rktnetes

Integrating a new container runtime with Kubernetes

Jonathan Boulle

github.com/jonboulle - @baronboulle

CoreOS

What is rkt?

A CLI for running app containers on Linux.

Focuses on:

- Security
- Composability
- Standards/Compatibility

What is an app container?

What is an app container?

- To rkt, an app container is a *pod*
  - Fundamental execution unit in rkt
- Grouping of one or more applications in a *shared execution context* (network, isolation, ...)
- rkt pod ≈ Kubernetes pod

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* (production)**
  - Runtime stability + interface guarantees
- < ... many more ... >
- **September 2016 - *v1.14.0***
  - Latest stable release

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
  - Restrict capabilities by default
- **Architecture: *Unix philosophy***
  - Well-defined operational scope
  - Clean integration points as a classic Unix process
  - 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 (cont.)

Classic and modern Linux technologies

- Fine-grained Linux capabilities
  - only let containers do what they *need* to do
- seccomp enabled by default
  - restrict application access to kernel
- Mask sensitive `/proc` and `/sys` paths

Security will never be "complete"; always an iterative process, refining over time

A CLI for running app containers on Linux.

Focuses on:

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

How rkt does composability

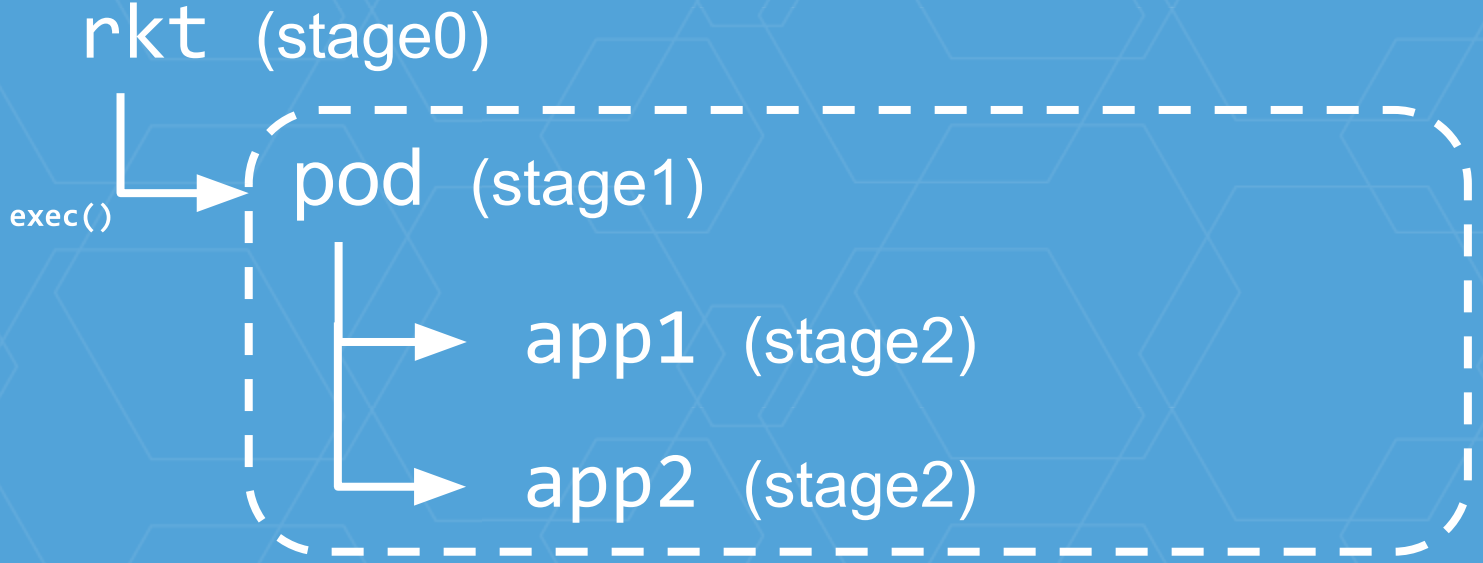
- **"*External*" composability**
  - Unix architecture; integrating well with other tools (init systems, orchestration tools) is a priority

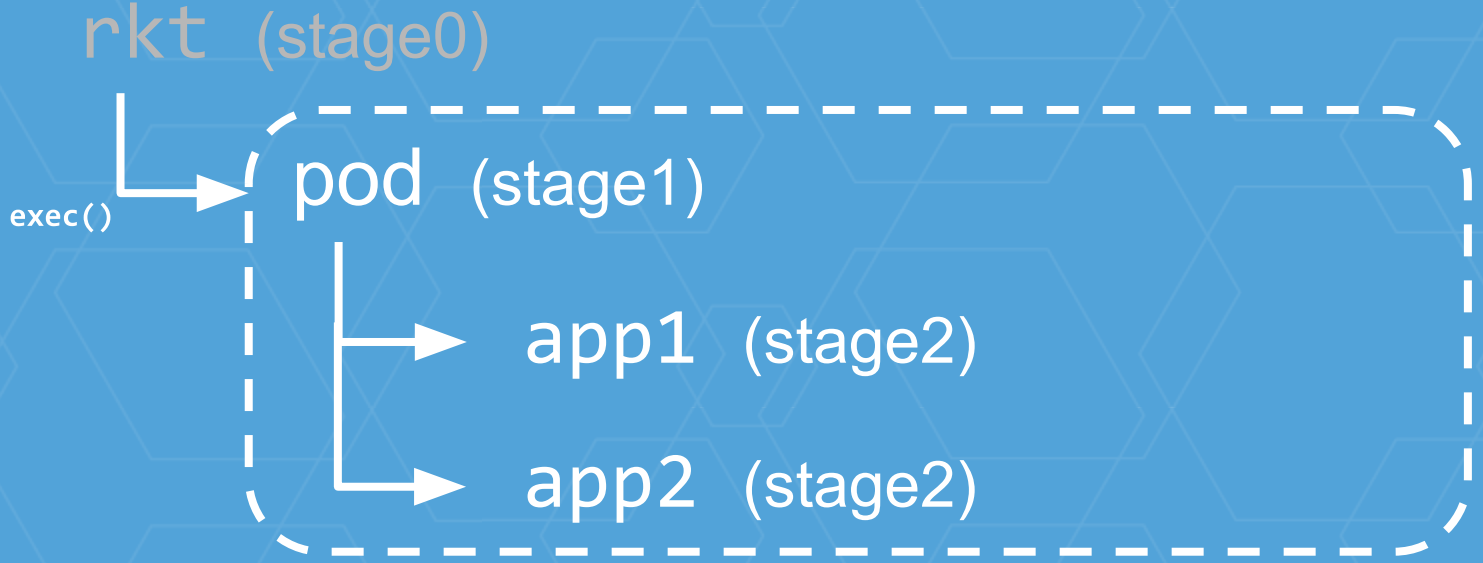
- **"*Internal*" composability**
  - Swappable execution engines (stage-based architecture) that actually runs the container

