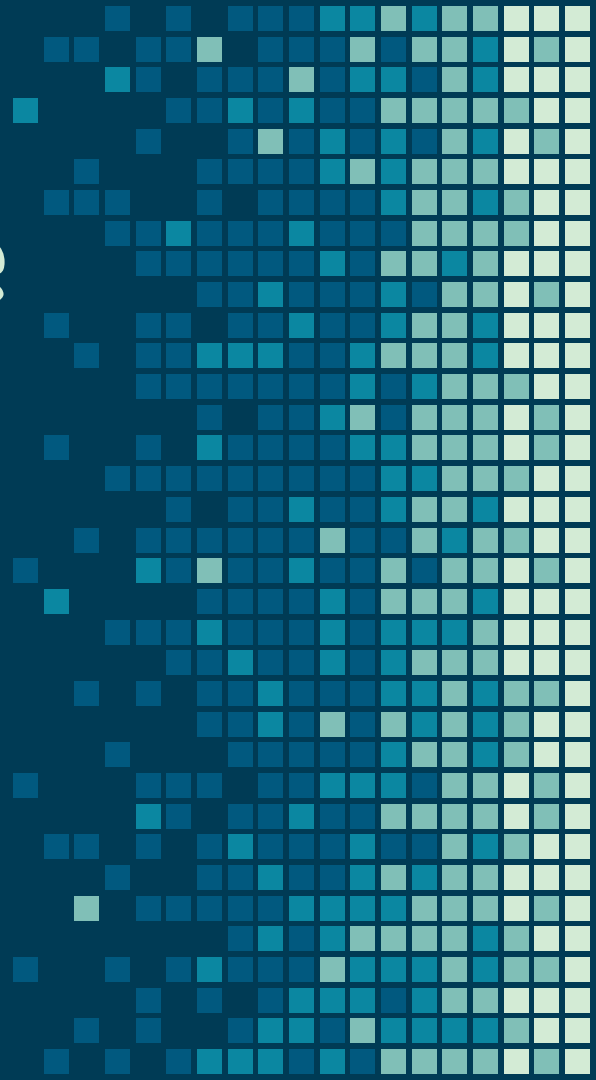




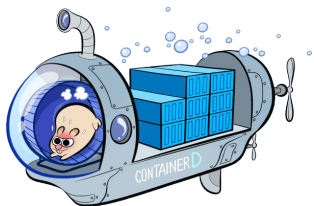
# containerd deep dive

Stephen Day  
@stevvooe

May 4, 2018  
KubeCon EU



# A Brief History



APRIL 2016 ..... Containerd "0.2" announced, Docker 1.11

***Management/Supervisor for the OCI runc executor***

Announce expansion of containerd OSS project ..... DECEMBER 2016



***Containerd 1.0: A core container runtime project for the industry***

MARCH 2017 ..... Containerd project contributed to CNCF



# <https://github.com/containerd/containerd>

containerd / containerd

Unwatch 167

Unstar 1,800

Fork 378

Code

Issues 84

Pull requests 16

Projects 0

Wiki

Insights

An open and reliable container runtime <https://containerd.io>

containerd

oci

containers

docker

cncf

2,673 commits

6 branches

25 releases

104 contributors

Apache-2.0

Branch: master

New pull request

Create new file

Upload files

Find file

Clone or download

mlaventure Merge pull request #1665 from crosbymichael/bump-runc

Latest commit 3679a55 3 days ago

api Refactor differ into separate package 12 days ago

archive Merge pull request #1631 from dmcgowan/cancel-unpack 6 days ago

cmd Merge pull request #1652 from crosbymichael/cr-image 5 days ago

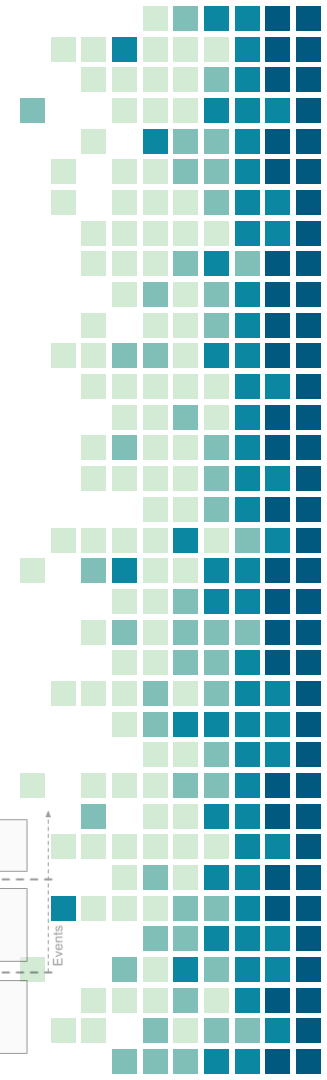
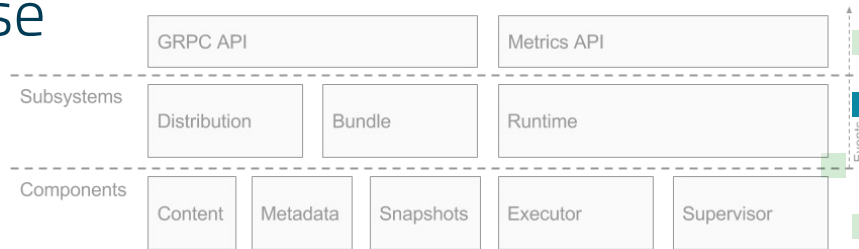
# Why Containerd 1.0?

- Continue projects **spun out** from monolithic Docker engine
- Expected use **beyond** Docker engine (Kubernetes CRI)
- Donation to **foundation** for broad industry collaboration
  - Similar to runc/libcontainer and the OCI



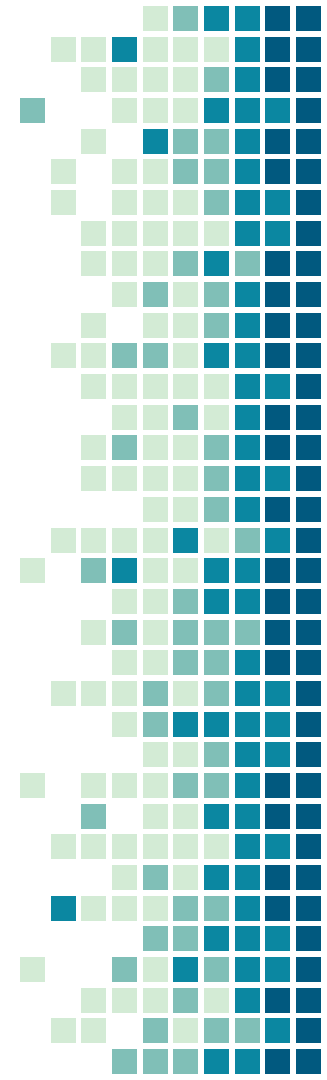
# Technical Goals/Intentions

- Clean **gRPC-based** API + client library
- Full **OCI** support (runtime and image spec)
- **Stability** and **performance** with tight, well-defined core of container function
- **Decoupled** systems (image, filesystem, runtime) for pluggability, reuse



