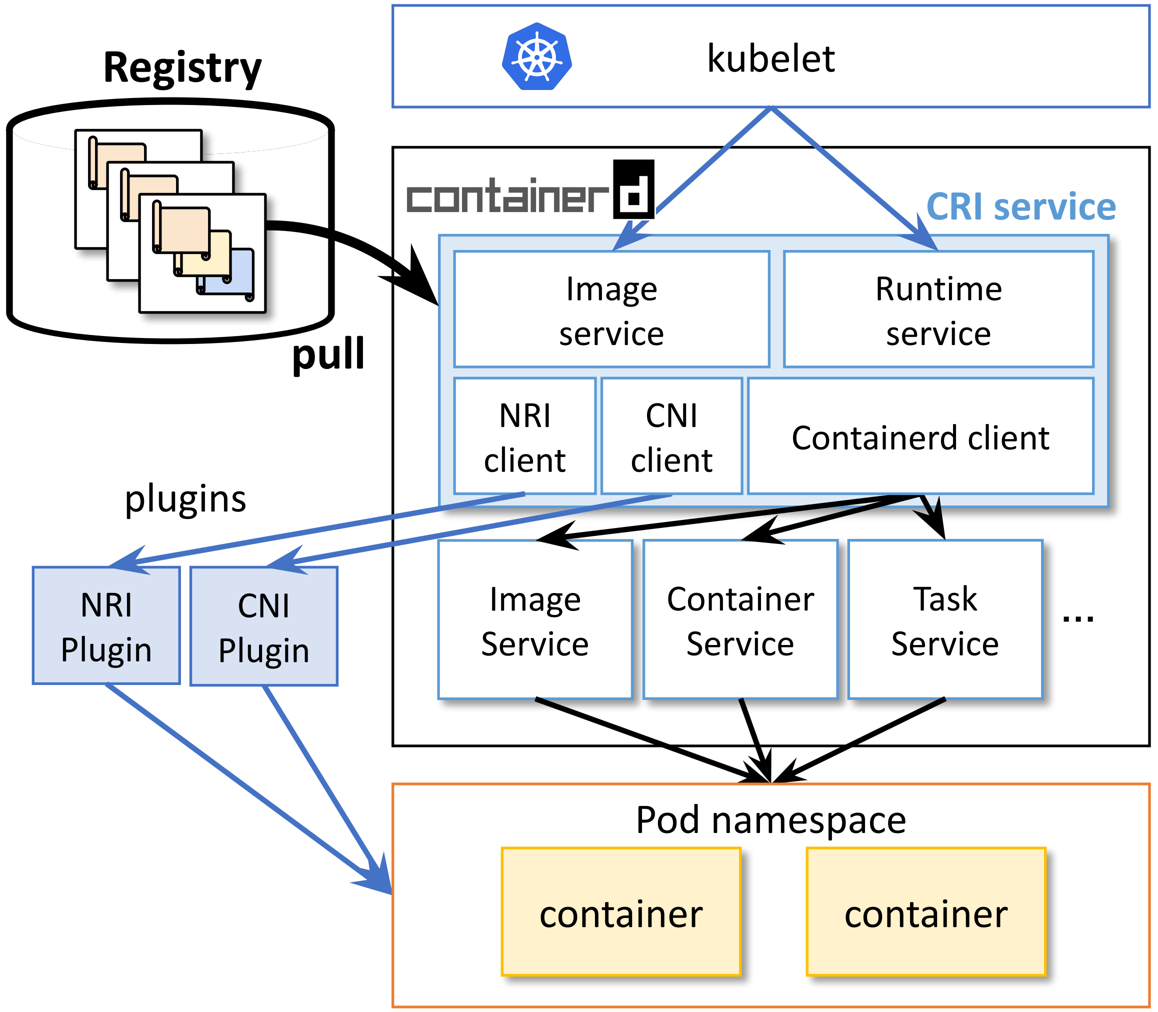
Containerd Core & API

- Micro services
  - Containerd API is the set of APIs of services
  - Services are loosely connected
- Shared metadata DB
  - bbolt-based
    - https://github.com/etcd-io/bbolt
  - Stores metadata of containers, images, contents, snapshots, etc.
  - Manages reference graph for GC



CRI Service

- CRI service implements CRI of Kubernetes
- Implemented as a builtin service
  - Initially repo and binary were separated from containerd
  - Merged to containerd/containerd since 1.5
- Depends on other services for container & image management
  - Communicates via function call
- Uses external CNI/NRI plugins for networking and resource management



Low-level Services

- **Content Store**
  - Stores image manifest and layers “as-is”
  - content addressable (keyed by digest)
- **Snapshotter**
  - Manages ”snapshots”
    - Extracted and stacked view of rootfs layers
  - Passed to OCI runtimes as rootfs
  - Snapshotter impl. per backing filesystem
    - Overlayfs, btrfs, aufs, FUSE, …
- **Runtime**
  - Executes low-level runtimes via “shim”
  - Shim is a wrapper daemon of OCI runtime
  - Well-suit to stateful runtimes (e.g. Kata Containers)