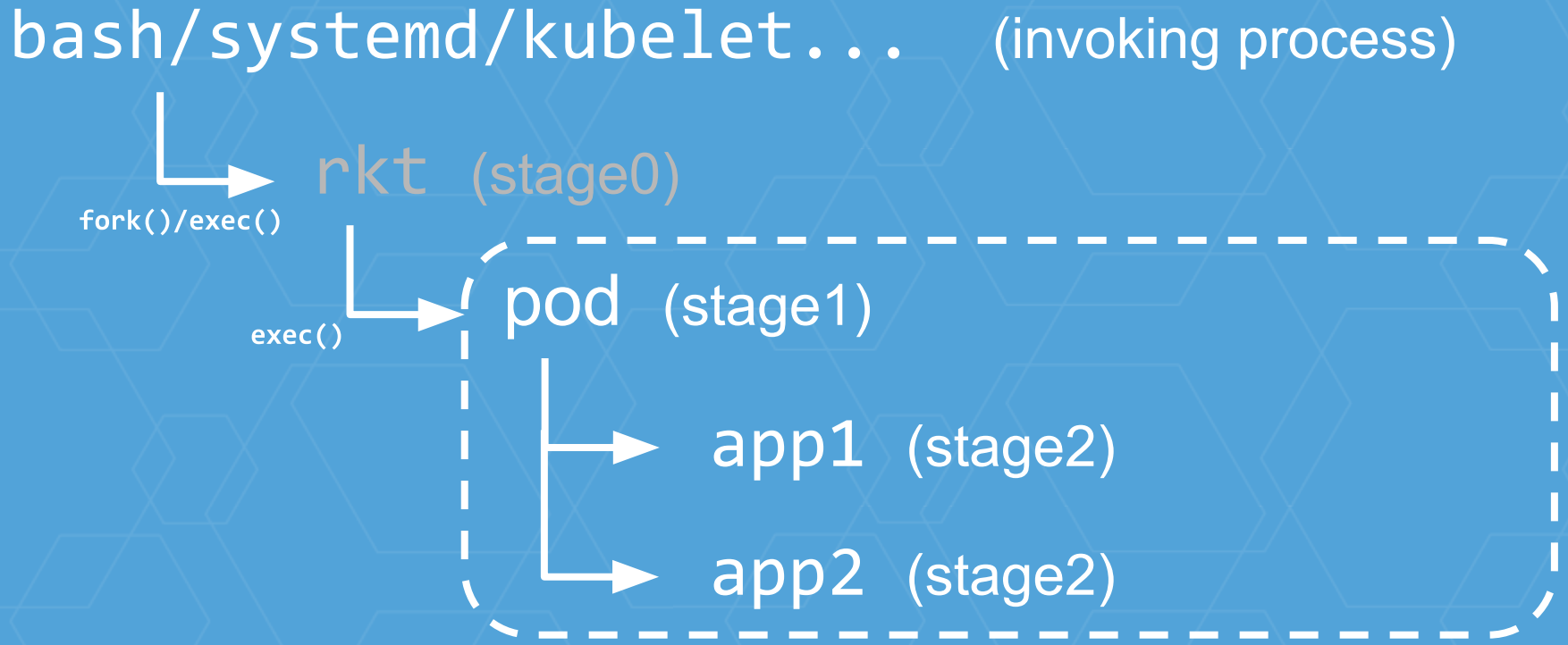
How rkt does composability

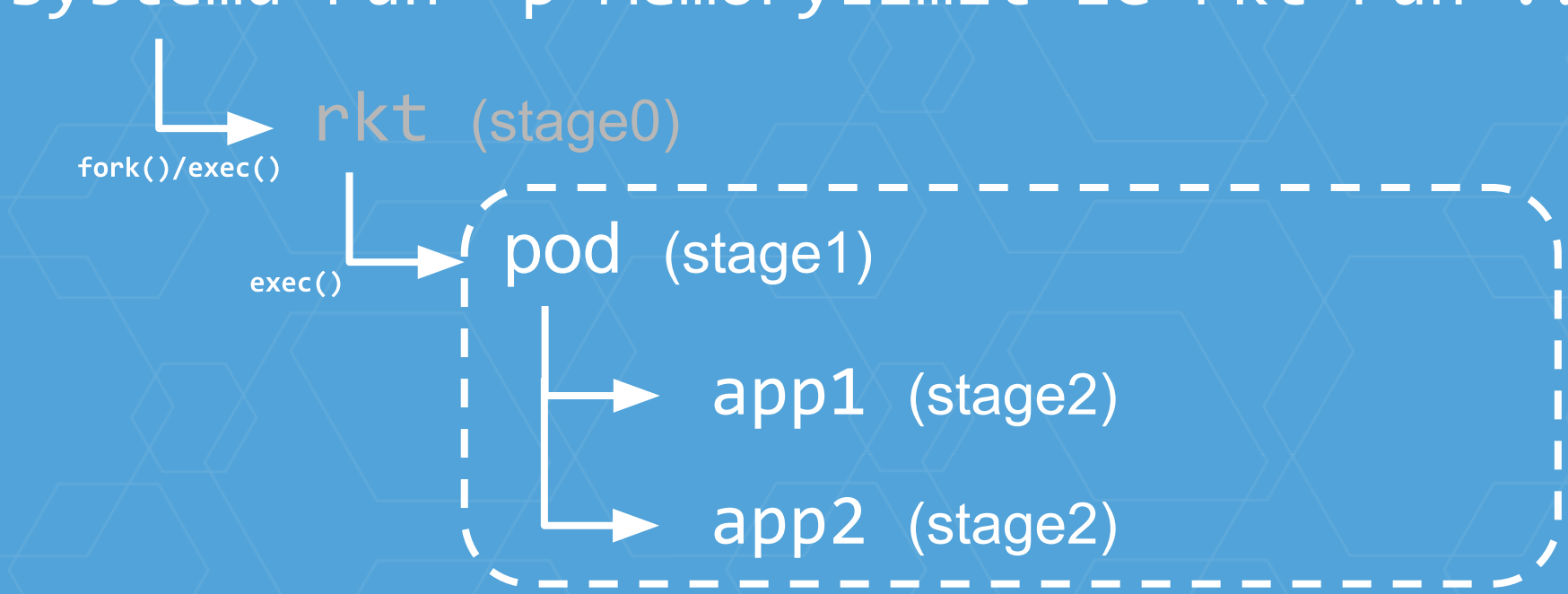
- **"*External*" composability**
  - Simple process model: a single rkt process *is* a pod
  - Any context applied to rkt (cgroups, etc) applies transitively to the pod and the apps inside
  - No mandatory daemon, but optional gRPC (HTTP2+Protobuf) API server to facilitate more efficient introspection
  - Pod-level and app-level properties


pod
app1
app2

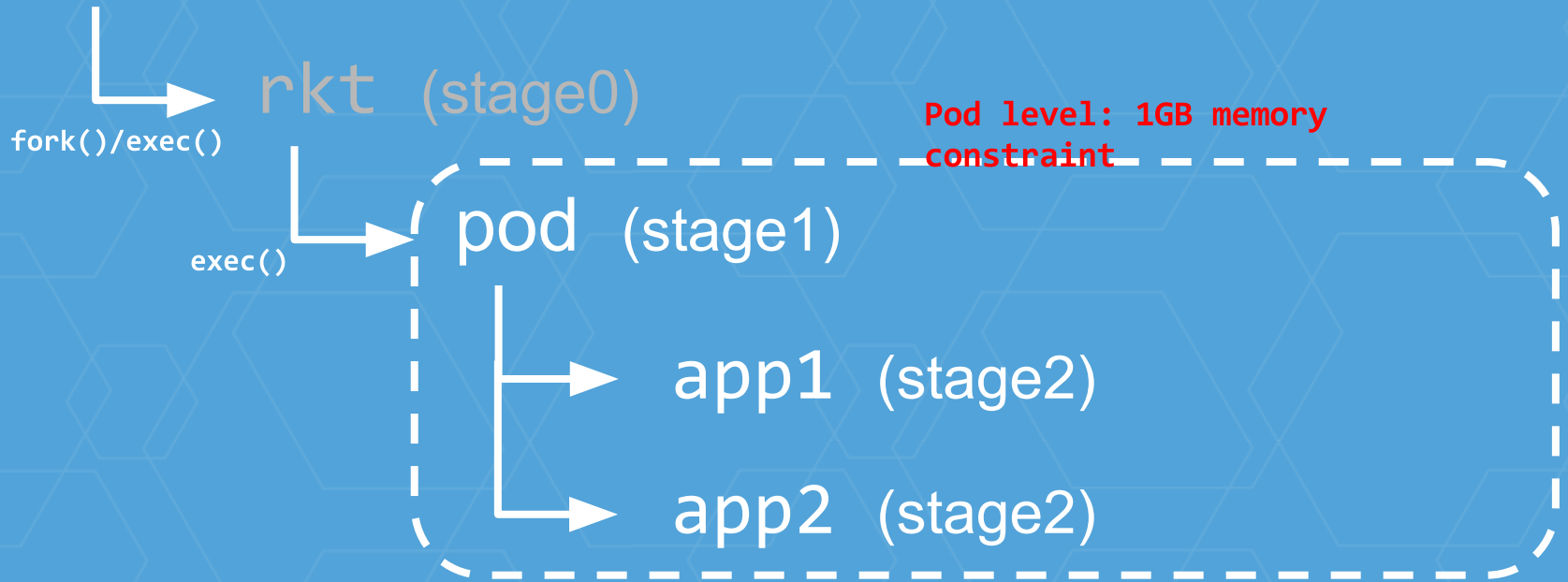

rkt (stage0)
exec()
pod (stage1)
app1 (stage2)
app2 (stage2)

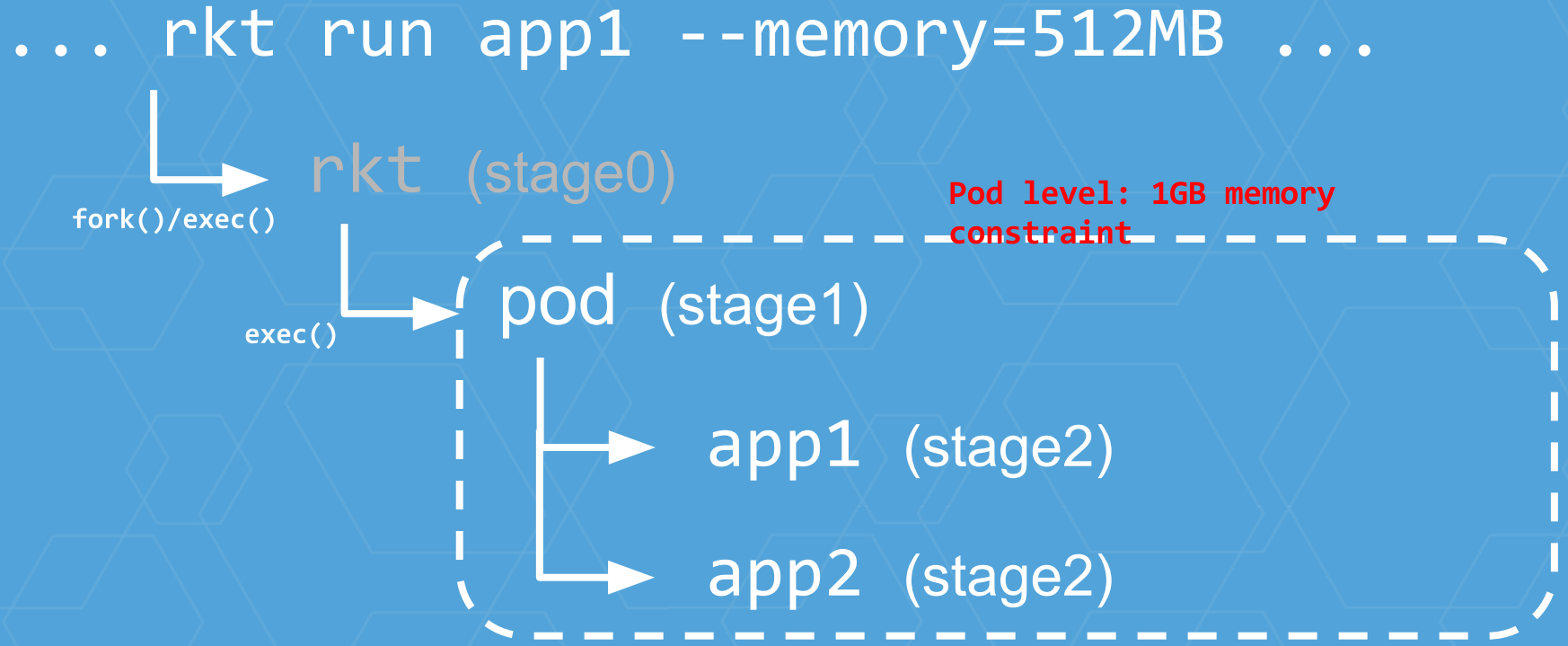

rkt (stage0)
exec()
pod (stage1)
app1 (stage2)
app2 (stage2)