# Requirements

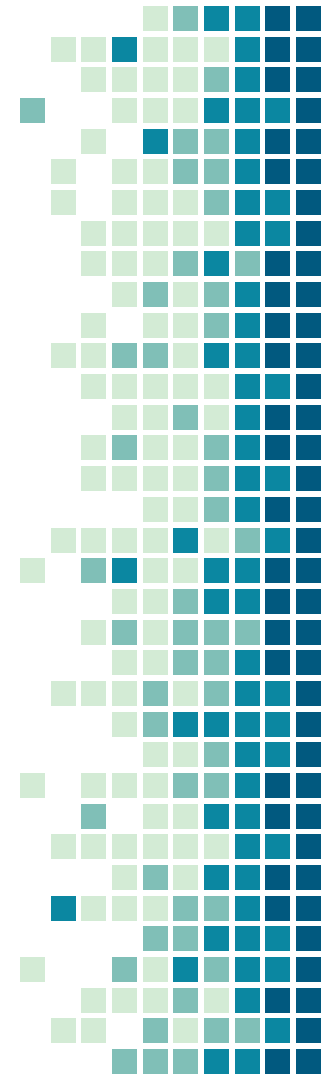
- **A la carte**: use only what is required
- Runtime **agility**: fits into different platforms
  - Pass-through container configuration (direct OCI)
- **Decoupled**
- Use **known-good** technology
  - OCI container runtime and images
  - gRPC for API
  - Prometheus for Metrics



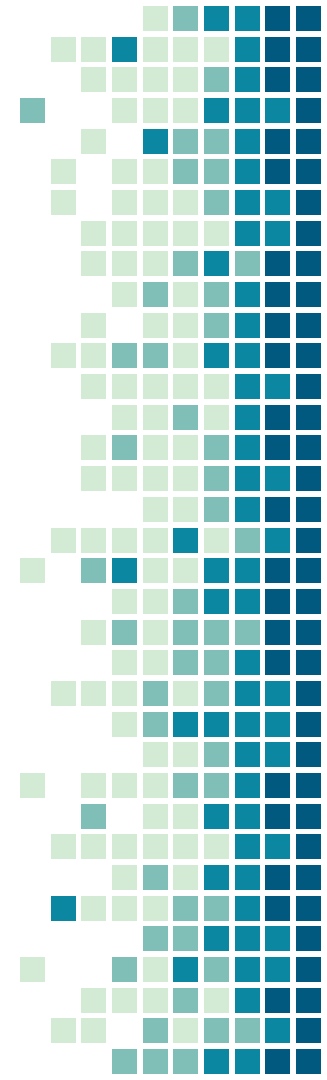
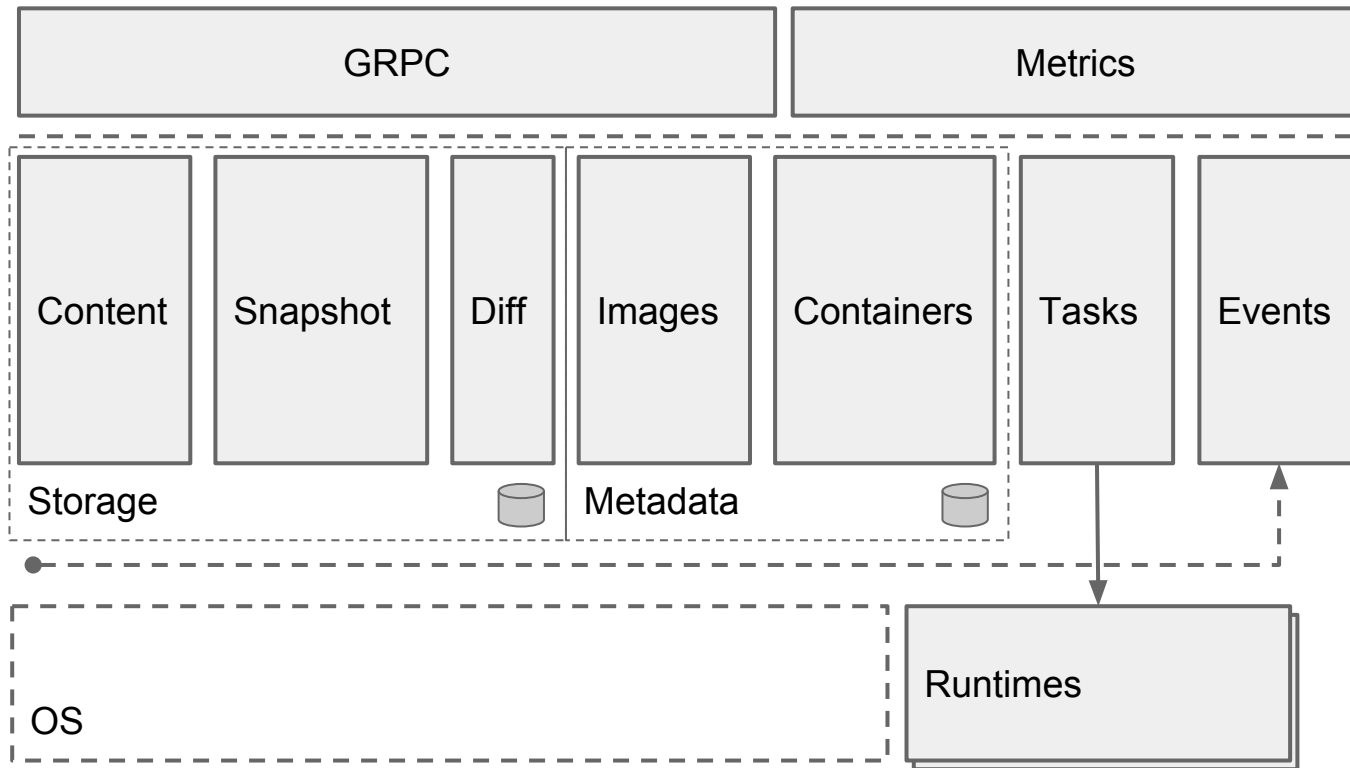


## Use cases

- Container API
  - Implementations
  - Building Images
  - Container OS
- **EXAMPLES**
  - Docker/Moby
  - Kubernetes CRI
  - alibaba/pouch
  - SwarmKit (experimental)
  - LinuxKit
  - BuildKit
  - IBM Cloud

