

# rkt and Kubernetes

What's new (and coming) with  
Container Runtimes and Kubernetes

---

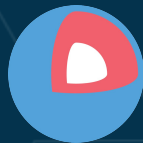




# Jonathan Boulle

[github.com/jonboulle](https://github.com/jonboulle) - @baronboulle

---



Core OS



A CLI for running app containers on Linux.

Focuses on:

- Security
- Composability
- Standards/Compatibility



# rkt - a brief history

- **December 2014 - v0.1.0**
  - Prototype
  - Drive conversation (security, standards) and competition (healthy OSS) in container ecosystem
- **February 2016 - v1.0.0**
  - (already) used in production
  - API stability guarantees
- **June 2016 - v1.8.0 (?)**
  - Packaged in Debian, Fedora, Arch, NixOS



A CLI for running app containers on Linux.

Focuses on:

- Security
- Composability
- Standards/Compatibility





A CLI for running app containers on Linux.

Focuses on:

- **Security**
- Composability
- Standards/Compatibility



# How rkt does security

- **UX: *"secure-by-default"***
  - Verify image signatures by default
  - Verify image integrity by default
- **Architecture: *Unix philosophy***
  - Well-defined operations
  - No central privileged long-running components
  - Separate privileges for different operations ("fetch" operations shouldn't need root)

# How rkt does security

## Classic and modern Linux technologies

- User namespaces
  - container euid != host euid
- SELinux contexts
  - isolate individual pods
- Support for VM containment
  - lightweight hypervisor (= hardware isolation)
- TPM measurements
  - Tamper-proof audit log of what's running



# How rkt does security (soon)

- Finer-grained Linux capabilities
- Tighter seccomp defaults
- SELinux across more platforms
- cgroup2 + cgroup namespaces support
- Support for more hypervisors (QEMU)
- Unprivileged containers (run entirely as non-root)



A CLI for running app containers on Linux.

Focuses on:

- **Security**
- Composability
- Standards/Compatibility





A CLI for running app containers on Linux.

Focuses on:

- Security
- **Composability**
- Standards/Compatibility



# How rkt does composability

- **"External" composability**
  - Integrating well with init system(s) is a priority
  - Aims to work well with other projects, like cluster orchestration (more on this in a moment)
- **"Internal" composability**
  - Stage-based architecture
  - Swappable execution engines that actually runs the container

# How rkt does composability

- **"External" composability**
  - Simple process model: rkt command *is* the container
  - Any context applied to rkt (cgroups, etc) applies transitively to the pod and the apps inside
  - Optional gRPC (HTTP2+Protobuf) API server to facilitate more efficient introspection

bash/systemd/kubelet... (invoking process)

└─▶ rkt (stage0)

└─▶ pod (stage1)

└─▶ app1 (stage2)

└─▶ app2 (stage2)

```
systemd-run -p MemoryLimit=1G rkt run ...
```

└─ rkt (stage0)

└─ pod (stage1)

└─ app1 (stage2)

└─ app2 (stage2)

# How rkt does composability

- **"Internal" composability**
  - Staged architecture
  - "rkt" is the UX/API, container technology is an implementation detail
- **Available stage1s**
  - cgroups + Linux namespaces (default)
  - LKVM
  - chroot ("fly")
  - QEMU (landing soon)



bash/systemd/kubelet... (invoking process)

└─ rkt (stage0)

└─ pod (stage1) - **systemd-nspawn**

└─ app1 (stage2)

└─ app2 (stage2)

bash/systemd/kubelet... (invoking process)

└─▶ rkt (stage0)

└─▶ pod (stage1) - 1kvm

└─▶ app1 (stage2)

└─▶ app2 (stage2)

bash/systemd/kubelet... (invoking process)

└─▶ rkt (stage0)

└─▶ pod (stage1) - qemu

└─▶ app1 (stage2)

└─▶ app2 (stage2)

bash/systemd/kubelet... (invoking process)



rkt (stage0)



app (stage1) - fly

bash/systemd/kubelet... (invoking process)



# rkt + fly

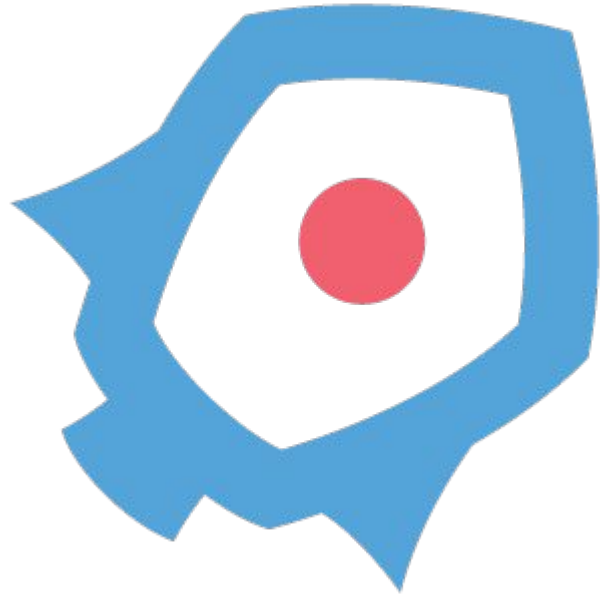
- Leverage the safety and security features of rkt (image retrieval, signature validation) without the containment
- Lightweight, isolated package manager
- Enables interesting use cases (more later)



A CLI for running app containers on Linux.

Focuses on:

- Security
- **Composability**
- Standards/Compatibility





A CLI for running app  
containers on Linux.