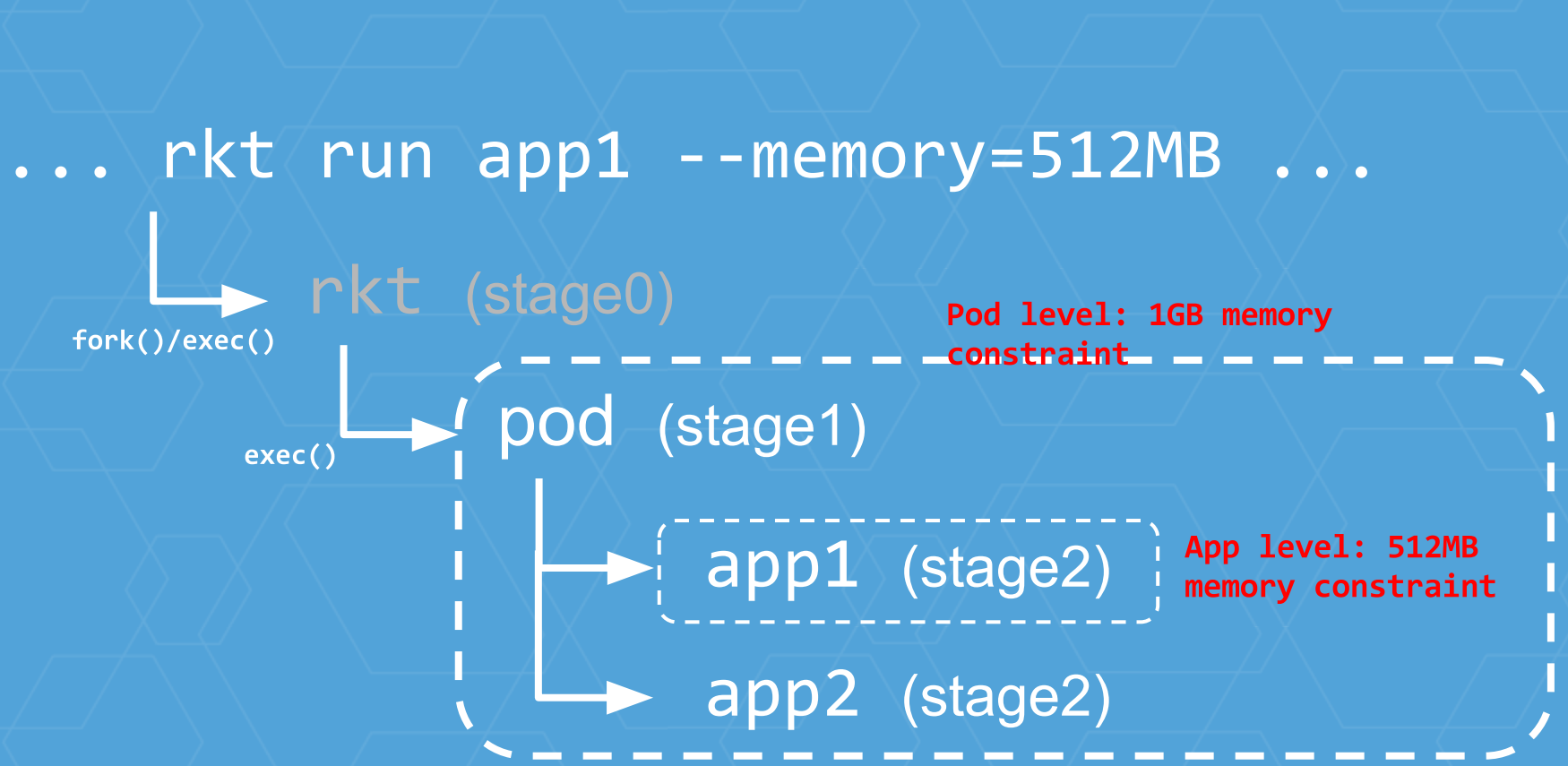

bash/systemd/kubelet...
(invoking process)
fork()/exec()
rkt (stage0)
exec()
pod (stage1)
app1 (stage2)
app2 (stage2)


systemd-run -p MemoryLimit=1G rkt run ...
fork()/exec()
rkt (stage0)
exec()
pod (stage1)
app1 (stage2)
app2 (stage2)