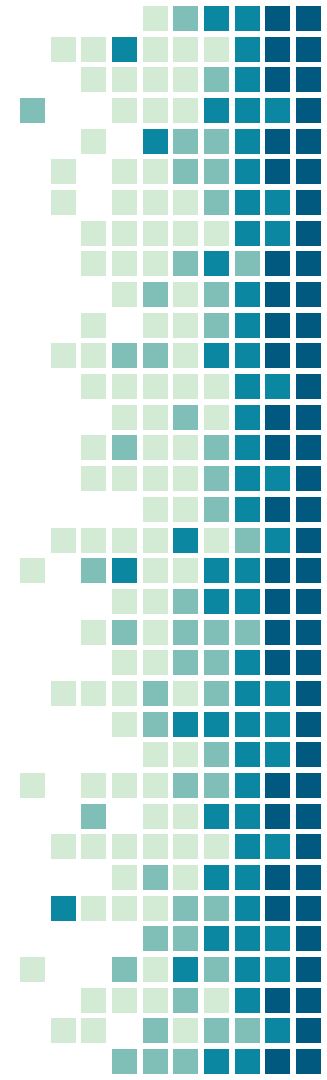
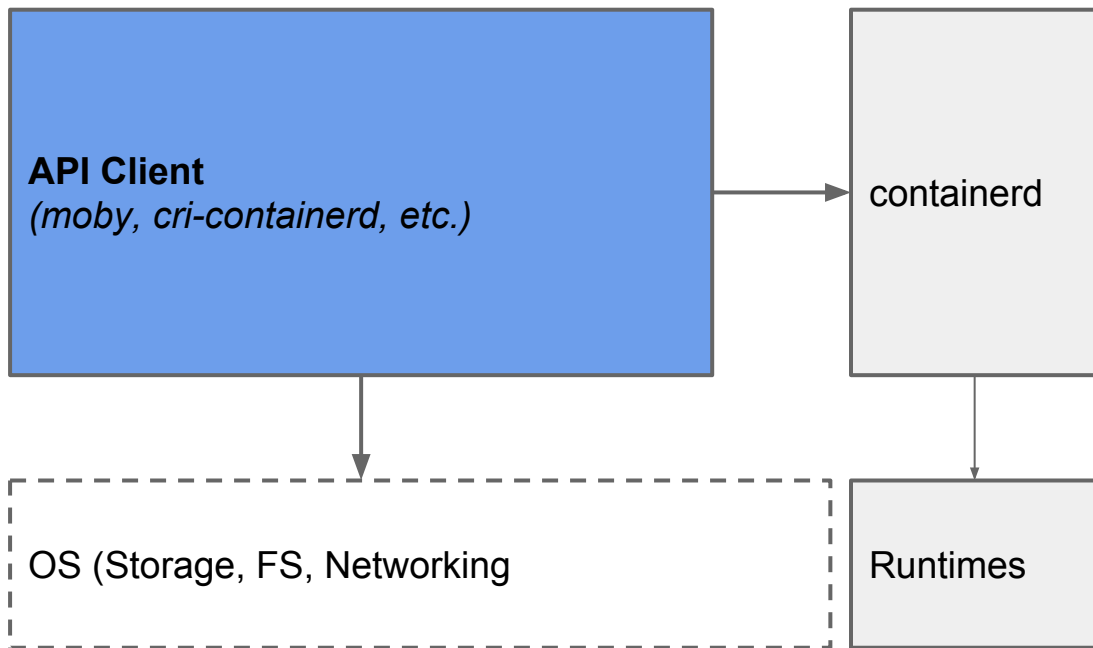


# Architecture





# Architecture



# Containerd: Rich Go API

## **Getting Started**

<https://github.com/containerd/containerd/blob/master/docs/getting-started.md>

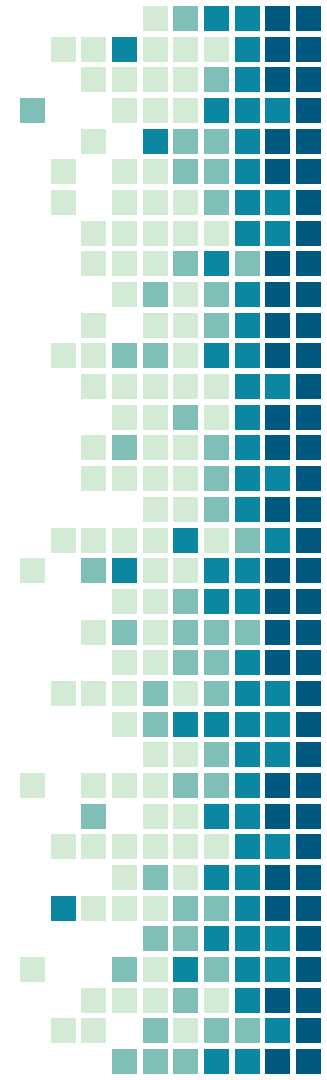
## **GoDoc**

<https://godoc.org/github.com/containerd/containerd>



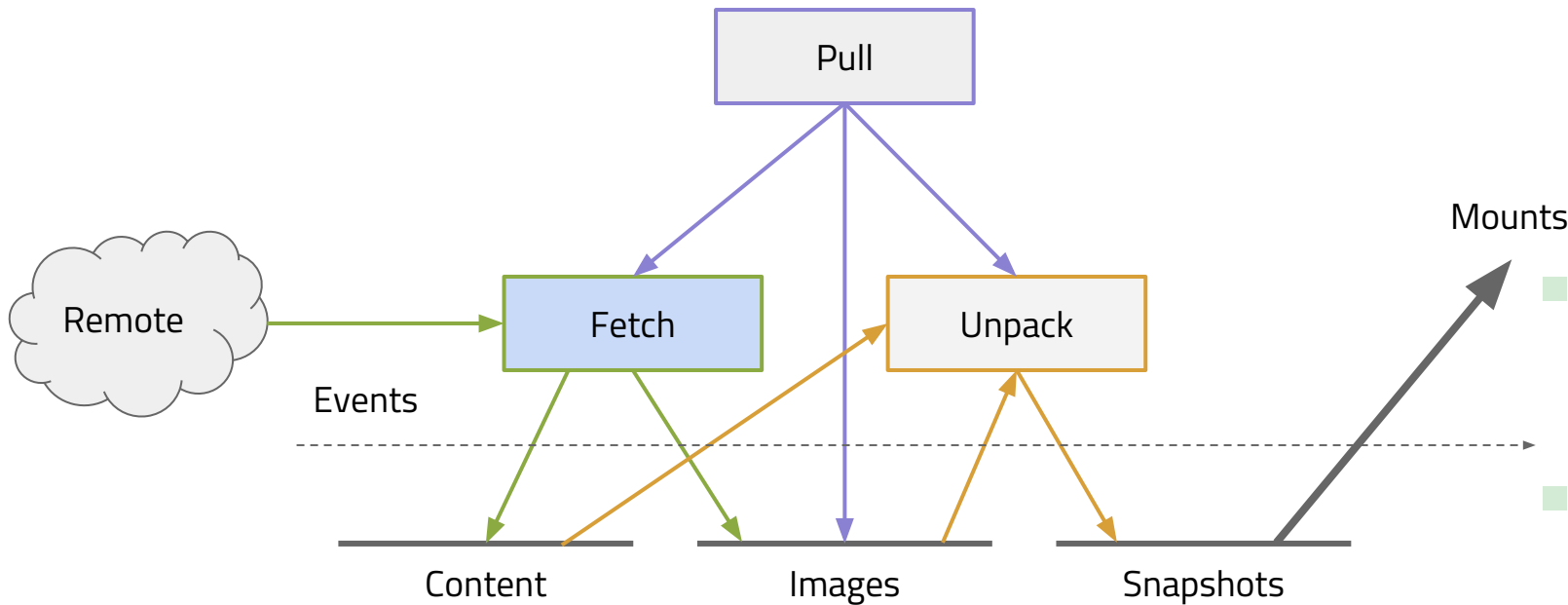
# Pulling an Image

What do runtimes need?