Focuses on:

- Security
- Composability
- **Standards/Compatibility**





# How rkt does standards/compatibility

- (original) implementation of **appc**
  - first attempt at a well-defined container spec
- Uses **CNI** for networking
  - common plumbing used by many other projects (Kubernetes, Cloud Foundry, Project Calico, Weave, ...)
- Can run **Docker images** natively (V1, V2, ...)
- Developers participate actively in **standardisation efforts**
  - appc, OCI, CNCF
  - rkt will be fully OCI compliant

# How do standards work?

- appc (December 2014)
  - some adoption, but (intended to be) deprecated in favour of
- OCI (June 2015)
  - ongoing questions about exact scope
- CNCF (December 2015)
  - "harmonising cloud-native technologies"



# How do (image format) standards work?

	Docker v1	appc	Docker v2.2	OCI (in progress)
<b>Introduced</b>	2013	December 2014	April 2015	April 2016
<b>Content-addressable</b>	No	Yes	Yes	Yes
<b>Signable</b>	No	Yes, optional	Yes, optional	Yes, optional
<b>Federated namespace</b>	Yes	Yes	Yes	Yes
<b>Delegatable DNS namespace</b>	No	Yes	No	Yes

# How do (networking) standards work?

CNI, the "Container Networking Interface"

Used by:

- rkt
- Kurma
- Kubernetes
- Cloud Foundry
- Weave
- Project Calico
- Contiv Networking

```
{
  "name": "mynet",
  "type": "bridge",
  "bridge": "cni0",
  "isGateway": true,
  "ipMasq": true,
  "ipam": {
    "type": "host-local",
    "subnet": "10.22.0.0/16",
    "routes": [
      { "dst": "0.0.0.0/0" }
    ]
  }
}
```



A CLI for running app containers on Linux.

Focuses on:

- Security
- Composability
- Standards/Compatibility



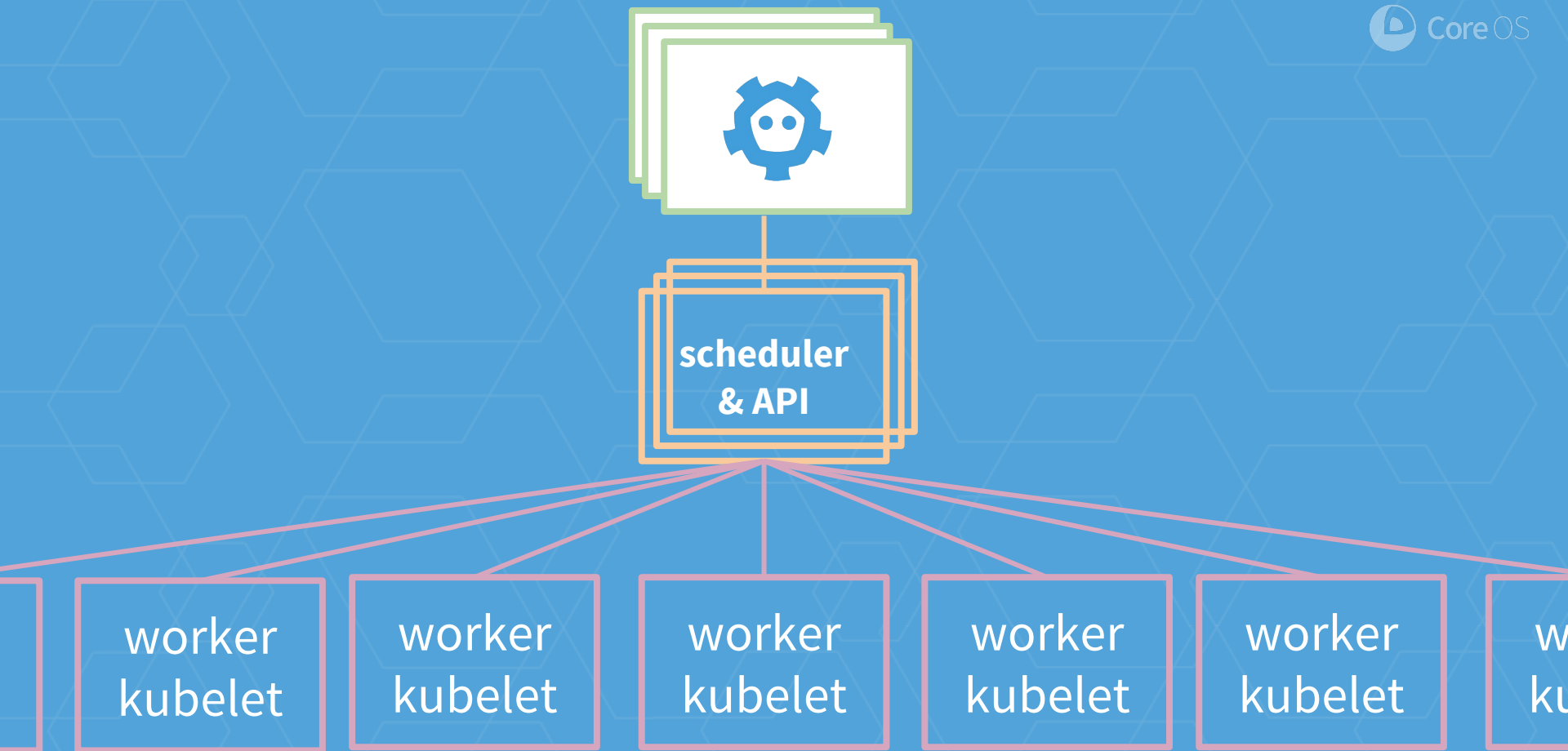
# Kubernetes

Cluster-level container orchestration.

Handles:

- Scheduling/Upgrades
- Failure recovery
- Scaling





# Kubernetes

Cluster-level **container** orchestration.

Handles:

- Scheduling/Upgrades
- Failure recovery
- Scaling





# Kubernetes

Cluster-level **container** orchestration.

- Today , container = Docker container
- No reason for this strictly to be the case
- Kubernetes API (mostly) exposes only *pods*



# Kubernetes

Cluster-level **pod** orchestration.

- Pod is a grouping of one or more applications sharing certain context (networking, volumes, ...)



# How does it work?

- kubelet is a daemon that runs on every worker node in a Kubernetes cluster
- Kubernetes schedules *pods* for a kubelet to run
- kubelet delegates to container runtime to perform all container-related operations (start container, fetch images, ...)