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

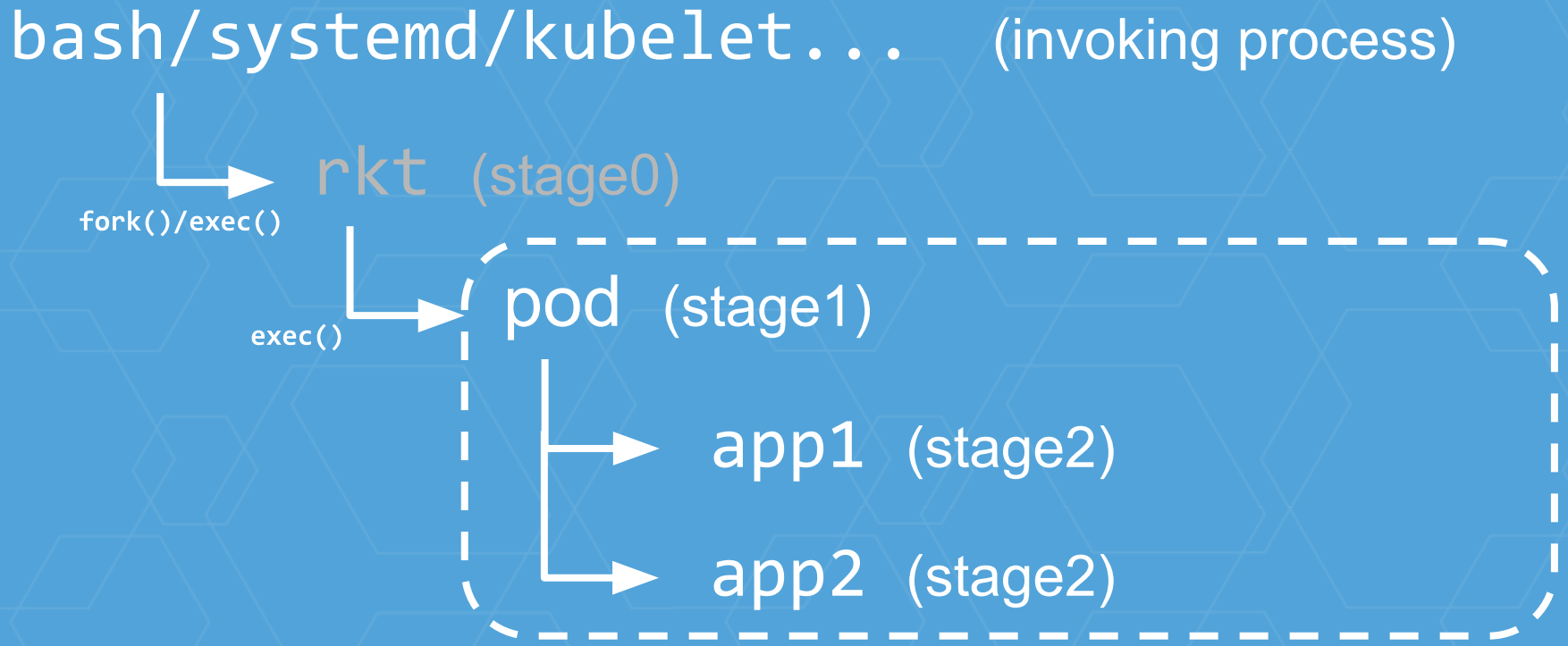


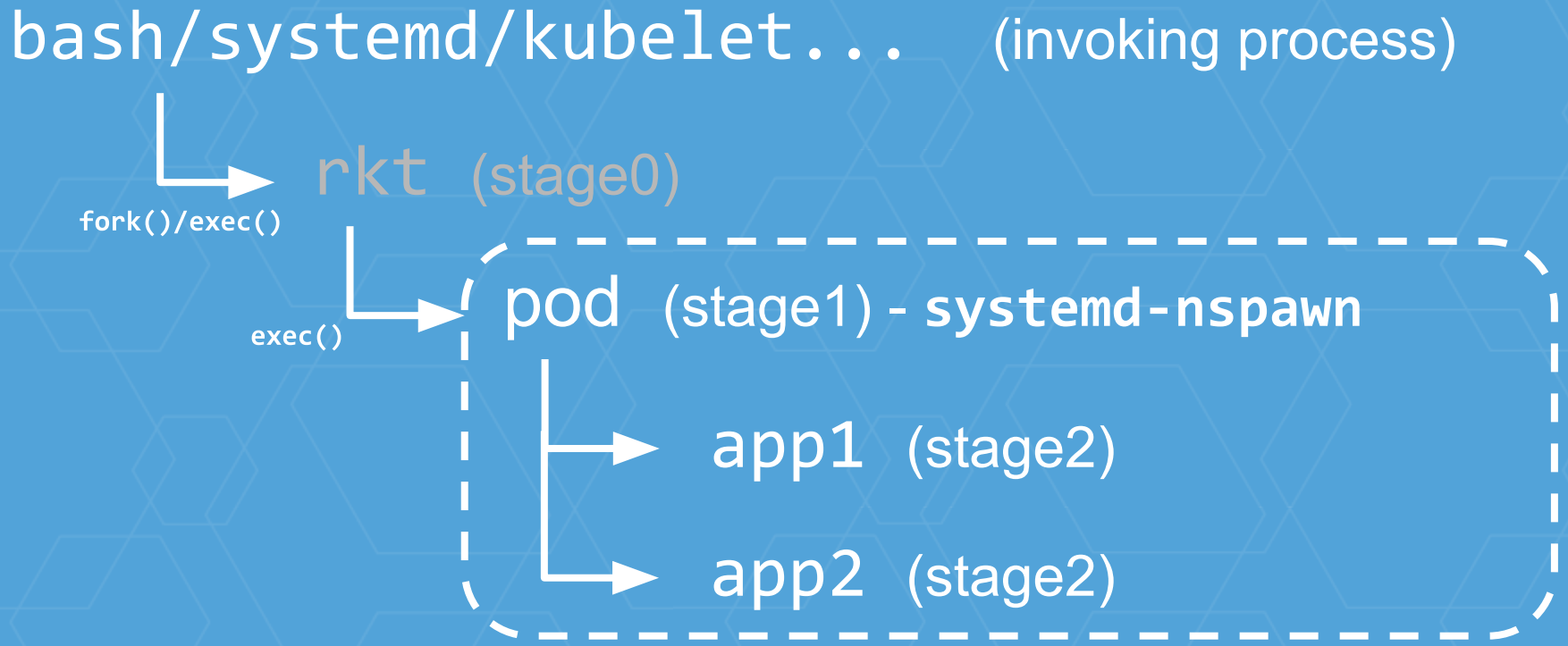

... rkt run app1 --memory=512MB ...
fork()/exec()
rkt (stage0)
Pod level: 1GB memory constraint
exec()
pod (stage1)
app1 (stage2)
app2 (stage2)

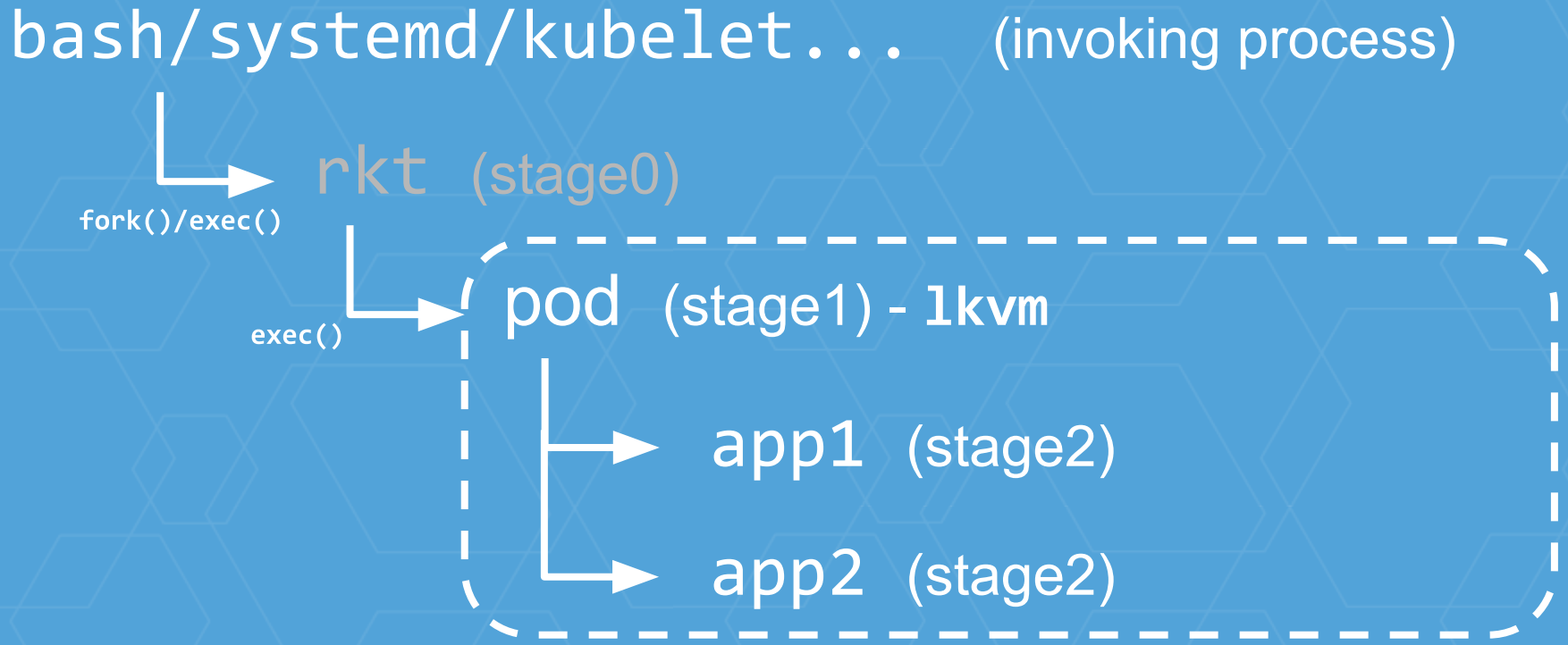

... rkt run app1 --memory=512MB ...
fork()/exec()
rkt (stage0)
Pod level: 1GB memory constraint
exec()
pod (stage1)
app1 (stage2)
App level: 512MB memory constraint
app2 (stage2)

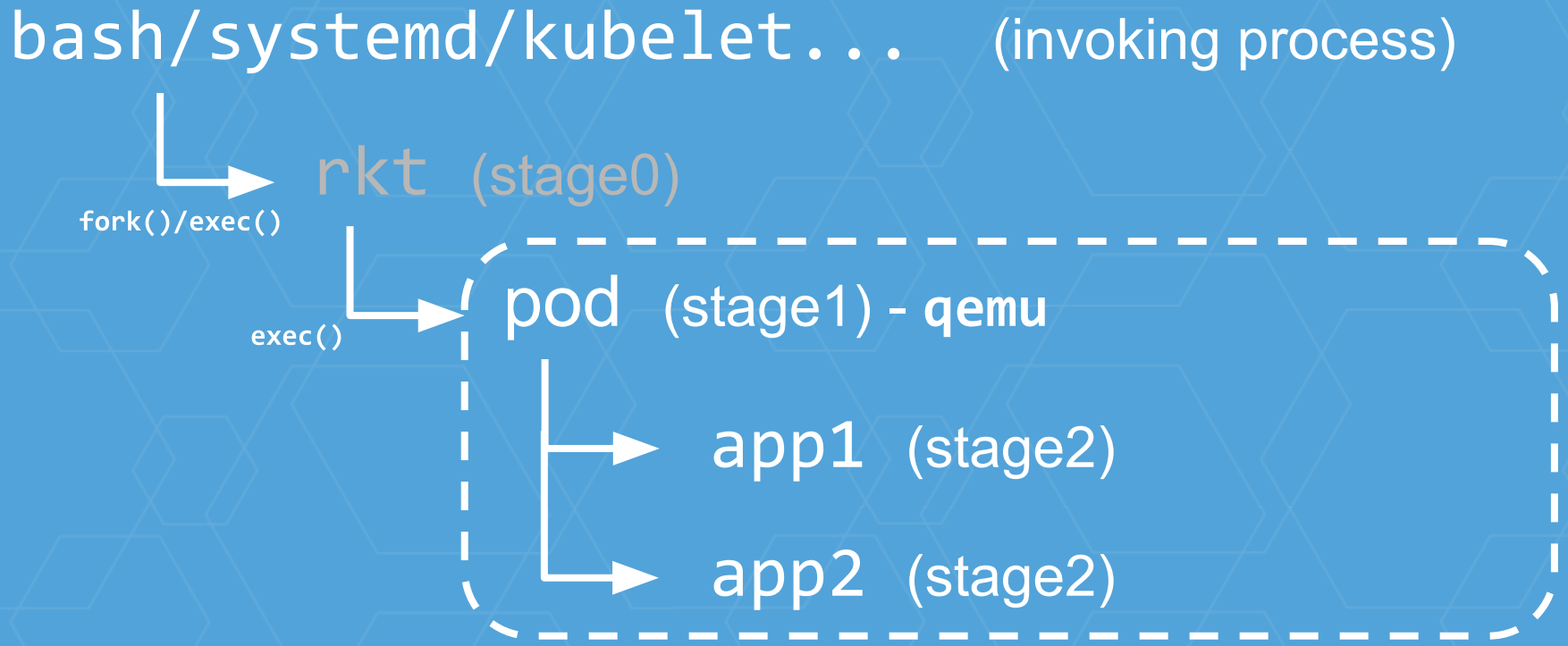
How rkt does composability

- **"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 (Real Soon Now™)