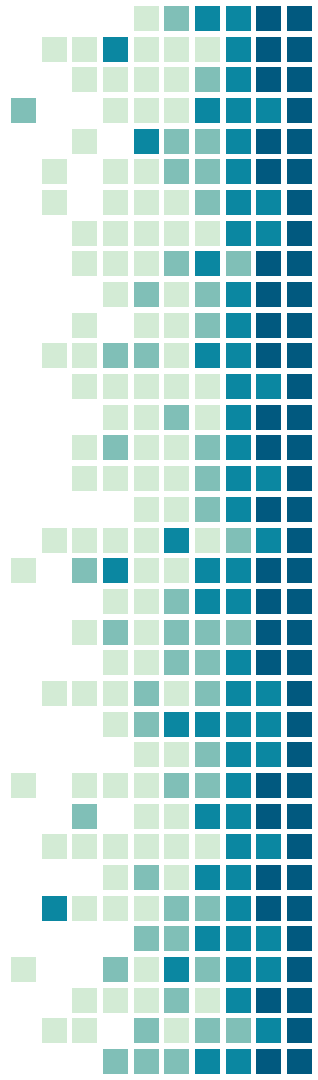
# Pulling an Image

## Data Flow

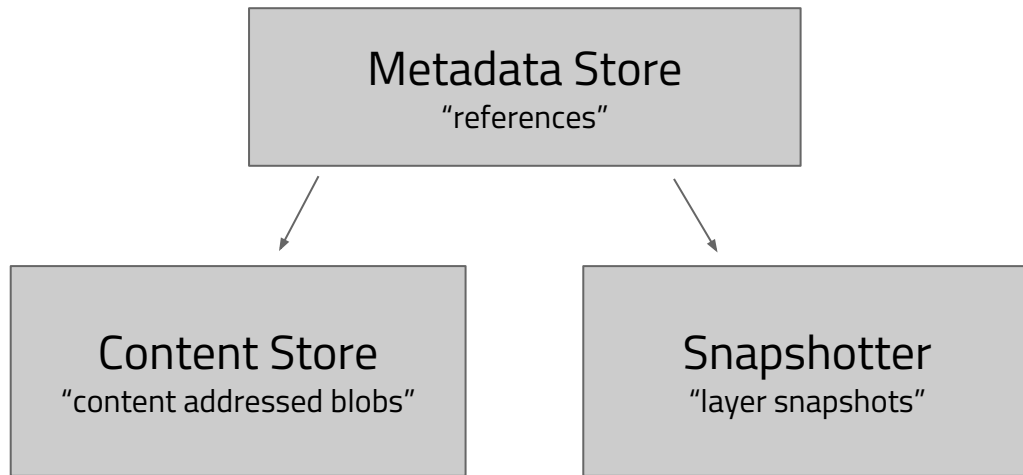


# Snapshotters

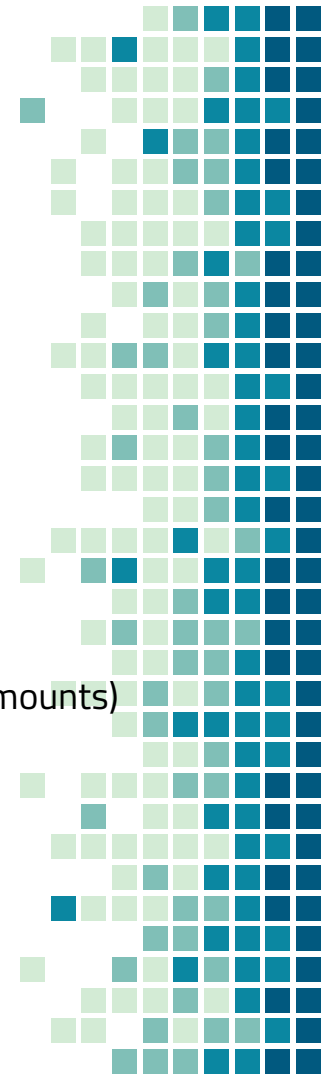
How do you build a container root filesystem?



# containerd Storage Architecture



Config  
Rootfs (mounts)