# How does it work?

- **Kubelet code provides a Runtime (golang) interface**
  - SyncPod()
  - GetPod()
  - KillPod()
  - ...
- **in theory, anyone can implement this**
- **in practice, lots of Docker assumptions**

# Kubernetes + rkt = rktnetes

Have Kubernetes use rkt as the container runtime.

rkt handles:

- Image discovery
- Image fetching
- Pod execution



# How does it work?

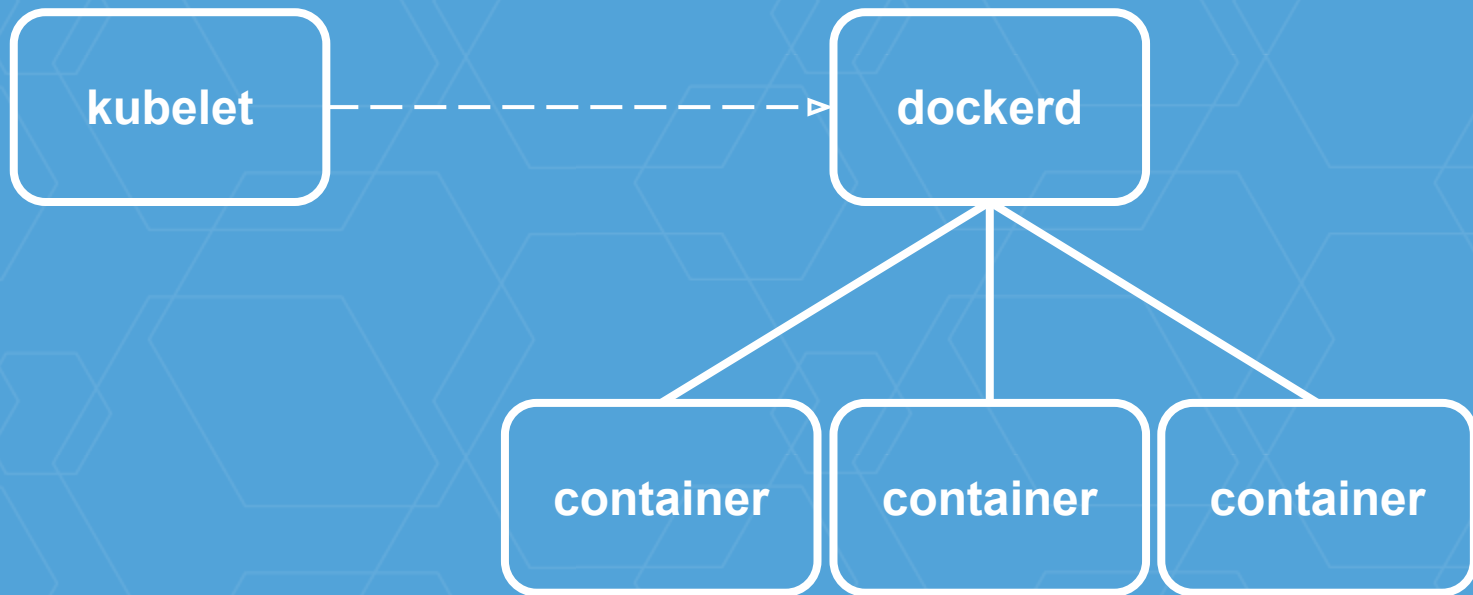
Pre-rktnetes (default):

- Kubelet talks to the Docker daemon for all tasks

With rktnetes changes:

- Kubelet talks to the rkt API daemon for read-only tasks
  - e.g. list pods, get logs
- Kubelet execs rkt directly for preparatory tasks
  - e.g. fetch images, create pod root filesystems
- Kubelet talks to systemd for running pods via rkt
  - e.g. launch containers

# Kubelet + Docker (default)



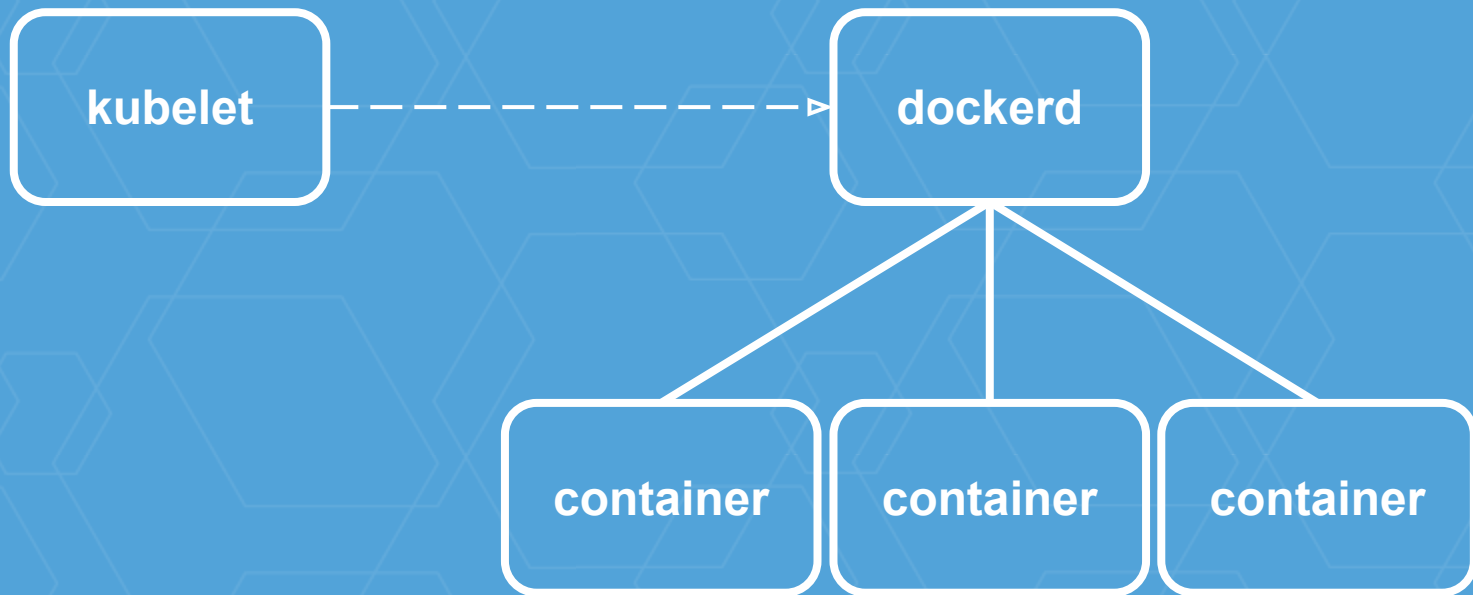
# Kubelet + Docker (default)