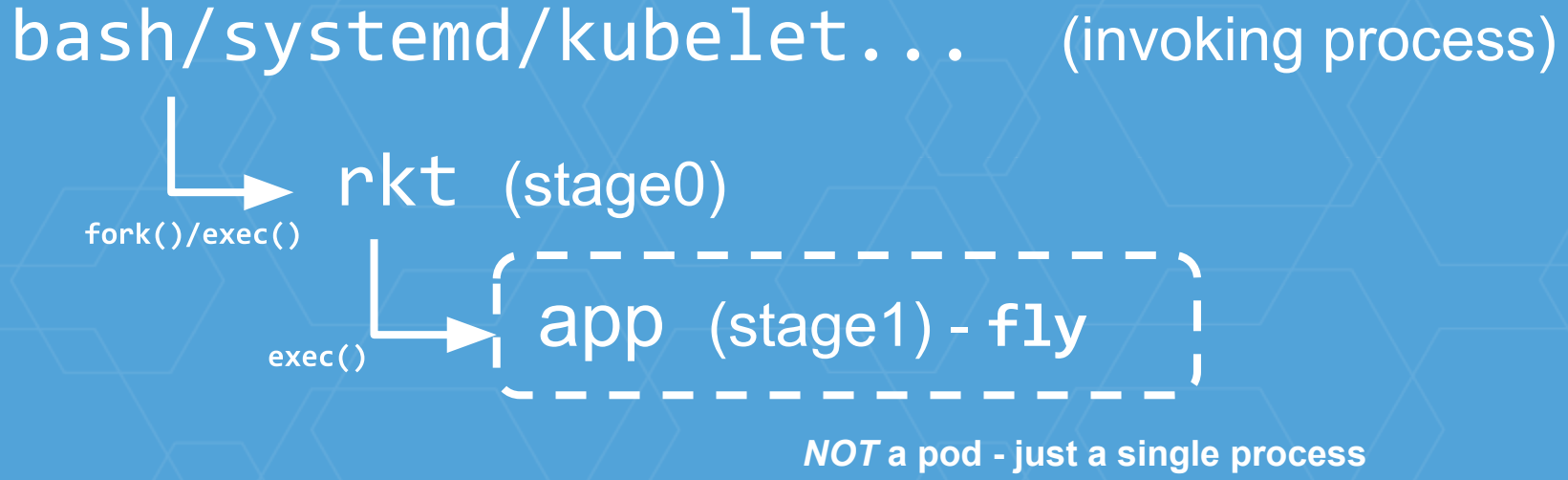

bash/systemd/kubelet...
(invoking process)
fork()/exec()
rkt (stage0)
exec()
pod (stage1)
app1 (stage2)
app2 (stage2)


bash/systemd/kubelet...
(invoking process)
fork()/exec()
rkt (stage0)
exec()
pod (stage1) - systemd-nspawn
app1 (stage2)
app2 (stage2)


bash/systemd/kubelet...
(invoking process)
fork()/exec()
rkt (stage0)
exec()
pod (stage1) - lkvm
app1 (stage2)
app2 (stage2)


bash/systemd/kubelet...
(invoking process)
fork()/exec()
rkt (stage0)
exec()
pod (stage1) - qemu
app1 (stage2)
app2 (stage2)


bash/systemd/kubelet...
(invoking process)
fork()/exec()
rkt (stage0)
exec()
app (stage1) - fly



***NOT* a pod - just a single process**

A CLI for running app containers on Linux.

Focuses on:

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

How rkt does standards/compatibility

- Started as an implementation of **appc**
  - first attempt at a well-defined container spec
- Networking plugin system became **CNI**
  - 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, CNI, OCI, CNCF
  - rkt will be fully OCI compliant

A brief standards history

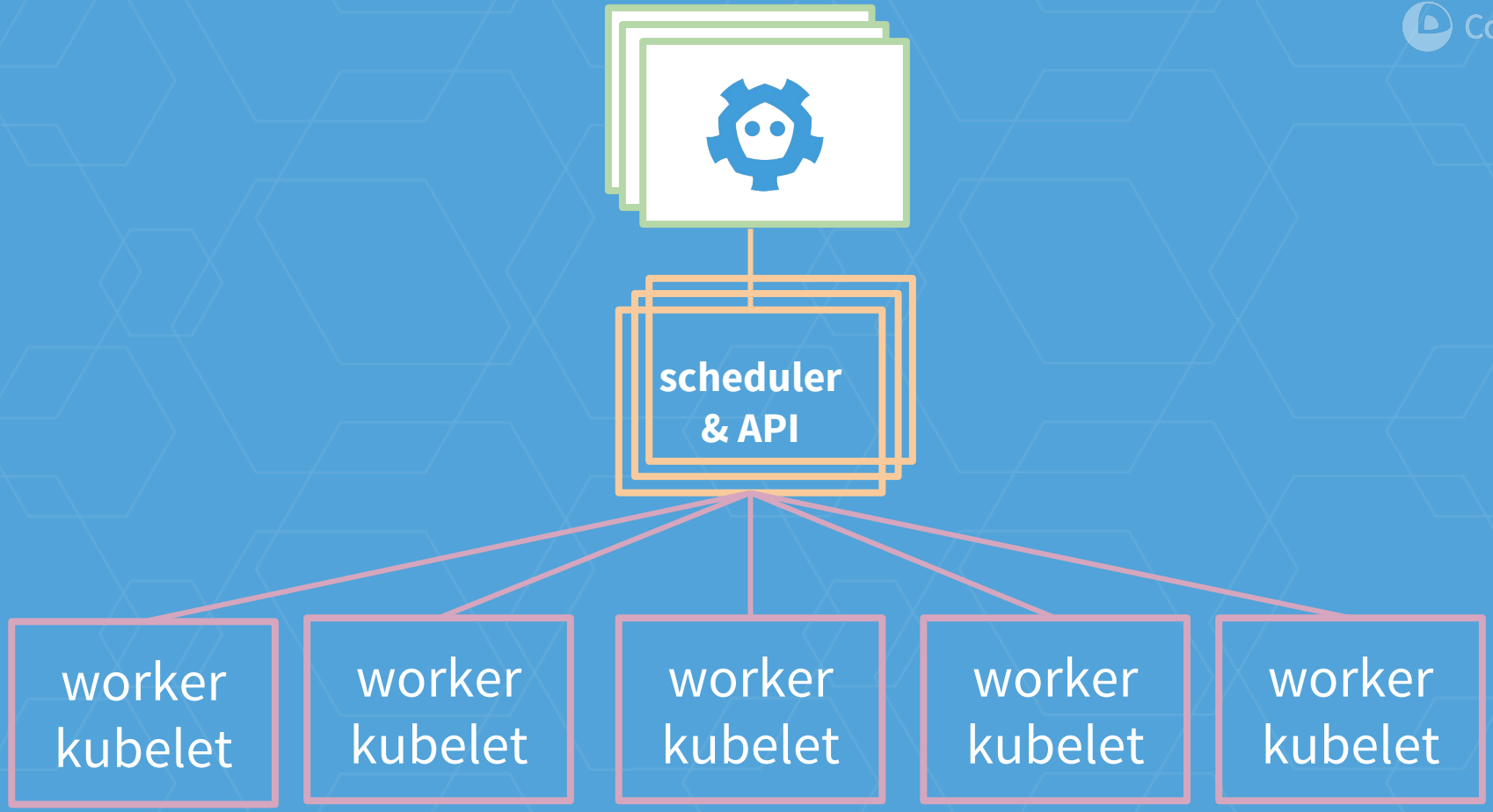
- appc (December 2014)
  - container images, runtime environment, and pods
  - some adoption, but (intended to be) deprecated in favour of
- OCI (June 2015)
  - initially runtime only, now container images too
- CNCF (December 2015)
  - "harmonising cloud-native technologies"

Kubernetes

Cluster-level container orchestration.

Handles:

- Scheduling/Upgrades
- Failure recovery
- Scaling


scheduler
& API
worker
kubelet
worker
kubelet
worker
kubelet
worker
kubelet
worker
kubelet

Kubernetes

Cluster-level **container** orchestration.

Handles:

- Scheduling/Upgrades
- Failure recovery
- Scaling

Kubernetes

Cluster-level **container** orchestration.

- Historically, 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 Kubernetes contain?

- **kubelet** is the daemon that runs on every worker node in a Kubernetes cluster
- kubelet runs the *pods* scheduled to it by Kubernetes
- kubelet delegates to container runtime to perform all container-related operations

How does Kubernetes contain?

Somewhere deep inside the Kubelet..

```
// Runtime interface defines the interfaces that
// should be implemented by a container runtime.
type Runtime interface {
...
    SyncPod(pod *api.Pod, ...)
    GetPods() ([]*Pod, error)
    KillPod(pod *api.Pod, ...)
...
```

How does Kubernetes contain?

- Want to add a new Container runtime? Just implement the interface!

How does Kubernetes contain?

- Want to add a new Container runtime? Just implement the interface!
  - ... and refactor the kubelet heavily to remove Dockerisms
  - One year and 200+ commits later...

Kubernetes + rkt = rktnetes

Have Kubernetes use rkt as
the container runtime.

rkt handles:

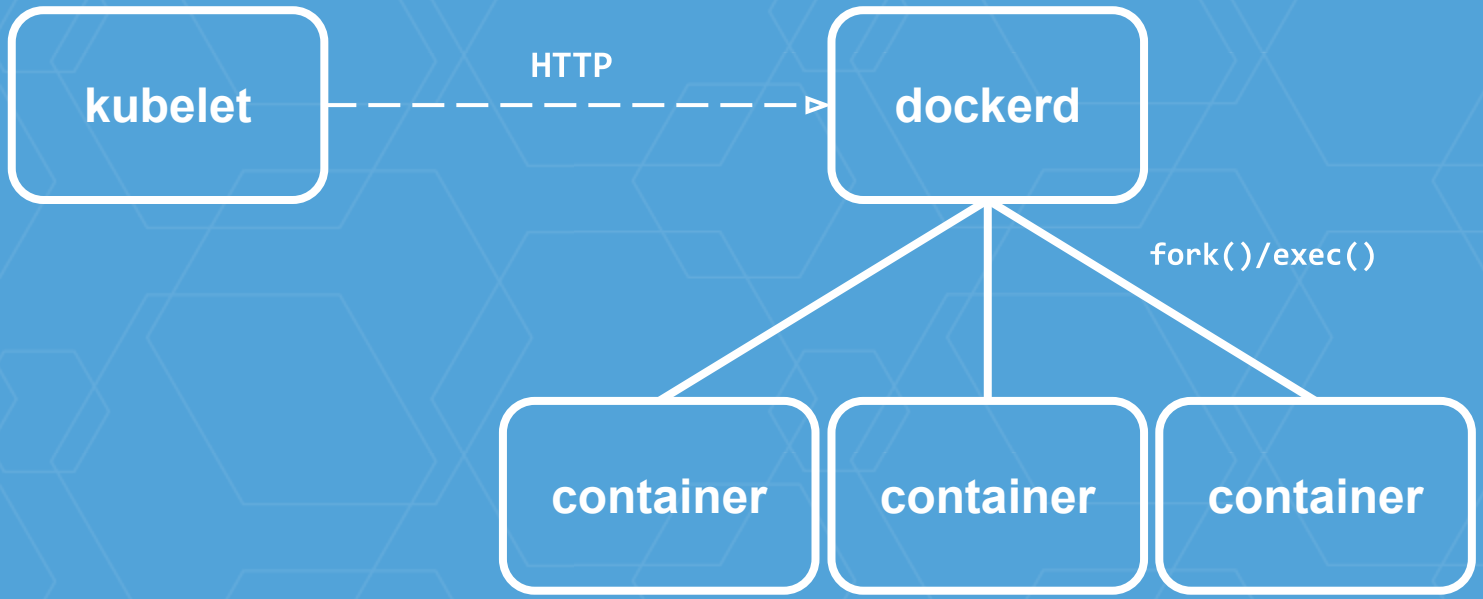
- Image discovery
- Image fetching
- Pod execution

How does it work?