# Example: Investigating Root Filesystem

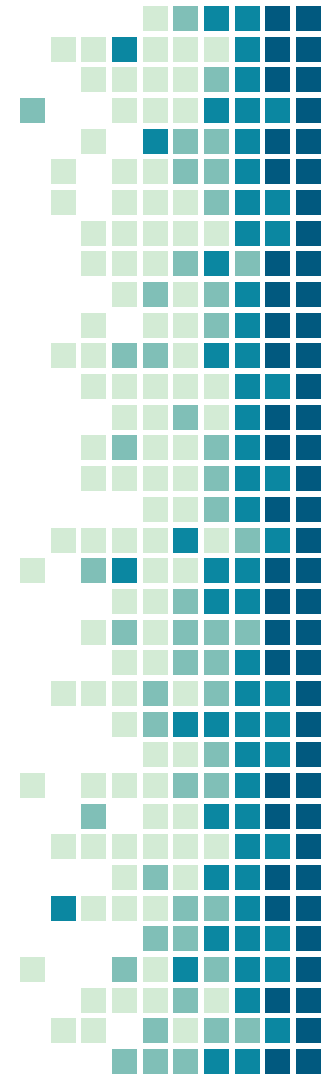
```
$ ctr snapshot ls
```

```
...
```

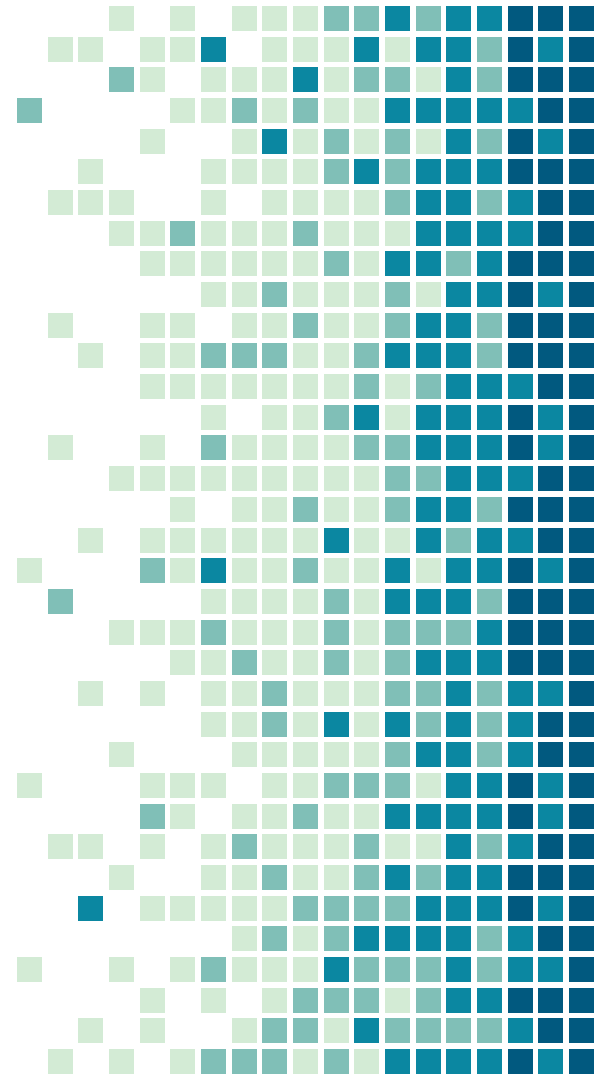
```
$ ctr snapshot tree
```

```
...
```

```
$ ctr snapshot mounts <target> <id>
```



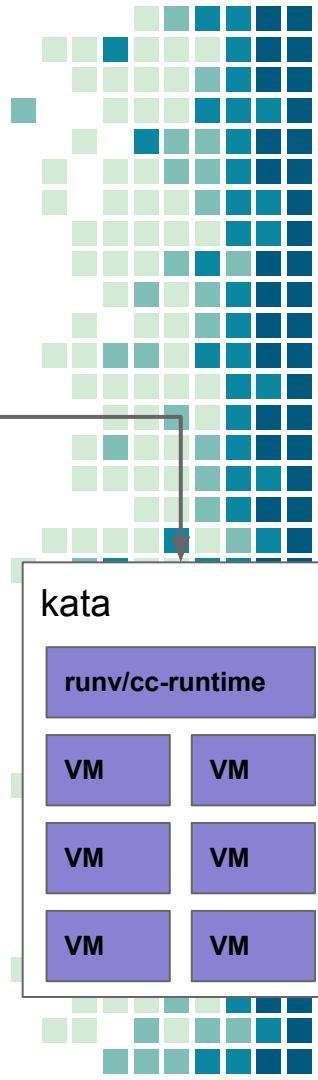
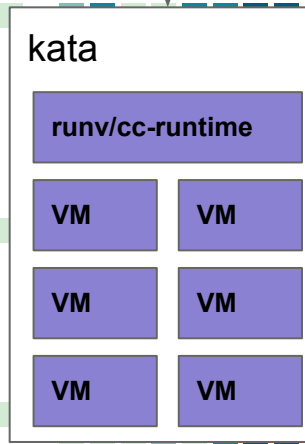
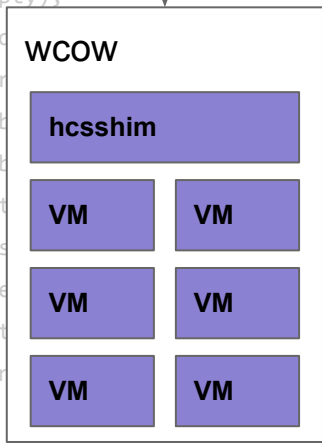
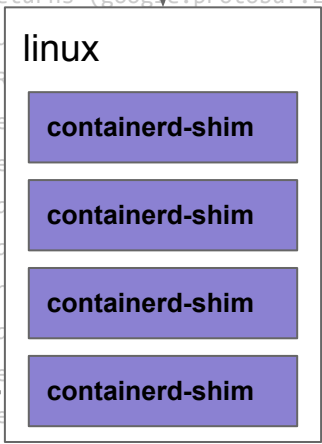
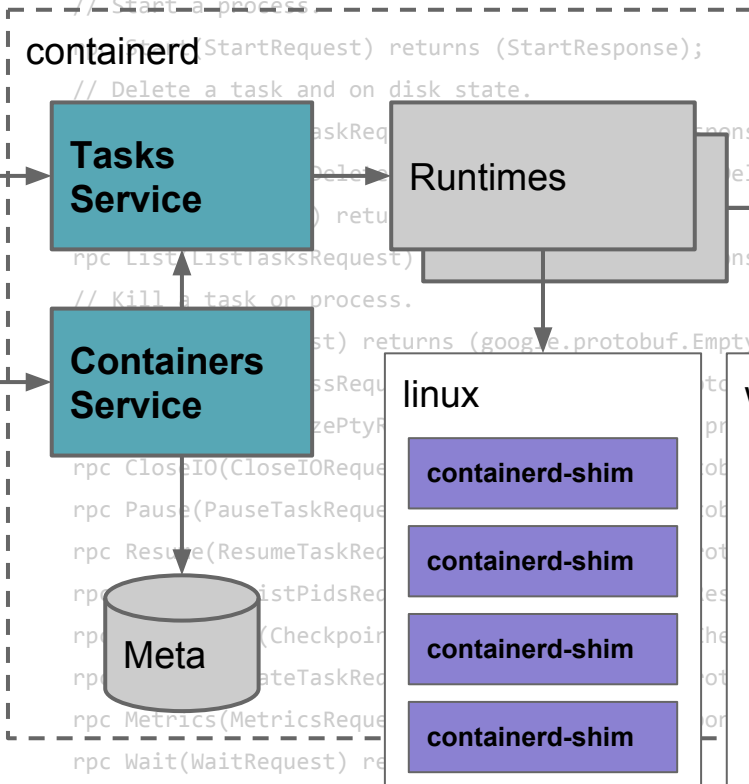
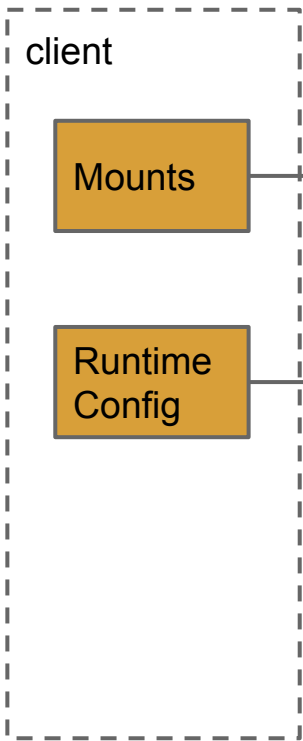
# Running a container