## Problems

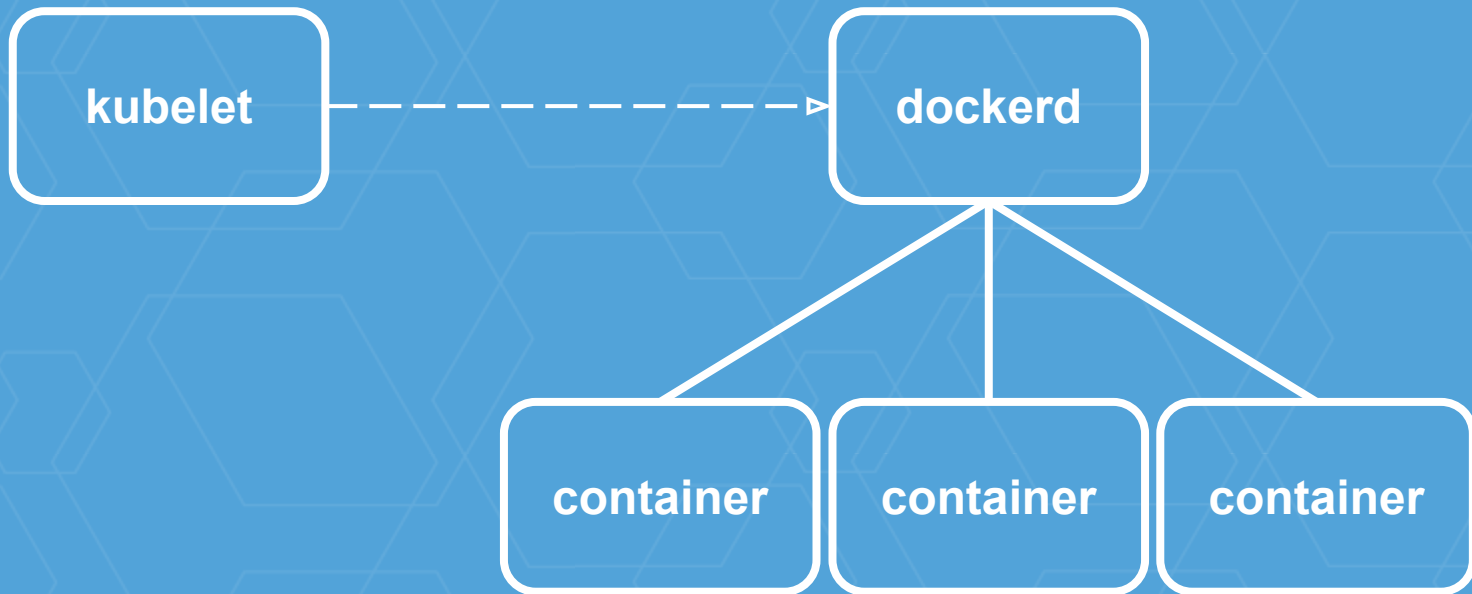
- ***Docker doesn't understand pods***
  - kubelet must maintain pod<->container mapping
  - "infra container" to hold namespaces for pod
- ***dockerd = SPOF for the node***
  - if Docker goes down, so do all containers
- ***Docker doesn't interact well with systemd***
  - who looks after cgroups?



# Kubelet + Docker (default)



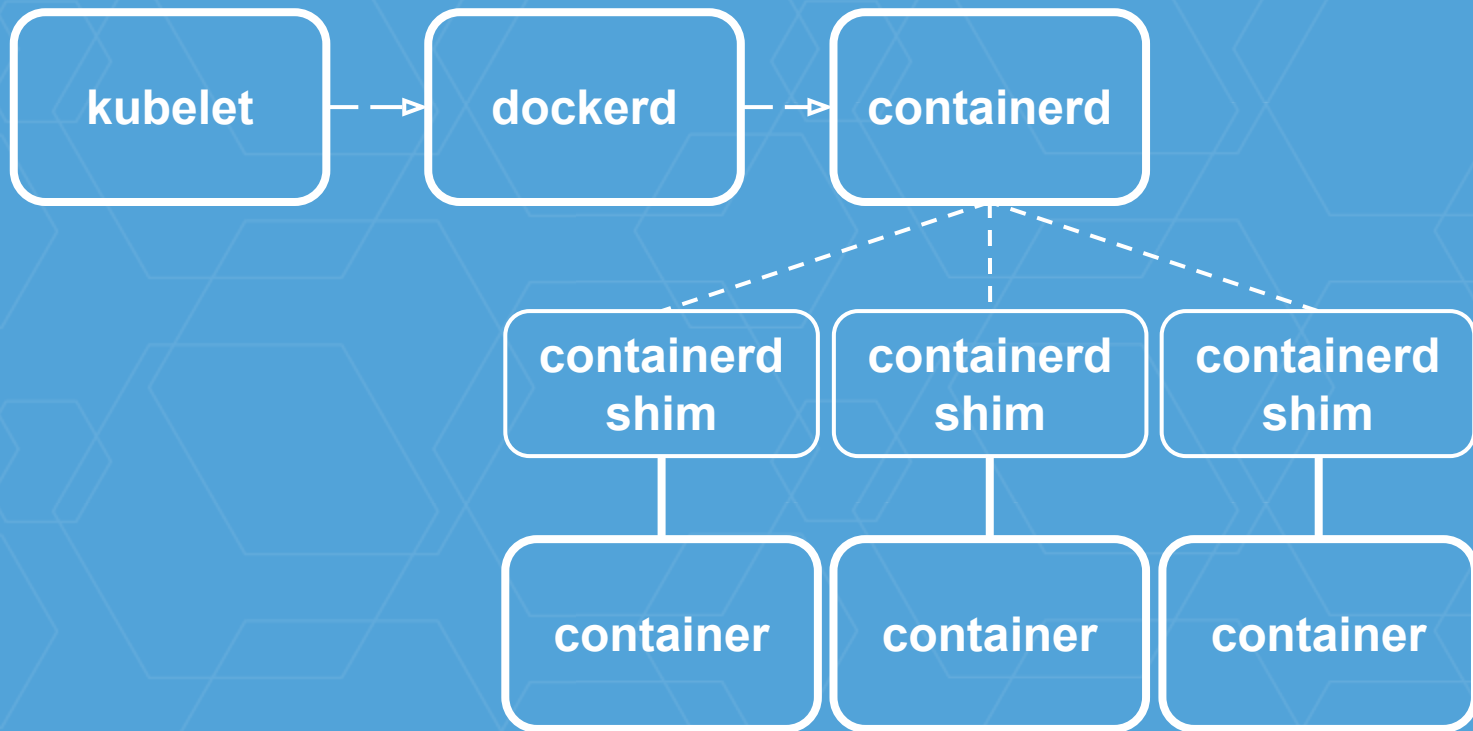
# Kubelet + Docker (before Docker 1.11)