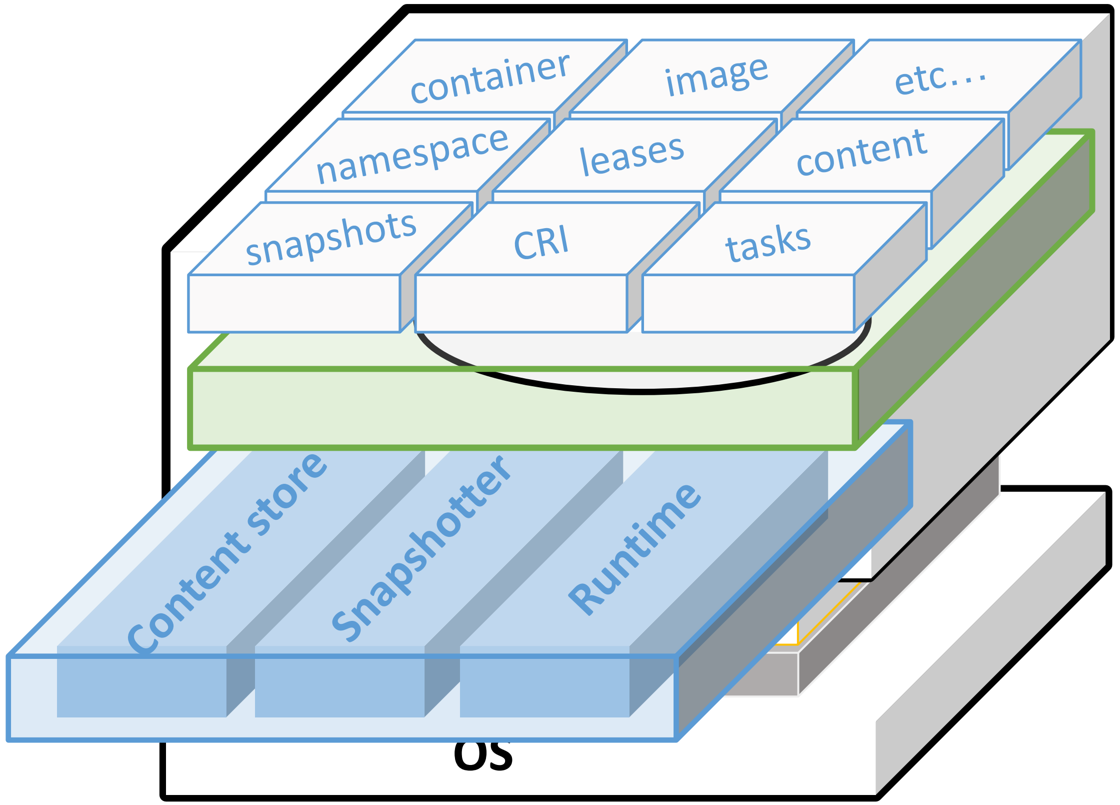
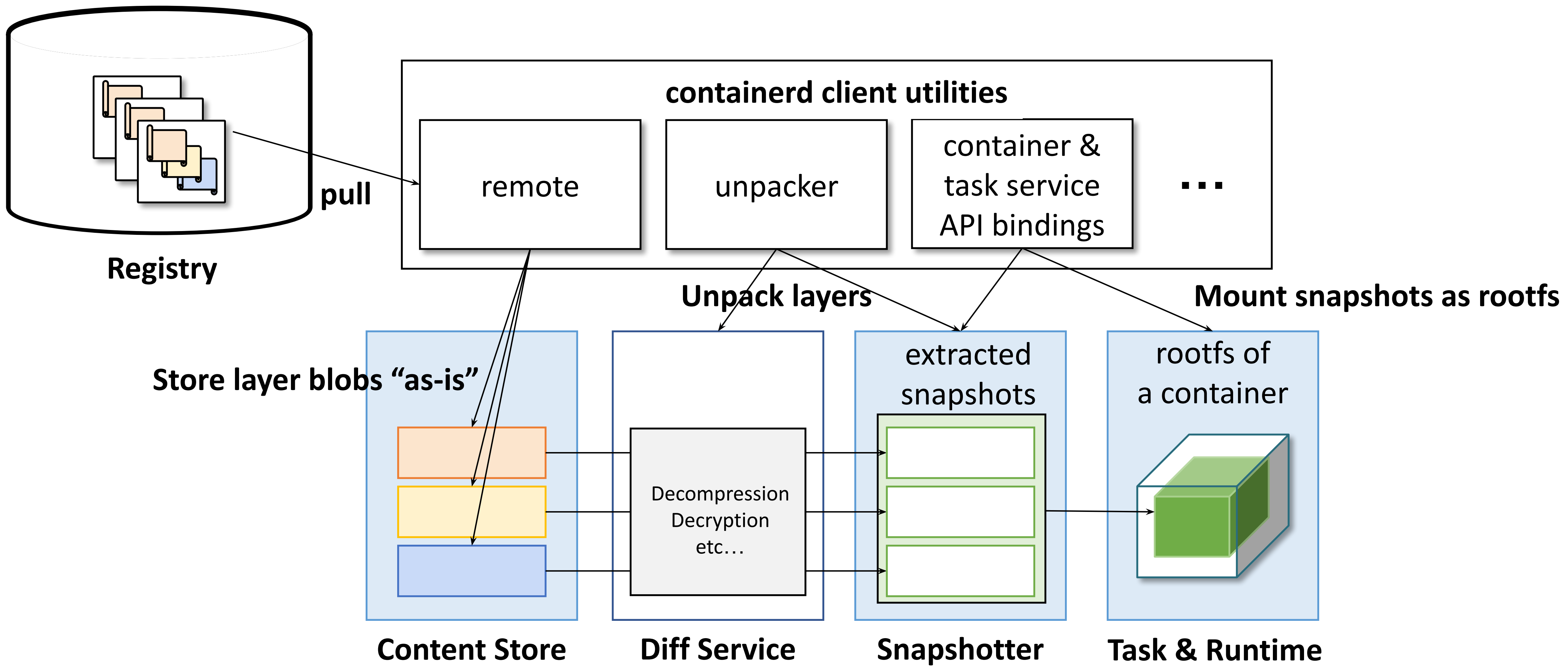