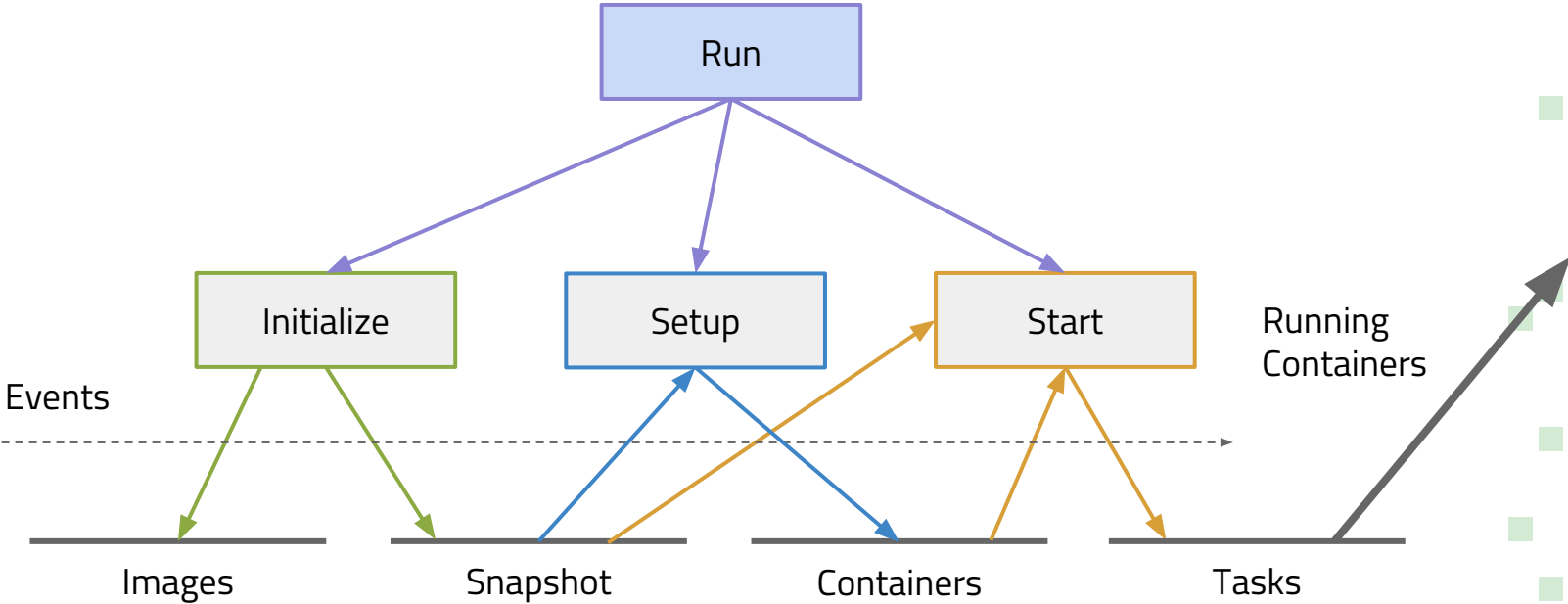


# Tasks and Runtime

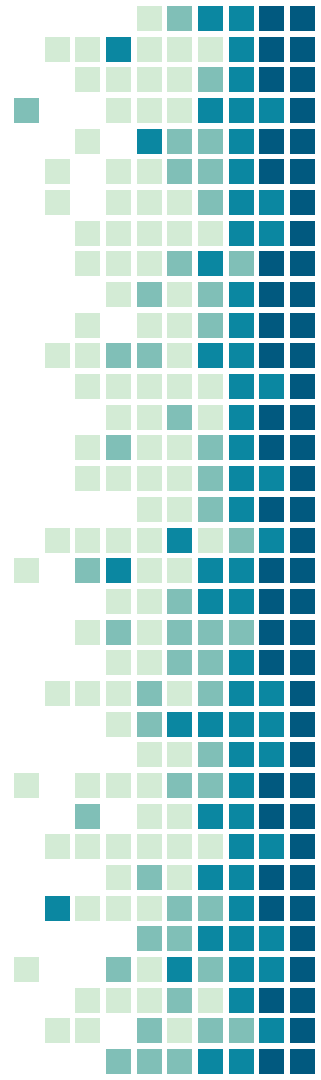
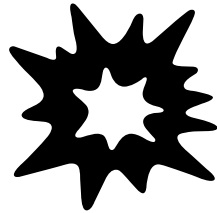
```
service Tasks {  
  // Create a task.  
  rpc Create(CreateTaskRequest) returns (CreateTaskResponse);  
  // Start a process.  
  rpc Start(StartRequest) returns (StartResponse);  
  // Delete a task and on disk state.  
  rpc Delete(DeleteTaskRequest) returns (DeleteTaskResponse);  
  // List tasks.  
  rpc List(ListTasksRequest) returns (ListTasksResponse);  
  // Kill a task on process.  
  rpc Kill(KillTaskRequest) returns (KillTaskResponse);  
  // Close IO.  
  rpc CloseIO(CloseIORequest) returns (CloseIOResponse);  
  // Pause a task.  
  rpc Pause(PauseTaskRequest) returns (PauseTaskResponse);  
  // Resume a task.  
  rpc Resume(ResumeTaskRequest) returns (ResumeTaskResponse);  
  // List PIDs.  
  rpc ListPids(ListPidsRequest) returns (ListPidsResponse);  
  // Checkpoint a task.  
  rpc Checkpoint(CheckpointTaskRequest) returns (CheckpointTaskResponse);  
  // Metrics.  
  rpc Metrics(MetricsRequest) returns (MetricsResponse);  
  // Wait.  
  rpc Wait(WaitRequest) returns (WaitResponse);  
}
```



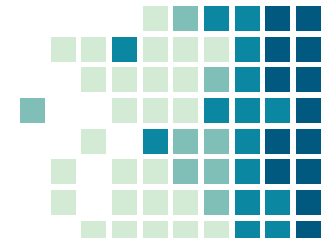
# Starting a Container



Demo



# Example: Pull an Image

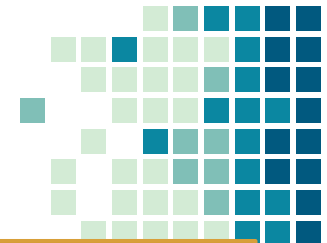


Via **ctr** client:

```
$ export \  
CONTAINERD_NAMESPACE=example  
  
$ ctr pull \  
docker.io/library/redis:alpine  
  
$ ctr image ls  
...
```

```
import (  
    "context"  
  
    "github.com/containerd/containerd"  
    "github.com/containerd/containerd/namespaces"  
)  
  
// connect to our containerd daemon  
client, err := containerd.New("run/containerd/containerd.sock")  
defer client.Close()  
  
// set our namespace to "example":  
ctx := namespaces.WithNamespace(context.Background(), "example")  
  
// pull the alpine-based redis image from DockerHub:  
image, err := client.Pull(ctx,  
    "docker.io/library/redis:alpine",  
    containerd.WithPullUnpack)
```