# Kubelet + Docker (future)

- **Docker 1.11 introduced *containerd***
  - New daemon (outside of dockerd) to control container lifecycle
  - each Docker container is started and monitored by an individual "shim" process

# Kubelet + Docker (1.11+ with containerd)



# Kubelet + Docker (1.11+ with containerd)

- dockerd is no longer SPOF, but (for now) containerd is
- In future (1.12+), containerd will support persistence
  - *"Upgrade daemon without restarting containers"*  
<https://github.com/docker/docker/issues/2658>
- But...
  - per-container overhead, many moving parts
  - still have systemd integration issues

# Kubelet + rkt (rktnetes)

- Using rkt as the kubelet's container runtime
- *A pod-native* runtime
- First-class integration with systemd hosts
- self-contained pods process model = no SPOF
- Multi-image compatibility (e.g. docker2aci)
- Transparently swappable - no user impact

# How does it work?

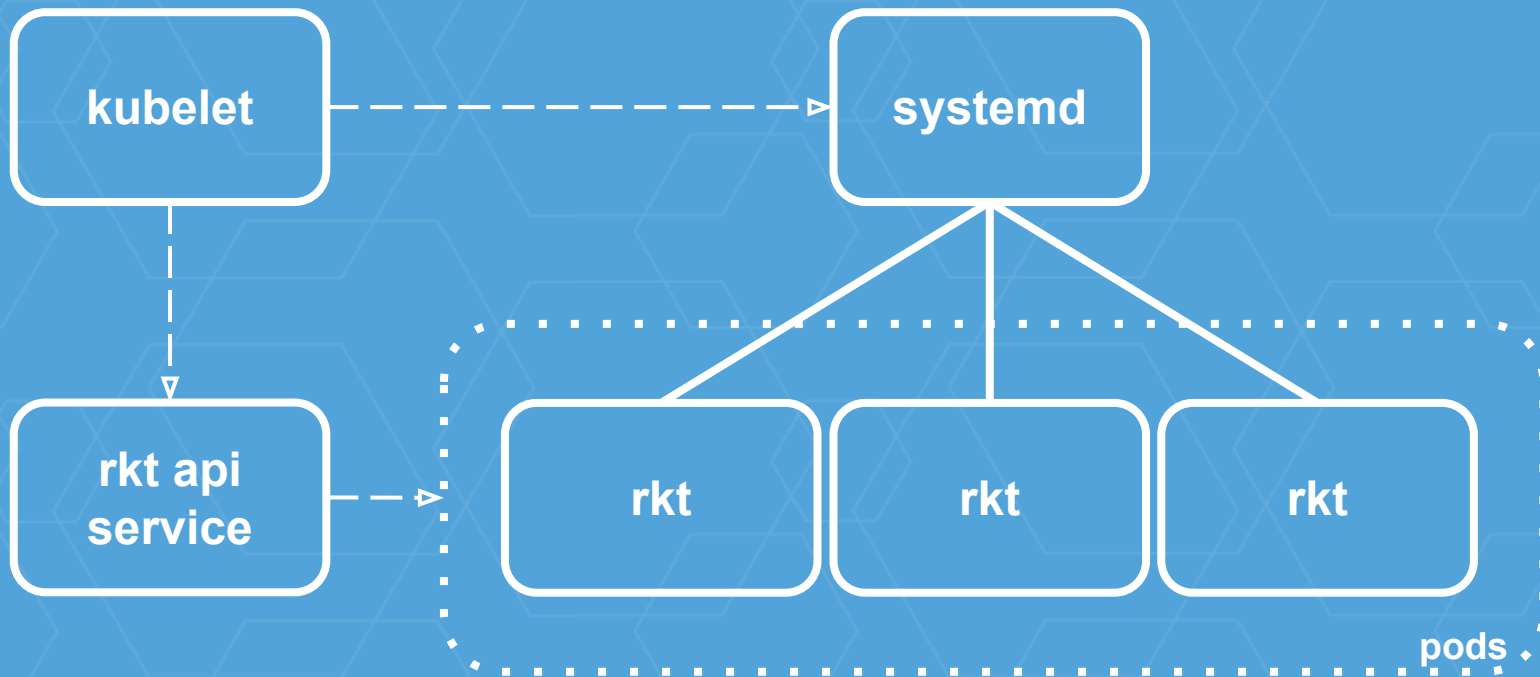
## **Pre-rktnetes:**

- Kubelet talks to the Docker daemon for all tasks

## **With rktnetes changes:**

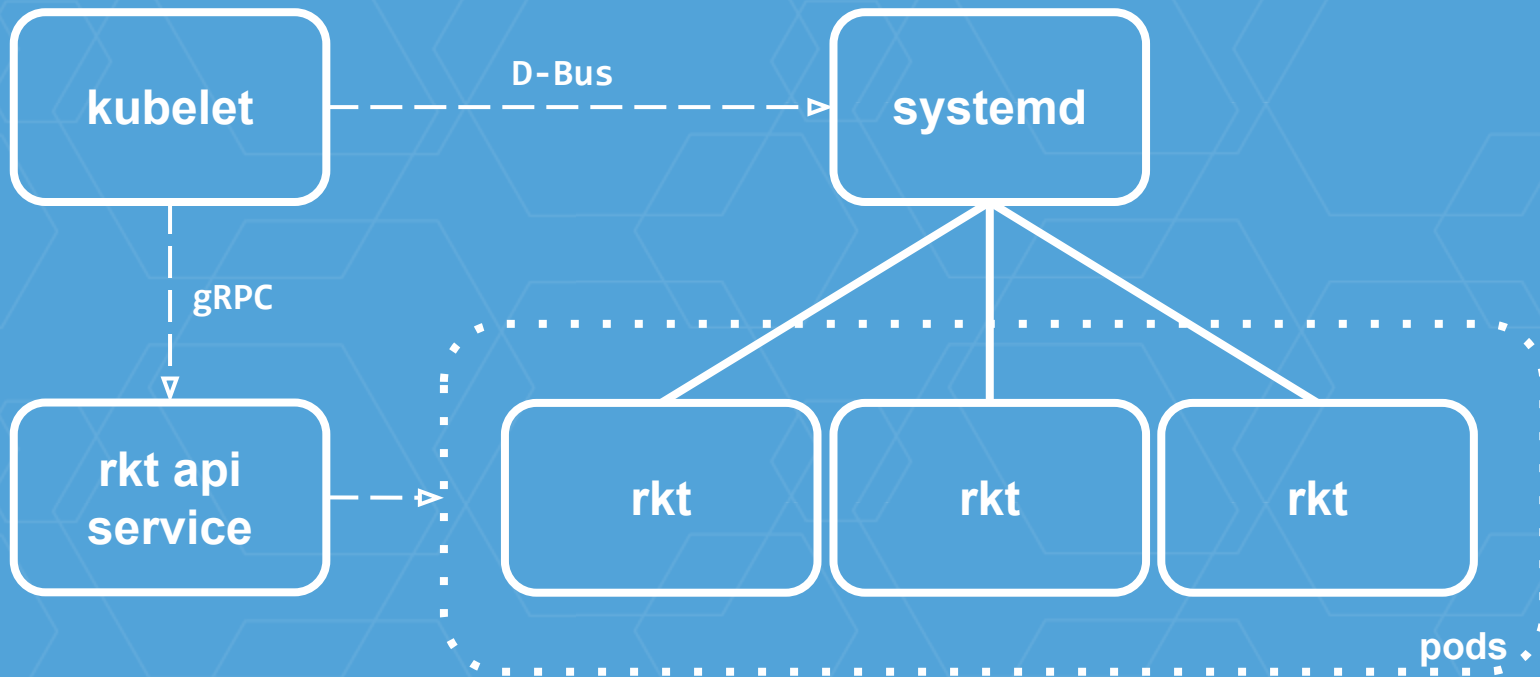
- Kubelet talks to the rkt API daemon for read-only tasks
- Kubelet execs rkt for preparatory tasks
- Kubelet talks to systemd for running pods via rkt

# Kubelet + rkt (rktnetes - with systemd)





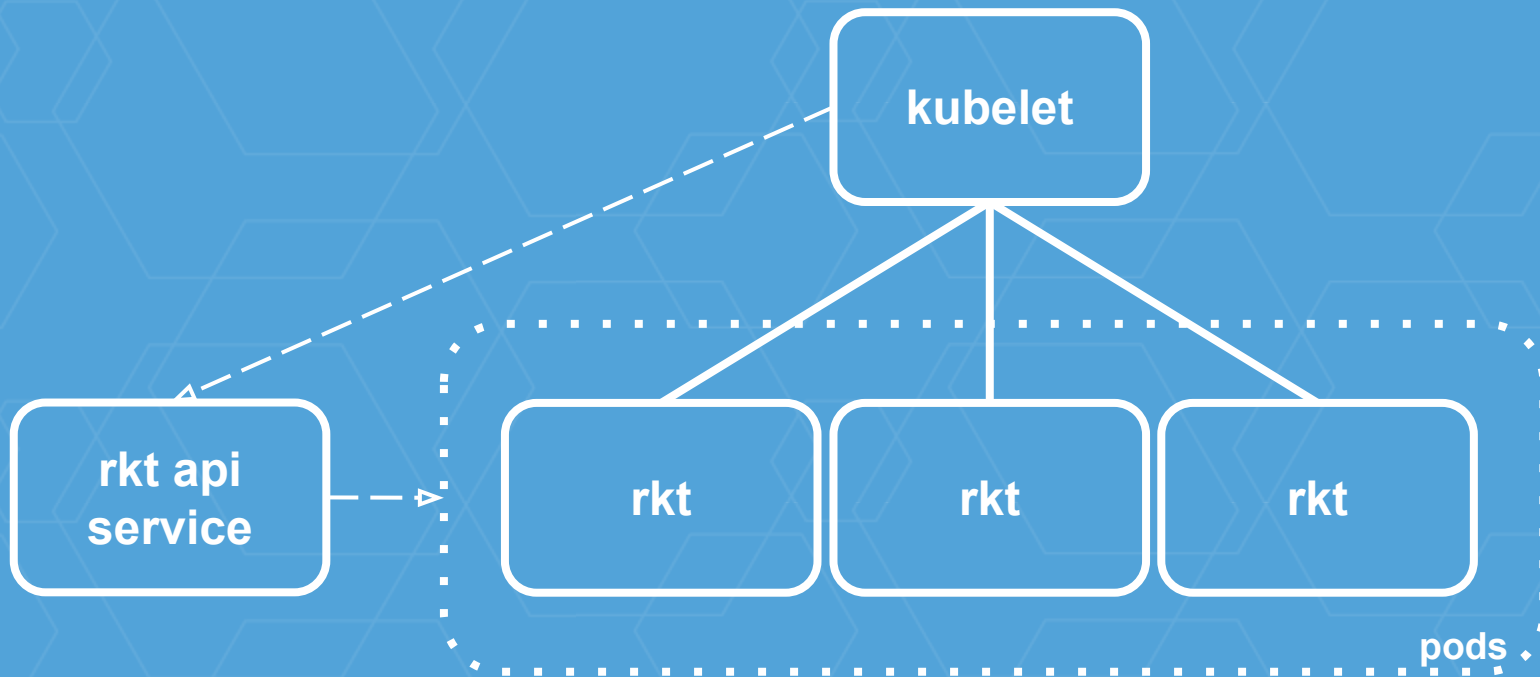
# Kubelet + rkt (rktnetes - with systemd)