Image content flow

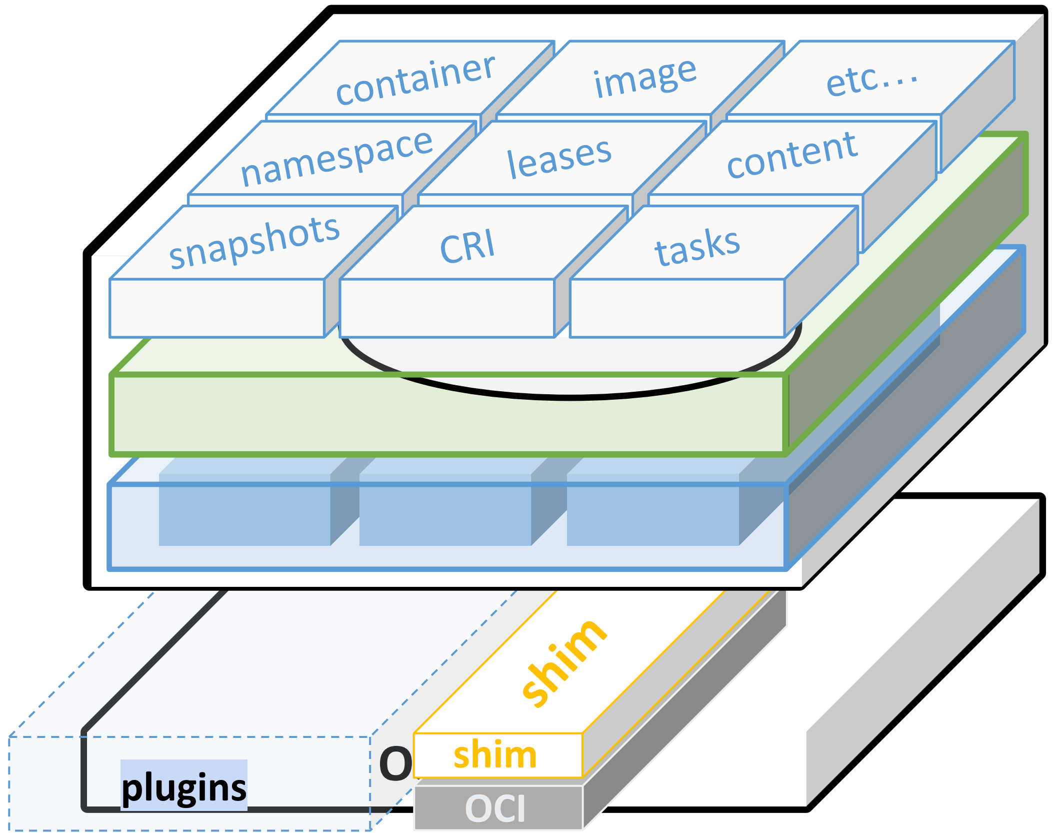


Containerd Extensibility

Kohei Tokunaga, NTT Corporation

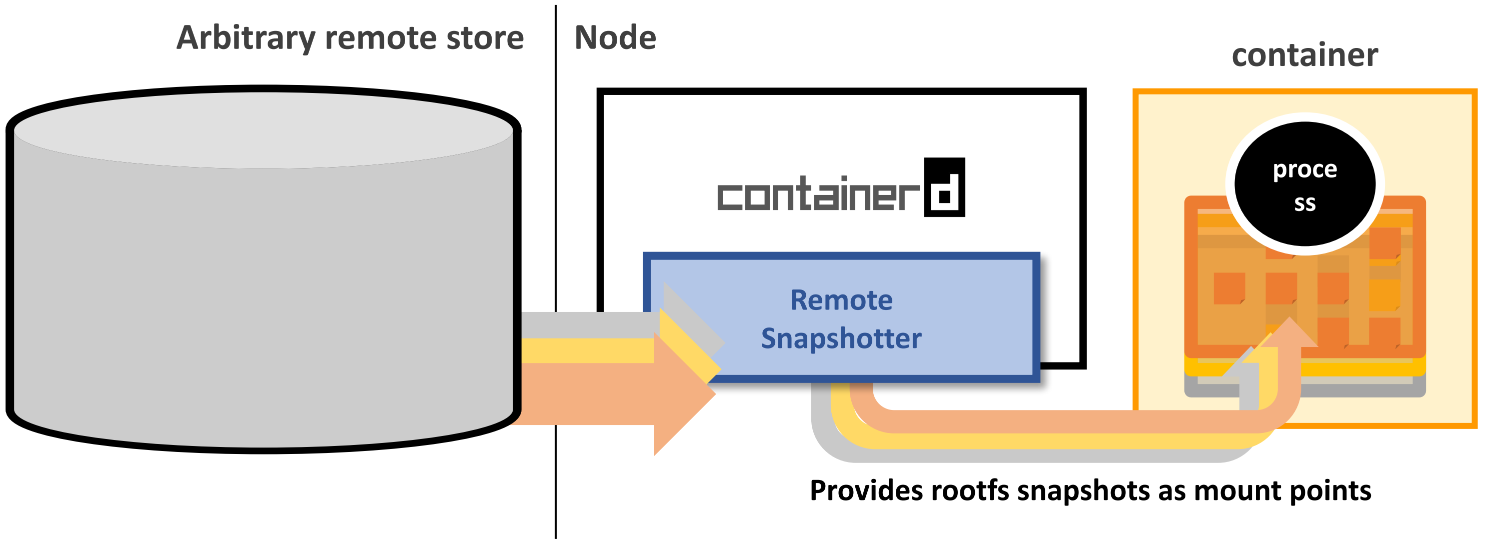
Extending containerd with plugins and services

- containerd is tightly scoped but highly extensible
- Custom low-level service; **no need to recompile**
  - external binary plugins
    - Plugin via unix socket (proxy snapshotter, proxy content store)
    - Plugin as an executable binary (stream processor, shim)
  - Go plugin
- API is extendable by implementing your own custom service
  - e.g. ”control API” of firecracker-containerd



Extension example 1: Lazy pulling

- **Remote snapshotter plugin**
  - allows “lazy pulling” of images from arbitrary remote store (not limited to the registry)
    - container can startup without waiting for the entire image contents being locally available
- Snapshotter can run as an external daemon (proxy snapshotter)
  - No re-compilation is required
  - Containerd talks with the snapshotter via unix socket
- **Stargz Snapshotter** enables lazy pulling of OCI-compatible eStargz/Stargz images from standard registry
  - https://github.com/containerd/stargz-snapshotter