# Example: Run a Container

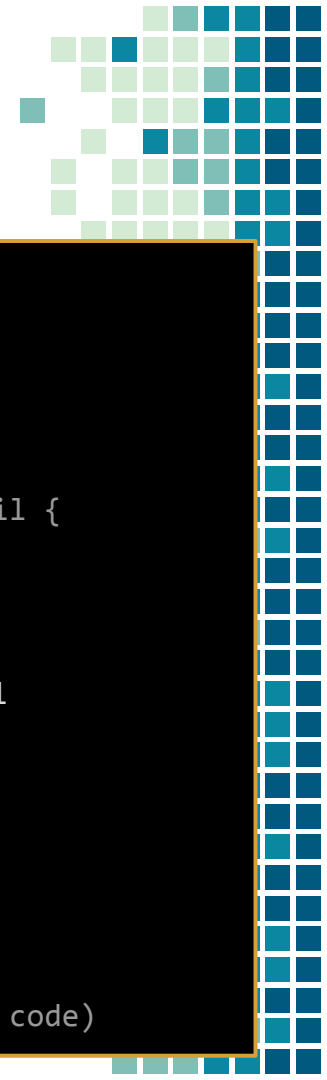


Via **ctr** client:

```
$ export \  
CONTAINERD_NAMESPACE=example  
  
$ ctr run -t \  
docker.io/library/redis:alpine \  
redis-server  
  
$ ctr c  
...
```

```
// create our container object and config  
container, err := client.NewContainer(ctx,  
    "redis-server",  
    containerd.WithImage(image),  
    containerd.WithNewSpec(containerd.WithImageConfig(image)),  
)  
defer container.Delete()  
  
// create a task from the container  
task, err := container.NewTask(ctx, containerd.Stdio)  
defer task.Delete(ctx)  
  
// make sure we wait before calling start  
exitStatusC, err := task.Wait(ctx)  
  
// call start on the task to execute the redis server  
if err := task.Start(ctx); err != nil {  
    return err  
}
```

# Example: Kill a Task



Via **ctr** client:

```
$ export \  
CONTAINERD_NAMESPACE=example  
  
$ ctr t kill redis-server  
  
$ ctr t ls  
...
```

```
// make sure we wait before calling start  
exitStatusC, err := task.Wait(ctx)  
  
time.Sleep(3 * time.Second)  
  
if err := task.Kill(ctx, syscall.SIGTERM); err != nil {  
    return err  
}  
  
// retrieve the process exit status from the channel  
status := <-exitStatusC  
code, exitedAt, err := status.Result()  
if err != nil {  
    return err  
}  
  
// print out the exit code from the process  
fmt.Printf("redis-server exited with status: %d\n", code)
```

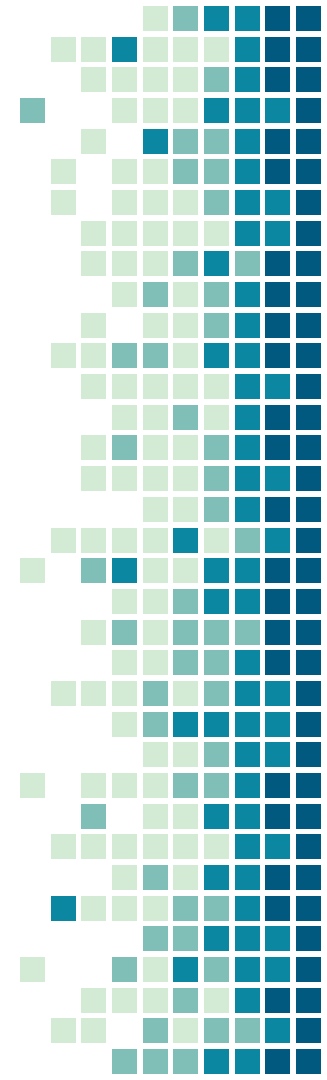
# Example: Customize OCI Configuration

```
// WithHtop configures a container to monitor the host via `htop`
func WithHtop(s *specs.Spec) error {
    // make sure we are in the host pid namespace
    if err := containerd.WithHostNamespace(specs.PIDNamespace)(s); err != nil {
        return err
    }
    // make sure we set htop as our arg
    s.Process.Args = []string{"htop"}
    // make sure we have a tty set for htop
    if err := containerd.WithTTY(s); err != nil {
        return err
    }
    return nil
}
```

**With{func}** functions cleanly separate modifiers

# Release

<https://github.com/containerd/containerd/blob/master/RELEASES.md>



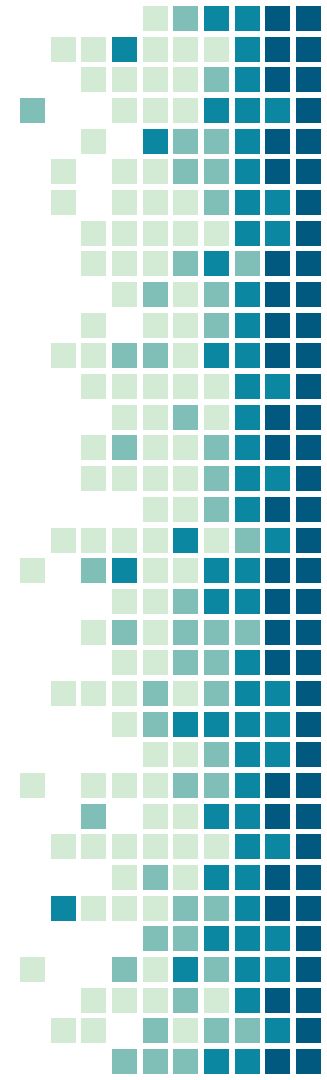


# Support Horizon

Release	Status	Start	End of Life
<a href="#">0.0</a>	End of Life	Dec 4, 2015	-
<a href="#">0.1</a>	End of Life	Mar 21, 2016	-
<a href="#">0.2</a>	End of Life	Apr 21, 2016	December 5, 2017
<a href="#">1.0</a>	Active	December 5, 2017	December 5, 2018
<a href="#">1.1</a>	Active	April 23, 2018	max(April 23, 2019, release of 1.2.0, Kubernetes 1.10 EOL)
<a href="#">1.2</a>	Next	TBD	max(TBD+1 year, release of 1.3.0)