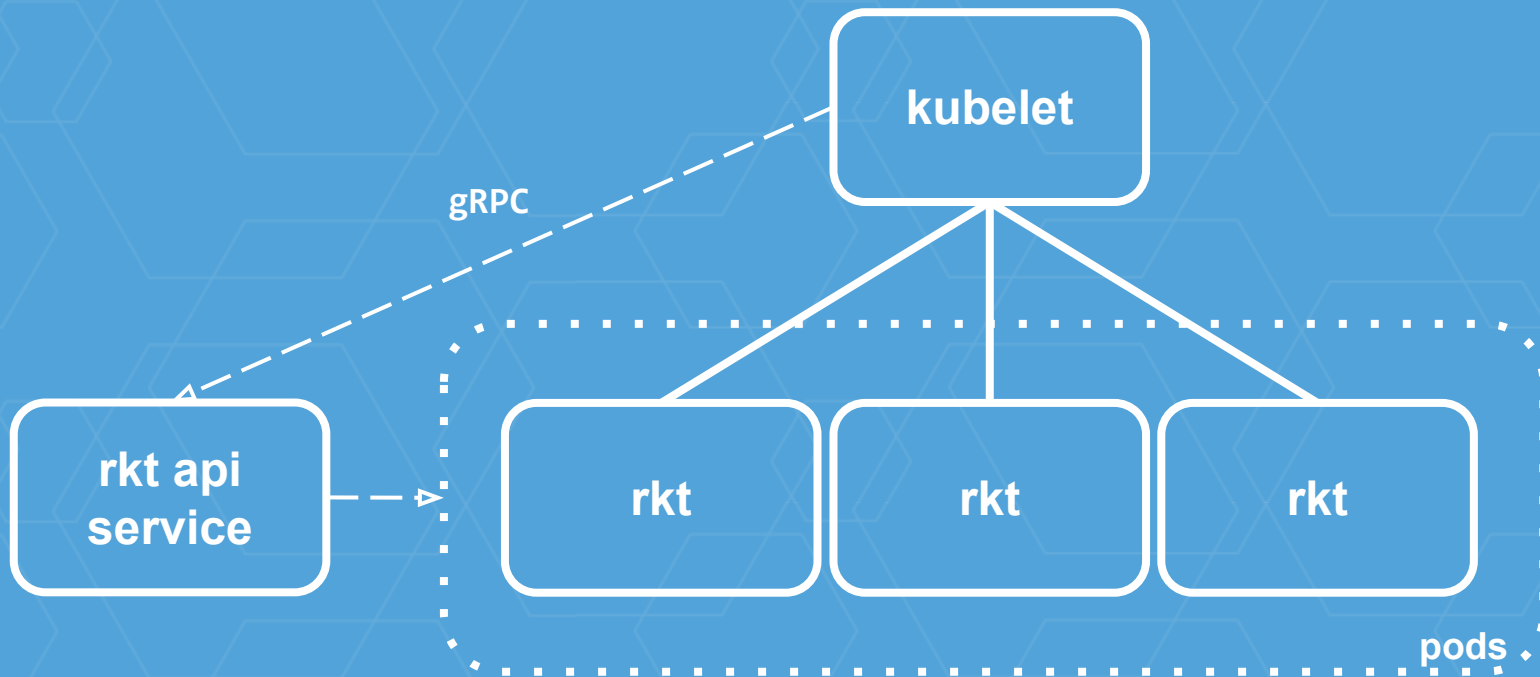
# What's the benefit in this?

- **No daemon running the containers**
  - live upgrades of the container runtime without affecting existing pods
- **Multiple stage1s provides more flexibility**
  - Swap in more advanced isolation technologies without needing to modify Kubernetes
- **Seamless integration with systemd**
  - machinectl, systemctl, journalctl Just Work™
  - Increasingly important as systemd adoption grows

# Kubelet + rkt (rktnetes - without systemd)



# Kubelet + rkt (rktnetes - without systemd)



# Kubelet + rkt (rktnetes - without systemd)

- **Benefits:**

- Kubelet can retain complete/granular control over process lifecycle of container runtime
- Remove one component from critical path (systemd)

- **Disadvantages:**

- Kubelet needs to be well-behaved process manager (and still be compatible with systemd)
- Kubelet is now SPOF for node

# rktnetes: *does it work?*

- **Yes!\***
  - where "yes" >90%, <100%
  - Currently passing >95% of common end-to-end tests, a few stragglers to go
- e2e test report:  
[http://storage.googleapis.com/kubernetes-test-history/static/suite-rktnetes\\$kubernetes-e2e-gce.html](http://storage.googleapis.com/kubernetes-test-history/static/suite-rktnetes$kubernetes-e2e-gce.html)
- Official release in Kubernetes 1.3, late June

# How can I use it?

- A getting started guide is in the Kubernetes docs: <http://kubernetes.io/docs/getting-started-guides/rkt/>
- Watch this space: <http://rktnetes.io>
- Fully supported, first-class backend in Kubernetes 1.3
  - Week of June 20

# What's coming with rkt and Kubernetes



# Coming soon in rkt

- New architecture support (ARM64)
  - <https://github.com/coreos/rkt/pull/2758>
- New hypervisor support (QEMU)
  - <https://github.com/coreos/rkt/pull/2684>

# Coming soon in rkt

- Unified cgroup hierarchy support
  - <https://github.com/coreos/rkt/issues/1757>
- Tighter privilege separation
  - Always drop euid when it's not needed
  - <https://github.com/coreos/rkt/issues/2482>

# Coming soon in rkt (on CoreOS)

- Running Kubelet using rkt
  - `/usr/lib64/coreos/kubelet-wrapper`
  - `$KUBELET_VERSION + rkt run (with fly)`
- Running Docker using rkt
  - Custom Docker versions using `docker-in-rkt`
  - <https://groups.google.com/d/msg/coreos-dev/icuel9OveRQ/0UliE43yAwAJ>
- Running everything using rkt!