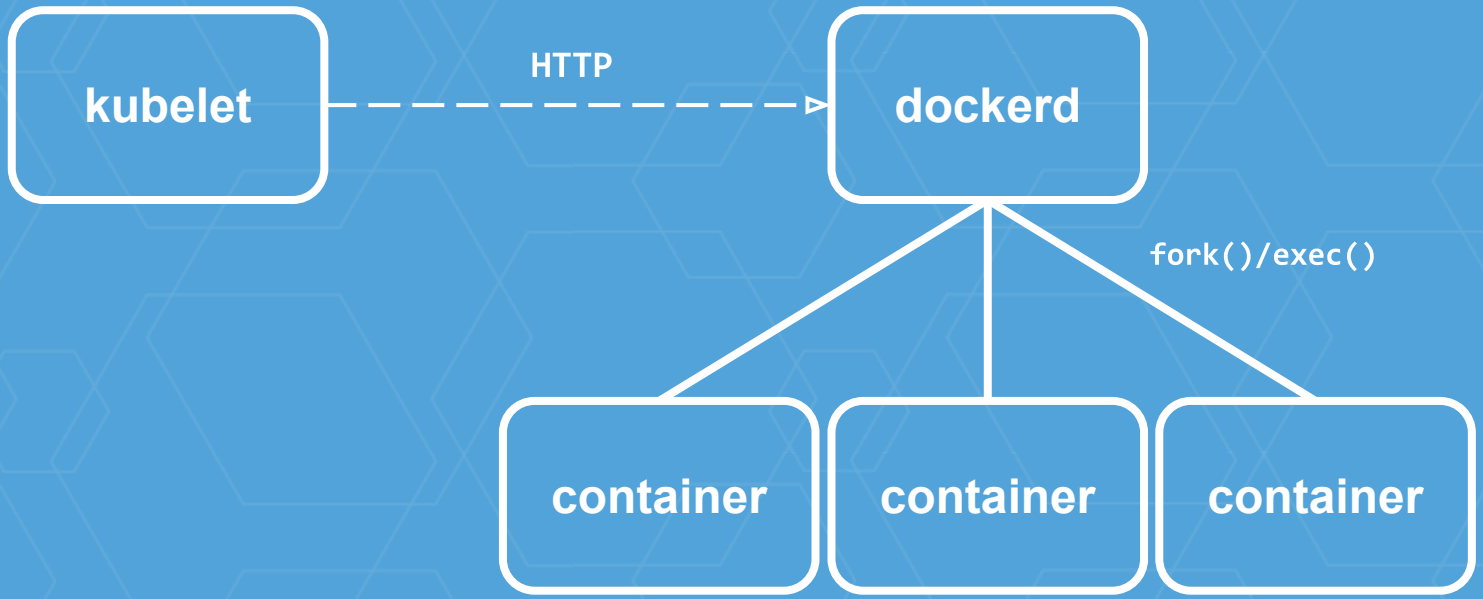
Pre-rktnetes (current default):

- Kubelet talks to the ***Docker daemon*** for all tasks

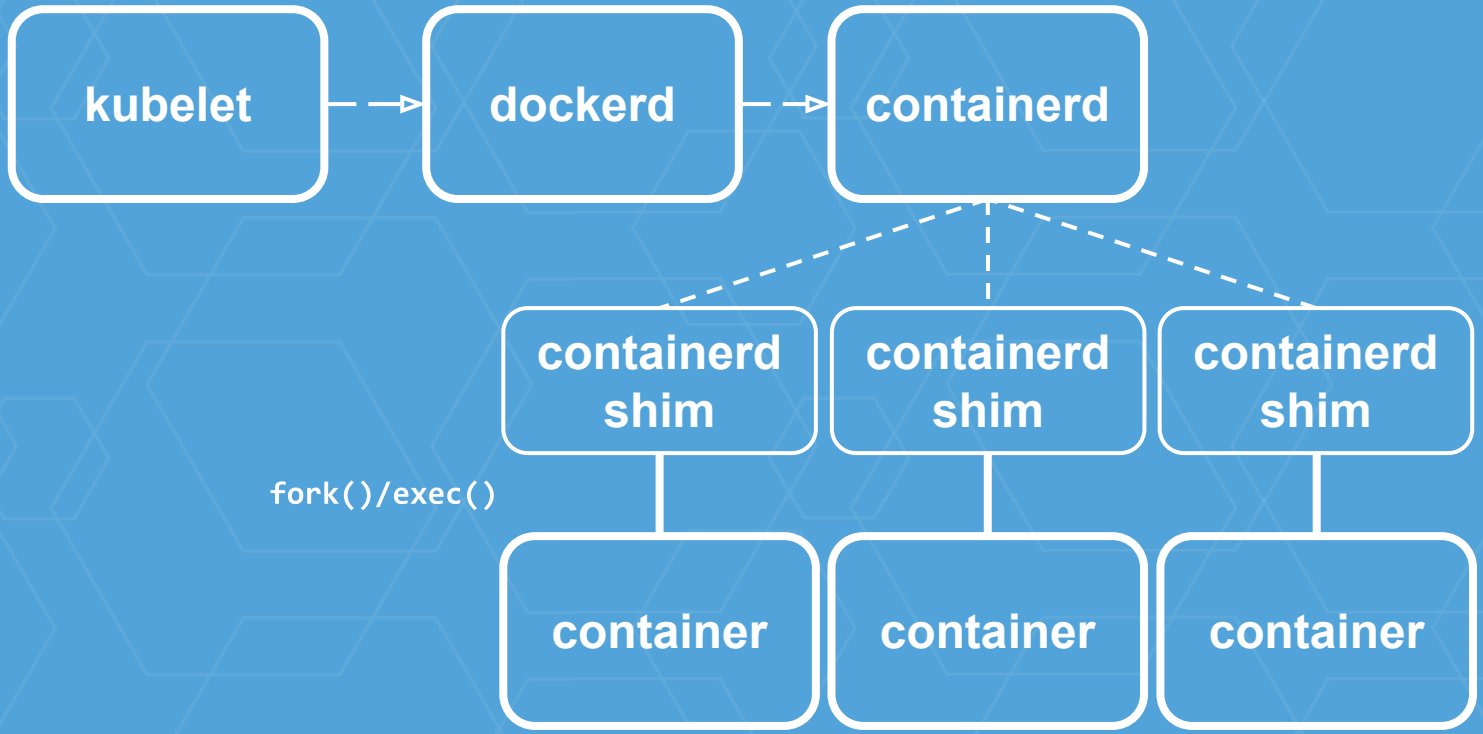
Kubelet + Docker (default)



Kubelet + Docker (before Docker 1.11)



Kubelet + Docker (1.11+ with containerd)



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?

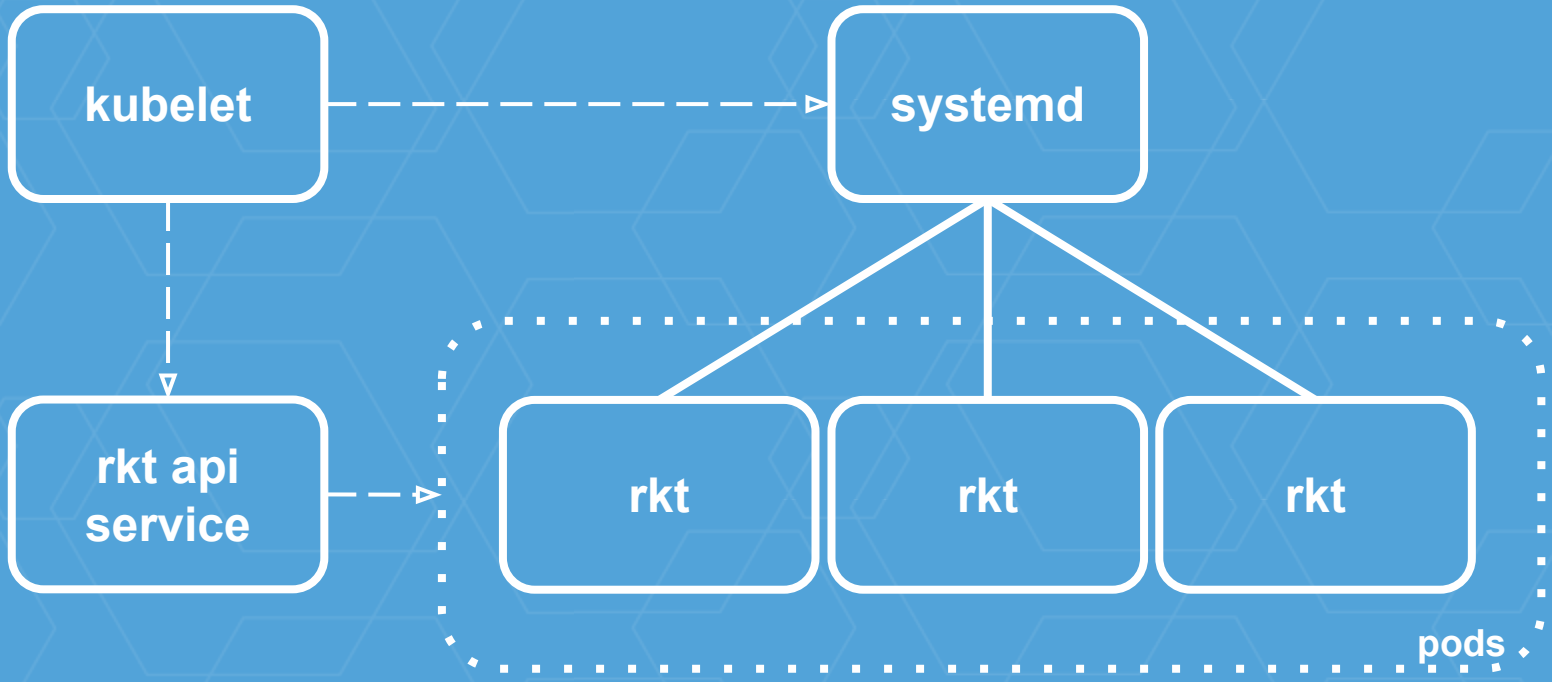
Pre-rktnetes (current default):