# Supported Components

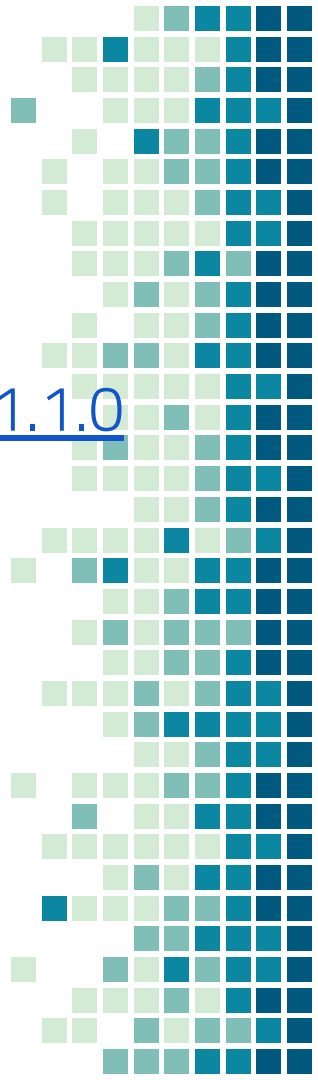
Component	Status	Stabilized Version	Links
GRPC API	Stable	1.0	<a href="#">api/</a>
Metrics API	Stable	1.0	-
Go client API	Unstable	1.2 <i>tentative</i>	<a href="#">godoc</a>
CRI GRPC API	Unstable	v1alpha2 <i>current</i>	<a href="#">api/</a>
ctr tool	Unstable	Out of scope	-



# 1.1

<https://github.com/containerd/containerd/releases/tag/v1.1.0>

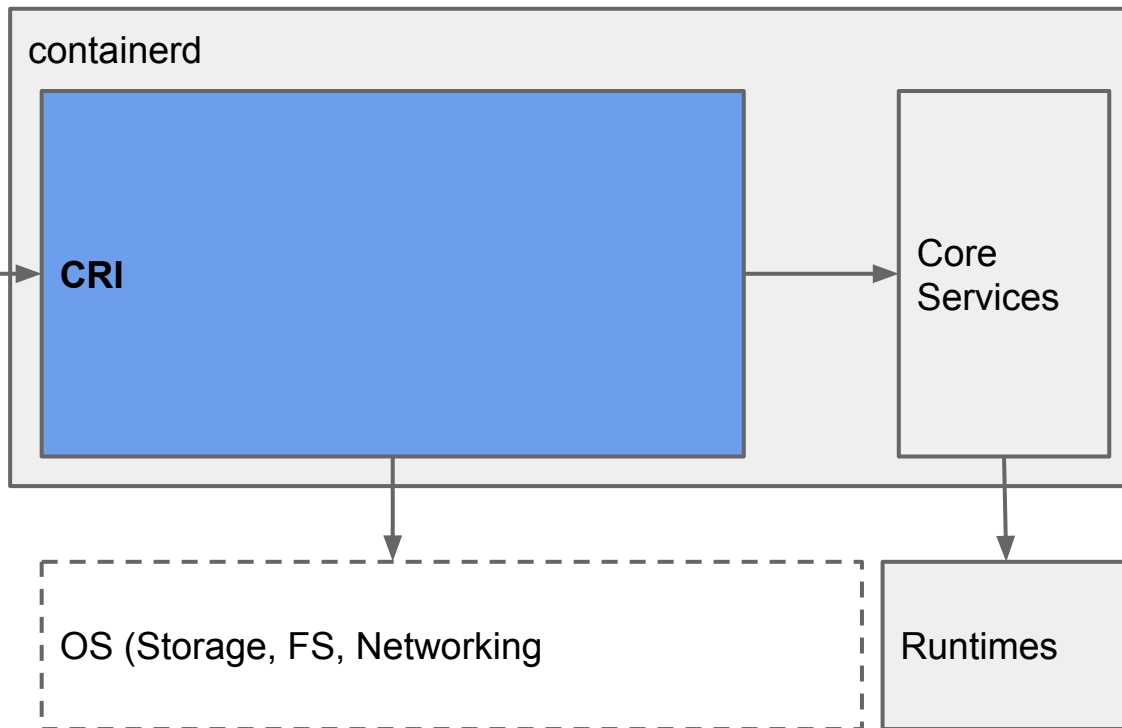
- Merged in Kubernetes CRI Support
- Additional Snapshotter: ZFS, AUFS and native



# Kubernetes CRI Support

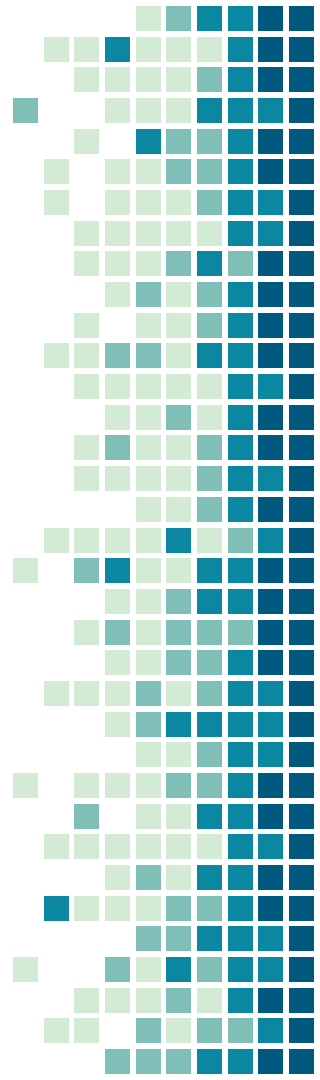


kubelet



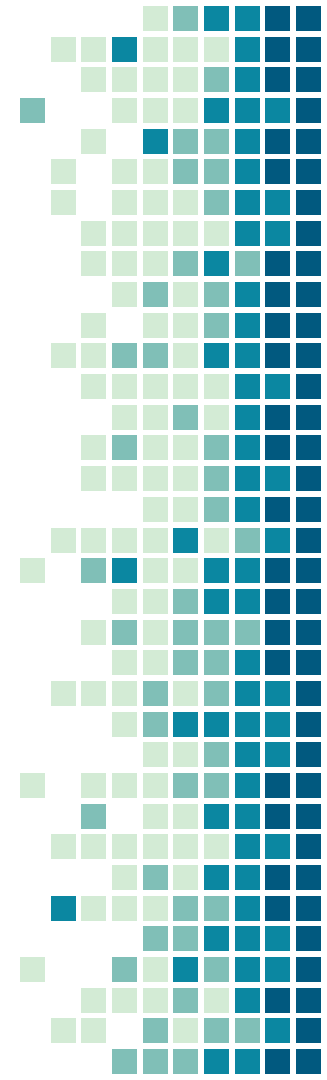
# Going further with **containerd**

- **Contributing:** <https://github.com/containerd/containerd>
  - Bug fixes, adding tests, improving docs, validation
- **Using:** See the getting started documentation in the docs folder of the repo
- **Porting/testing:** Other architectures & OSs, stress testing (see bucketbench, containerd-stress):
  - `git clone <repo>`, `make binaries`, `sudo make install`
- **Upstream Testing:**  
<https://k8s-testgrid.appspot.com/sig-node-containerd>



# KubeCon Talks

- **Take Control of your Filesystems with containerd's Snapshotters**
  - Wednesday May 2, 2018 16:25 - 17:00
  - C1-M5
- **containerd Deep Dive**
  - Friday May 4, 2018 15:40 - 16:15
  - B5-M1+3
- 



# Thank You! Questions?

- **Stephen Day**
  - <https://github.com/stevvooe>
  - @stevvooe
  - Docker Community Slack