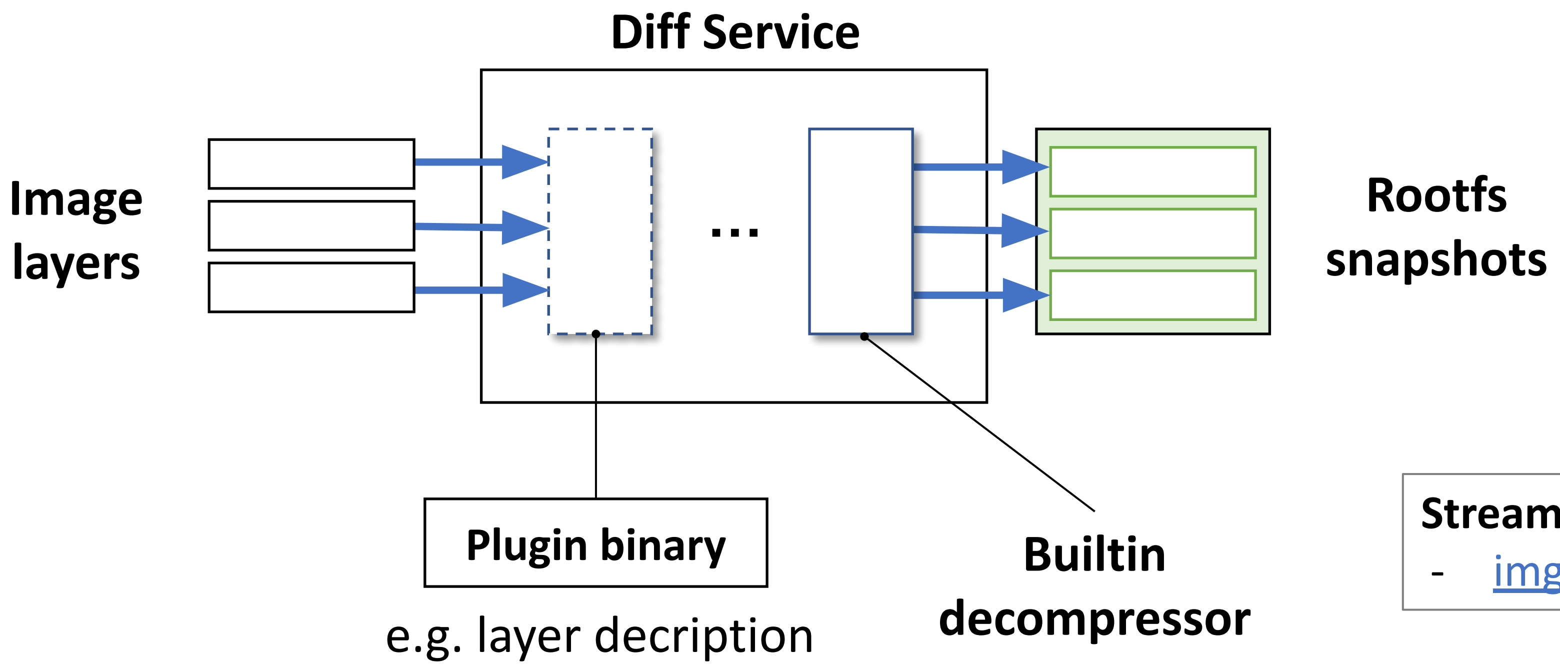


Remote Snapshotters in community

- Stargz Snapshotter
- CVMFS-snapshotter
- Nydus-snapshotter
- OverlayBD-snapshotter

Extension example 2: Generic image layers

- Containerd can handle arbitrary image layers, not limited to OCI standards
  - gzip, zstd, encrypted layers…
- Stream Processor plugin converts arbitrary media type to another (e.g. OCI standard types)
- Separated binary can plug into containerd, without re-compilation

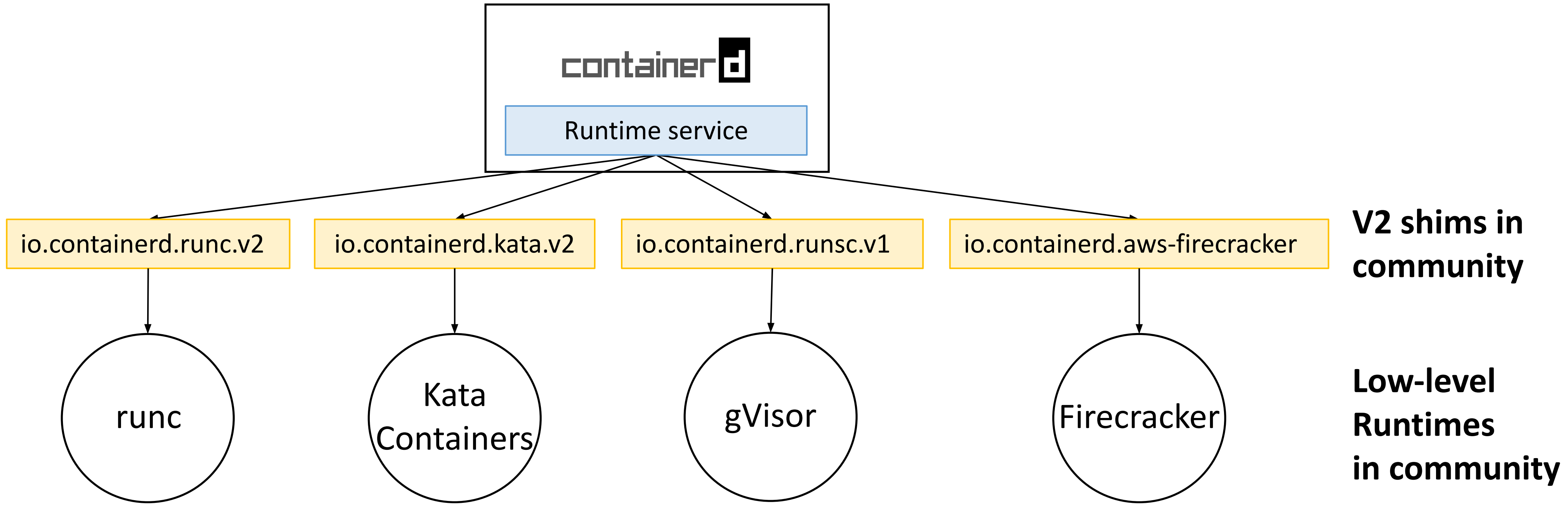


Stream Processor in community

- imgcrypt for encrypted images

Extension example 3: Integrating low-level runtimes

- V2 Shim per low-level runtime
- Both of OCI (e.g. runc) and Non-OCI (e.g. Firecracker) runtime can integrate to containerd
- Binary naming convention: io.containerd.runc.v2 -> containerd-shim-runc-v2
- Pluggable logging destination
  - fifo(Linux), npipe(Windows), external binary(Linux, Windows), file(Linux, Windows)



Implementing your own containerd client

Akihiro Suda, NTT Corporation

Implementing your own containerd client

Two APIs are available

	containerd API	CRI API
Consumers	Docker/Moby, BuildKit, faasd, nerdctl...	Kubernetes
Paradigm	Container-oriented	Pod-oriented
Flexibility	Good	Bad
Simplicity	Bad	Good
Transportation	gRPC over UNIX socket	gRPC over UNIX socket