- Kubelet talks to the ***Docker daemon*** for all tasks

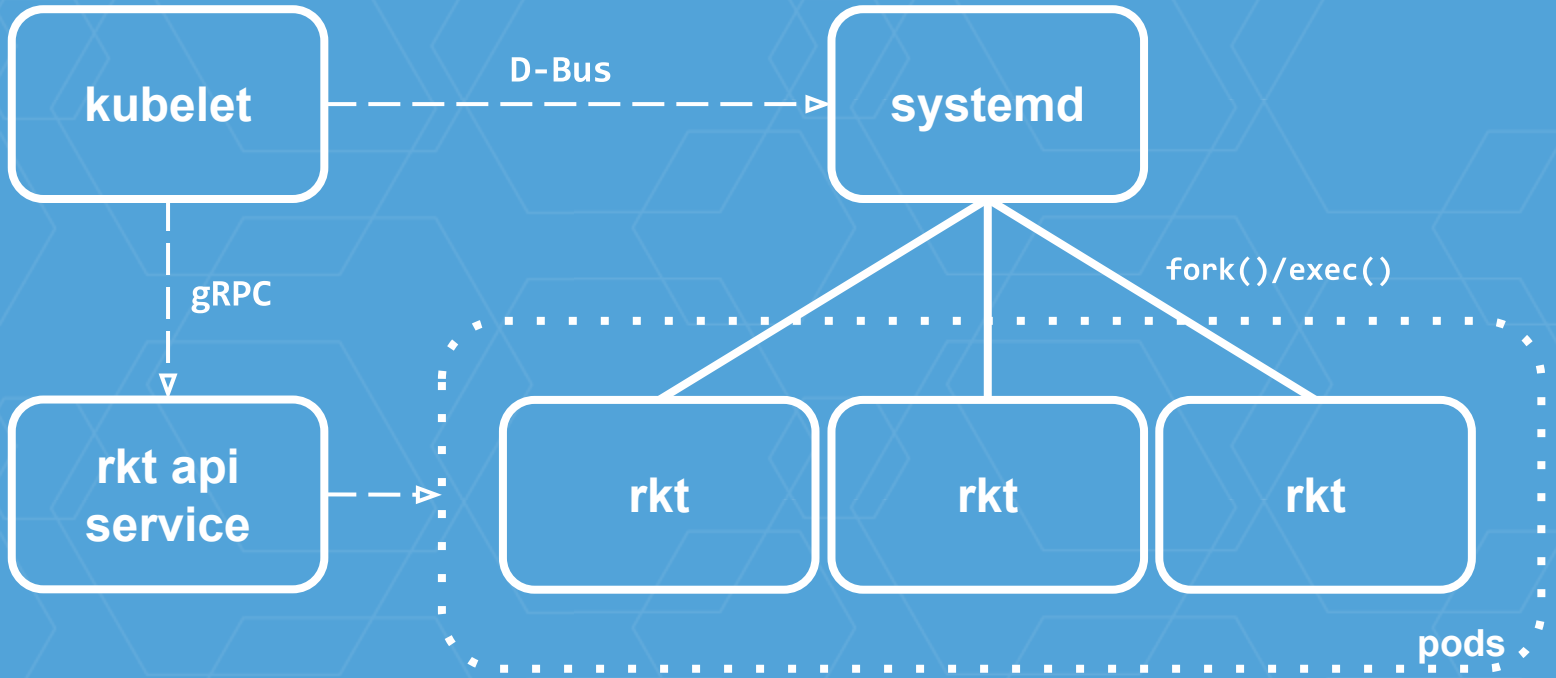
With rktnetes:

- 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 + rkt (rktnetes)



Kubelet + rkt (rktnetes)



What's the benefit in this?

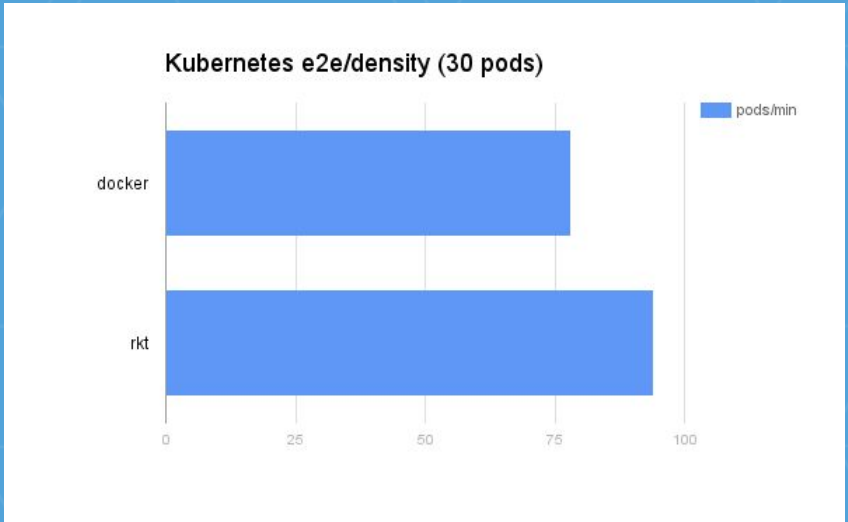
- **No daemon running the containers**
  - live upgrades of the container runtime without affecting existing pods
- **Multiple stage1s provides more flexibility**
  - Swap in more advanced isolation technologies without needing to modify Kubernetes
- **Seamless integration with systemd**
  - `machinectl`,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

What's the benefit in this?

- **Paves the way for more options**
  - runc
  - Hyper
  - Kurma
  - Windows containers
- **Keep Kubernetes honest**
  - Maintain contract of what Kubelet is responsible for, what container runtimes are responsible for

What's the benefit in this?

- Allow runtimes to compete on features/speed within Kubernetes



(don't trust simple comparisons :-)

rktnetes: *does* it work?

- **Yes!**

- Official release in Kubernetes 1.3

http://blog.kubernetes.io/2016/07/rktnetes-brings-rkt-container-engine-to-Kubernetes.html

- Tracking 100% parity for Kubernetes 1.5

https://github.com/kubernetes/features/issues/58

How can I use it?

- A getting started guide is in the Kubernetes docs: http://kubernetes.io/docs/getting-started-guides/rkt/
- Check out Minikube: https://github.com/kubernetes/minikube
- Watch this space: http://rktnetes.io

What's next with rktnetes?

New container runtime interface

- k8s is reworking the interface between the kubelet and the container runtime
  - kubelet wants fine-grained control over containers
  - Move away from declarative, monolithic functions (`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

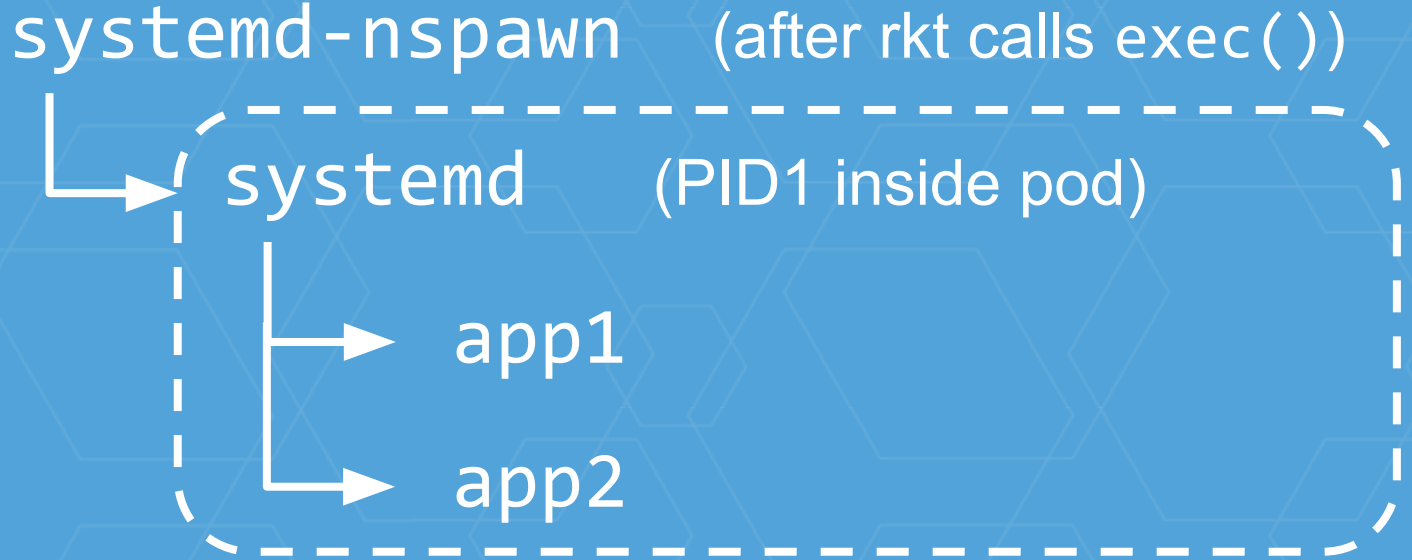
- Next version of rkt integration: *rktlet*
  - https://github.com/kubernetes-incubator/rktlet