# Coming soon in rkt and Kubernetes

- seccomp isolation
  - <https://github.com/kubernetes/kubernetes/pull/24602>
  - <https://github.com/coreos/rkt/pull/2753>
- sysctl support
  - <https://github.com/kubernetes/kubernetes/pull/26057>
  - <https://github.com/coreos/rkt/issues/2694>

# New container runtime interface

- k8s-sig-node wants to rework the interface between the kubelet and the container runtime
  - kubelet wants fine-grained control over containers
  - Move away from declarative monolithic function (SyncPod) to granular imperative operations (CreatePod, CreateContainerInPod, etc)
- Draft proposal up, targeted for Kubernetes 1.4+
  - <https://github.com/kubernetes/kubernetes/pull/25899>

# New container runtime interface

- Container-level operations: what about pods?!
  - rkt gets container level operations "for free" by leveraging systemd:
    - `journalctl -M <podname> -u <app>`
    - `systemctl -M <podname> start <app>`
  - Still retain benefits of first-class pods
- <https://github.com/coreos/rkt/issues/2375>
- <https://github.com/coreos/rkt/issues/1463>

bash/systemd/kubelet...

└─▶ rkt (stage0)

└─▶ pod (stage1)

└─▶ app1 (stage2)

└─▶ app2 (stage2)

rkt



systemd-nspawn

pod



systemd



application(s)





rkt



systemd-nspawn

pod



systemd



application(s)



systemd (on host)  
(systemctl)



rkt



systemd-nspawn



systemd



application(s)

pod



systemd (on host)  
(systemctl)

└─▶ rkt

└─▶ systemd-nspawn

systemd-machined  
(machinedctl)



systemd (on host)  
(systemctl)

└─▶ rkt

└─▶ systemd-nspawn

register



systemd-machined  
(machinedctl)



systemd

application(s)

systemd (on host)  
(systemctl)

└─▶ rkt

└─▶ systemd-nspawn

systemd-machined  
(machinedctl)

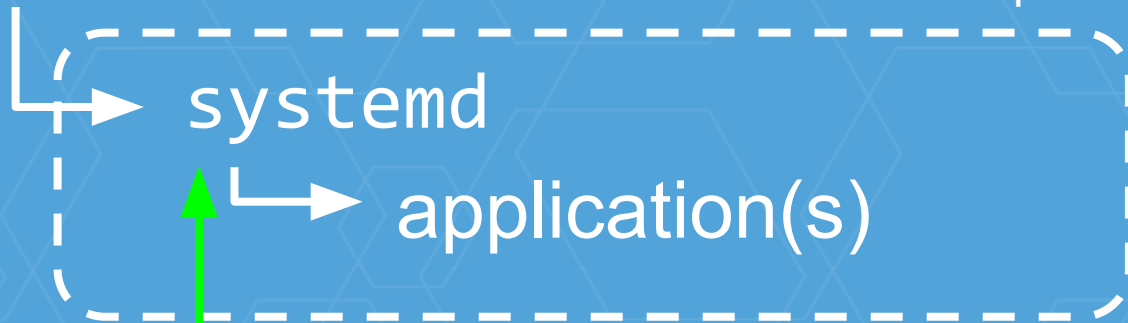


systemd (on host)  
(systemctl)

└─▶ rkt

└─▶ systemd-nspawn

systemd-machined  
(machinedctl)



kubelet?

control

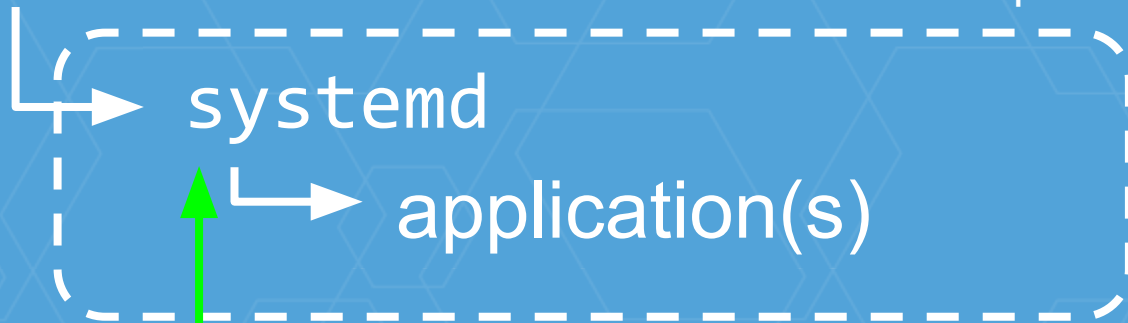


systemd (on host)  
(systemctl)

└─▶ rkt

└─▶ systemd-nspawn

systemd-machined  
(machinedctl)



kubelet?

restart application, add new application, ...

# New\* container image format

- **OCI: a container image format we can *all* agree on**
  - Based on Docker v2.2 image format (\*not really "new")
  - Adding optional components like signing and naming
  - Active work on tooling (octool)
  - First, reach a 1.0: then, push this image format into the Kubernetes API



# How do I find out more?

- **Reach out on GitHub or IRC**
  - [github.com/coreos/rkt](https://github.com/coreos/rkt)
  - #rkt-dev / #rkt on Freenode
- **Join the Kubernetes "Node" special interest group**
  - <https://groups.google.com/forum/#!forum/kubernetes-sig-node>
- **Join us!**
  - Hiring rkt developers in Berlin

# CoreOS is Running the World's Containers

## OPEN SOURCE

90+ Projects on GitHub, 1,000+ Contributors

CoreOS.com - @coreoslinux - github/coreos



## ENTERPRISE

Secure solutions, support plans, training + more

sales@coreos.com - tectonic.com - quay.io



**We're hiring in all departments! Email: [careers@coreos.com](mailto:careers@coreos.com) Positions: [coreos.com/careers](https://coreos.com/careers)**