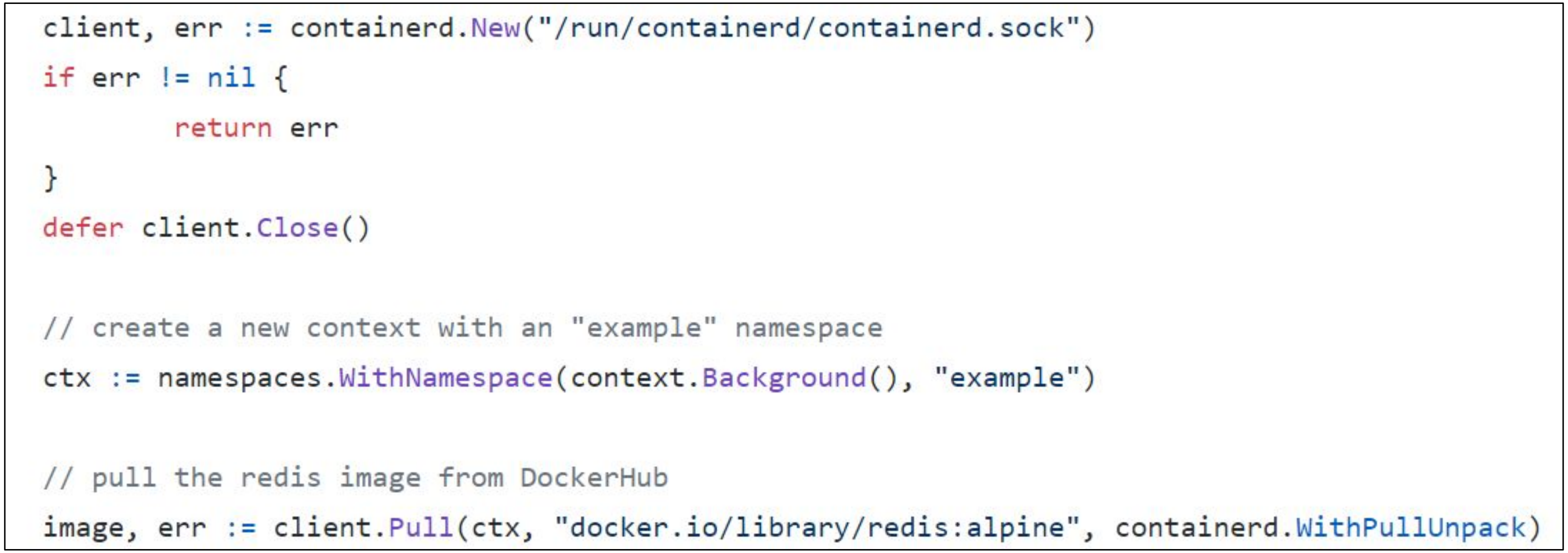
containerd API is recommended for most use cases, but CRI API might be easier to get started

Implementing your own containerd client

- Both containerd API and CRI API use gRPC
- In theory you could use any language for your own client
- But containerd API depends on “smart client” written in Go, especially for pulling images
- So, currently, Go is the best language for Native API
- Contribution is wanted for other languages

Implementing your own containerd client

Example: https://containerd.io/docs/getting-started/



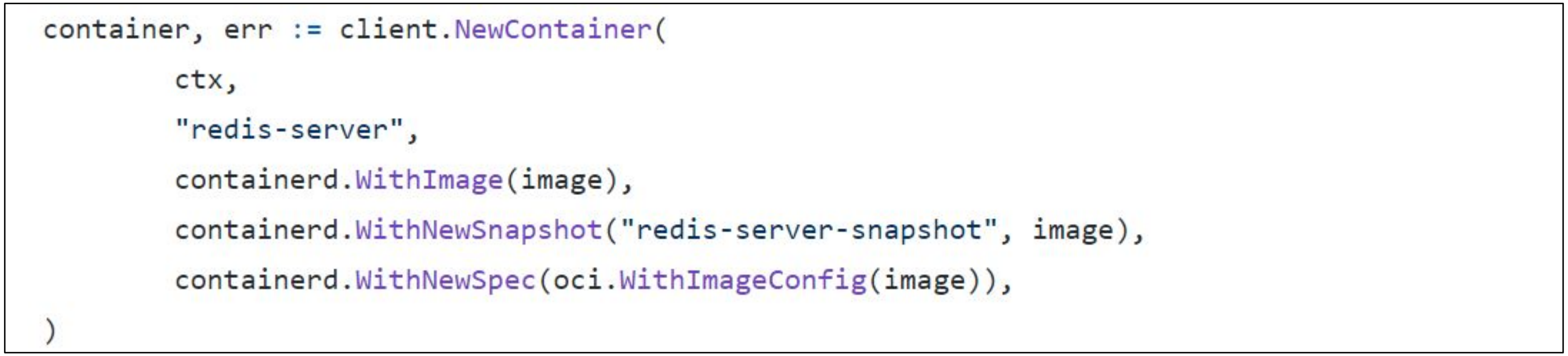
```go
client, err := containerd.New("/run/containerd/containerd.sock")
if err != nil {
        return err
}
defer client.Close()

// create a new context with an "example" namespace
ctx := namespaces.WithNamespace(context.Background(), "example")

// pull the redis image from DockerHub
image, err := client.Pull(ctx, "docker.io/library/redis:alpine", containerd.WithPullUnpack)
```

Implementing your own containerd client

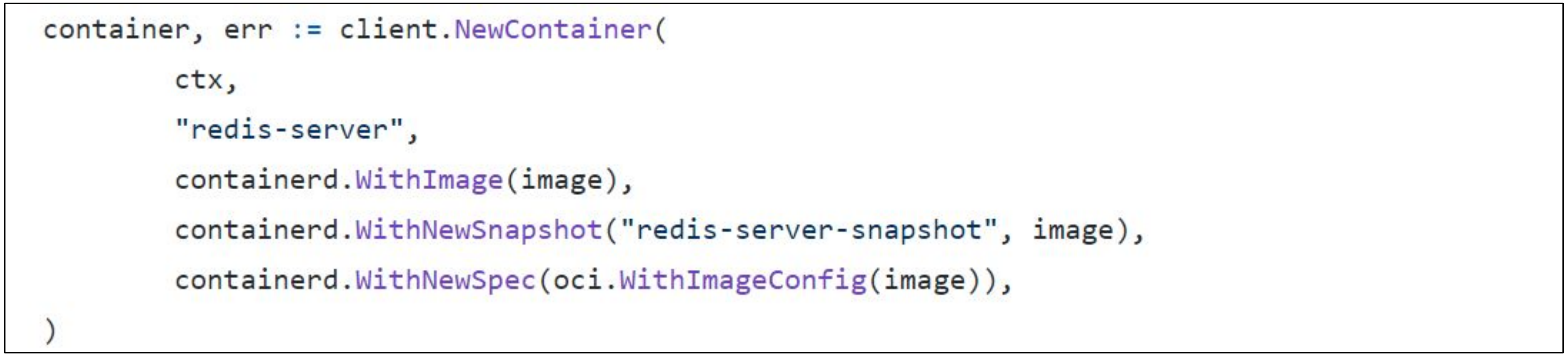
Example: https://containerd.io/docs/getting-started/



```
container, err := client.NewContainer(
        ctx,
        "redis-server",
        containerd.WithImage(image),
        containerd.WithNewSnapshot("redis-server-snapshot", image),
        containerd.WithNewSpec(oci.WithImageConfig(image)),
)
```

Implementing your own containerd client

Example: https://containerd.io/docs/getting-started/



```
container, err := client.NewContainer(
        ctx,
        "redis-server",
        containerd.WithImage(image),
        containerd.WithNewSnapshot("redis-server-snapshot", image),
        containerd.WithNewSpec(oci.WithImageConfig(image)),
)
```

You will add `WithXXX` options here:

- `oci.WithProcessArgs`
- `oci.WithMounts`
- `oci.WithMemoryLimit`
- `seccomp.WithProfile`
- ...

Implementing your own containerd client

In addition to the client, you will also want to implement OCI hooks and logger binary

- **OCI Hooks**: custom commands called on creation and deletion of containers
  - e.g., for setting up and tearing down CNI bridge and portmap
  - Optional, but necessary if you want your containers to be restarted automatically on host reboot
  - Example: https://github.com/containerd/nerdctl/blob/v0.7.2/run.go#L629-L663

- **Logger Binary**: custom command for handling container logs
  - e.g., store as a local file, transfer to fluentd, …
  - Example: https://github.com/containerd/nerdctl/blob/v0.7.2/run.go#L618-L627

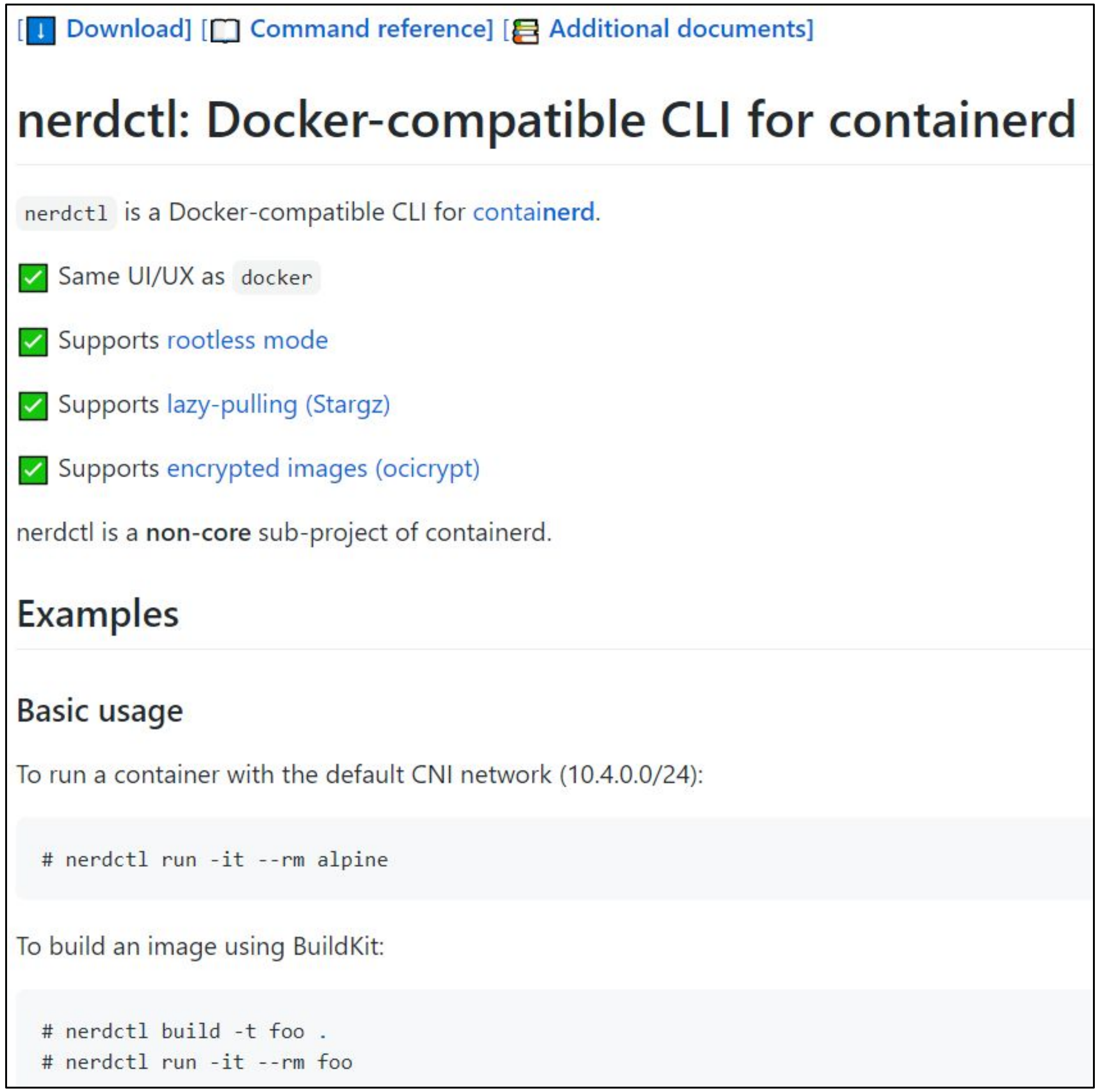
Implementing your own containerd client

Full example: **nerdctl**
https://github.com/containerd/nerdctl

Spun out from `ctr` tool with more practical features:

- Automatic restarting
- Port forwarding
- Logging
- Rootless
- Stargz
- OCIcrypt
- …

You may copy the code as the **"starter pack"** to create your own client :)



[⬇ Download] [Command reference] [Additional documents]

nerdctl: Docker-compatible CLI for containerd

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

✅ Same UI/UX as `docker`

✅ Supports rootless mode

✅ Supports lazy-pulling (Stargz)

✅ Supports encrypted images (ocicrypt)

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

Examples

Basic usage

To run a container with the default CNI network (10.4.0.0/24):

```
# nerdctl run -it --rm alpine
```

To build an image using BuildKit:

```
# nerdctl build -t foo .
# nerdctl run -it --rm foo
```

containerd 1.5 updates and future plan

Akihiro Suda, NTT Corporation

containerd 1.5 updates (April)

- Support **zstd** as an image compression algorithm
  - Faster than gzip
  - https://facebook.github.io/zstd/

- Support **NRI**: Node Resource Interface
  - Akin to CNI, but for managing resources, e.g., cgroup
  - https://github.com/containerd/nri

- Enable **OCIcrypt** decryption by default
  - Supported since 1.3, but it was not enabled by default
  - https://github.com/containers/ocicrypt https://github.com/containerd/imgcrypt

- **nerdctl** (contaiNERD ctl) joined containerd, as a non-core subproject
  - Docker-compatible CLI but with stargz and ocicrypt
  - https://github.com/containerd/nerdctl

containerd 1.5 updates (April)

- The CRI plugin repo (github.com/containerd/cri) is now merged into the main repo (github.com/containerd/containerd)
  - No visible change to users, but significantly simplifies contribution process

- Client library is now available as a Go module

Future plan

- Filesystem quota (#759)
- CRI support for user namespaces (KEP #2101)
  - Run Kubernetes pods as a user that is different from the daemon user
  - Akin to “Rootless Containers”, but different (and does not conflict, either)
- Chown-less user namespaces (#4734)
  - Requires idmapped mounts, introduced in kernel 5.12
- Pause-less pod sandboxes (#4131)
- More documentation (help wanted! 🙏)

Third party plugin updates

- Nydus Snapshotter https://github.com/dragonflyoss/image-service
  - Similar to Stargz Snapshotter but with a different image format
- OverlayBD Snapshotter https://github.com/alibaba/accelerated-container-image
  - Boot containers from iSCSI
- runu https://github.com/ukontainer/runu
  - Linux containers on macOS, using LKL (Linux Kernel Library)
- runj https://github.com/samuelkarp/runj
  - FreeBSD containers

Recap

- The de facto standard runtime for Kubernetes, but not only for Kubernetes

- Extensible with plugins
  - Runtime plugins, e.g., gVisor, Kata
  - Snapshotter plugins, e.g., Stargz Snapshotter
  - Stream processor plugins, e.g., OCIcrypt
  - Logging binary plugins, e.g., json-file
  - ...

- New subproject: nerdctl (https://github.com/containerd/nerdctl)
  - Like \`docker\` but with full features of containerd
  - Like \`ctr\` but with full user experience of \`docker\`
  - nerdctl run -d -p 80:80 --restart=always nginx

KubeCon | CloudNativeCon

Europe 2021

Virtual