- New app-level interfaces to rkt
  - `rkt app sandbox` (create an empty pod)
  - `rkt app add app1` (add an app to a pod)

- Still retain benefits of first-class pods + systemd integration

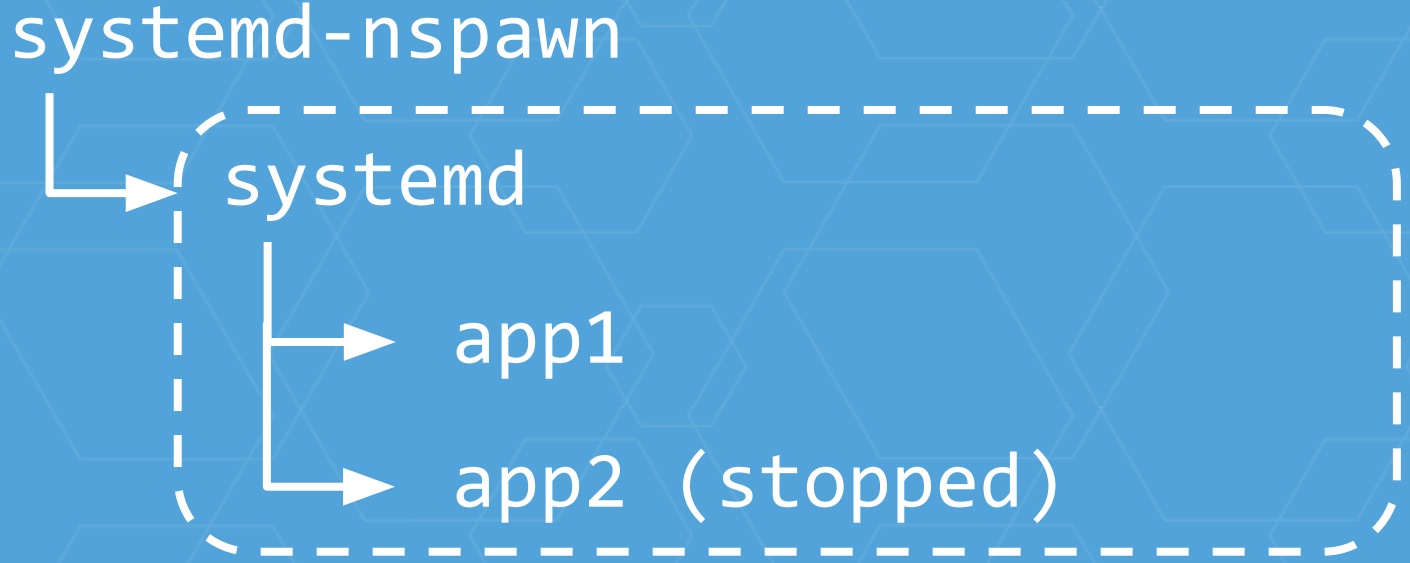
rkt pods with app level interfaces

rkt pods with app level interfaces



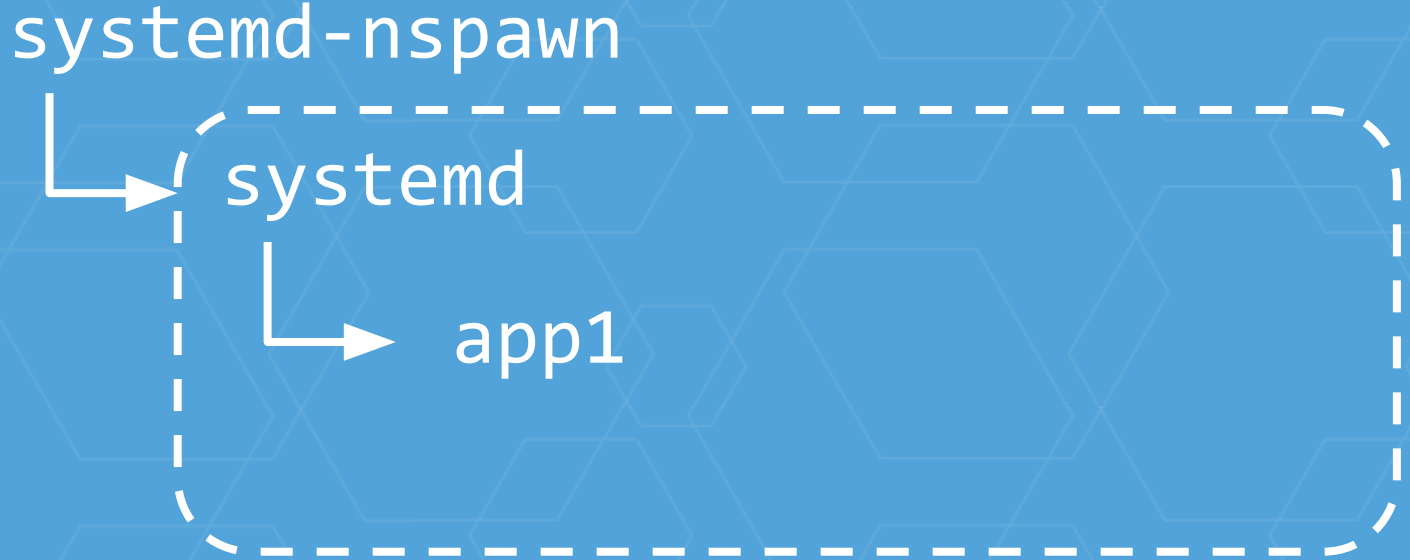
rkt pods with app level interfaces

```
$ rkt app stop <mypod> app2
```



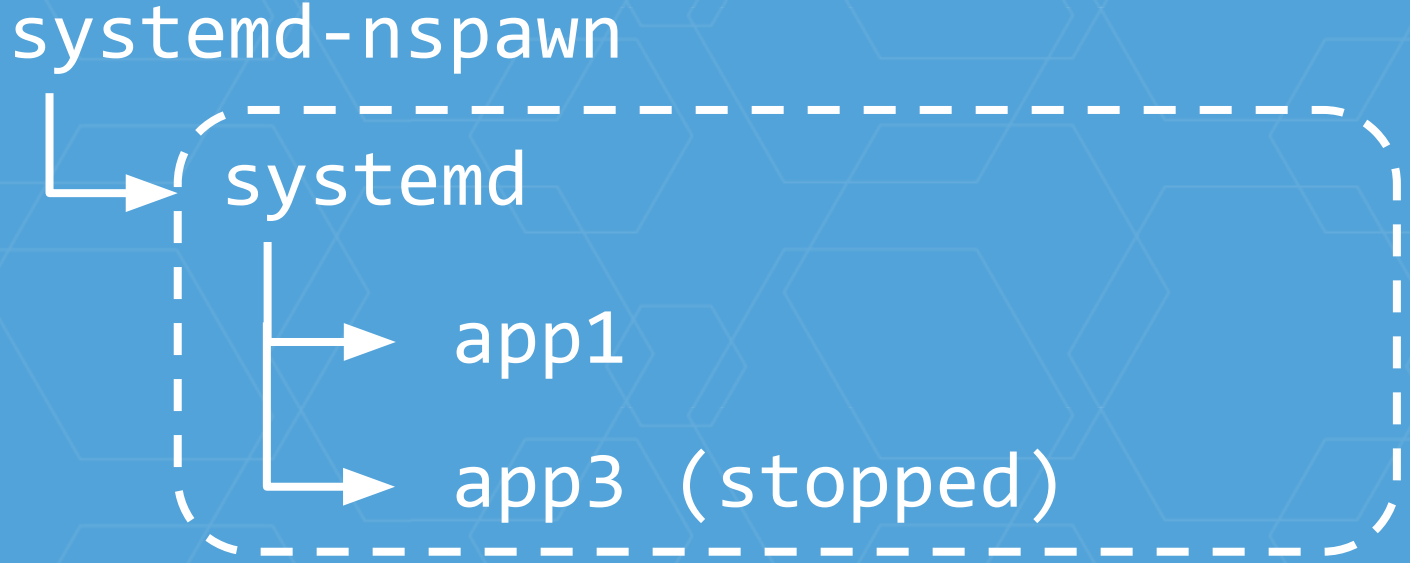
rkt pods with app level interfaces

```
$ rkt app rm <mypod> app2
```



rkt pods with app level interfaces

```
$ rkt app add <mypod> app3
```



New* container image format

- **OCI: a container image format we can *all* agree on**
  - Based on Docker v2.2 image format (*not really "new")
  - + optional components like signing and naming
  - Maintainers from Docker, CoreOS, Red Hat, Google
  - First, reach a 1.0 (soon!): then, push this image format into the Kubernetes API
  - https://github.com/kubernetes/features/issues/63

How do I find out more?

- **Reach out on GitHub or IRC**
  - github.com/coreos/rkt, #rkt-dev / #rkt on Freenode
- **Join a Kubernetes Special Interest Group (SIG)**
  - https://groups.google.com/forum/#!forum/kubernetes-sig-node
  - https://groups.google.com/forum/#!forum/kubernetes-sig-rktnetes
  - `#sig-node` / `#sig-rktnetes` on Kubernetes Slack
- **Join us!**
  - Hiring rkt developers in Berlin

Questions?

thanks for listening 

CoreOS

CoreOS is Running the World's Containers

OPEN SOURCE

90+ Projects on GitHub, 1,000+ Contributors

CoreOS.com - @coreoslinux - github/coreos

CoreOS rkt etcd

ENTERPRISE

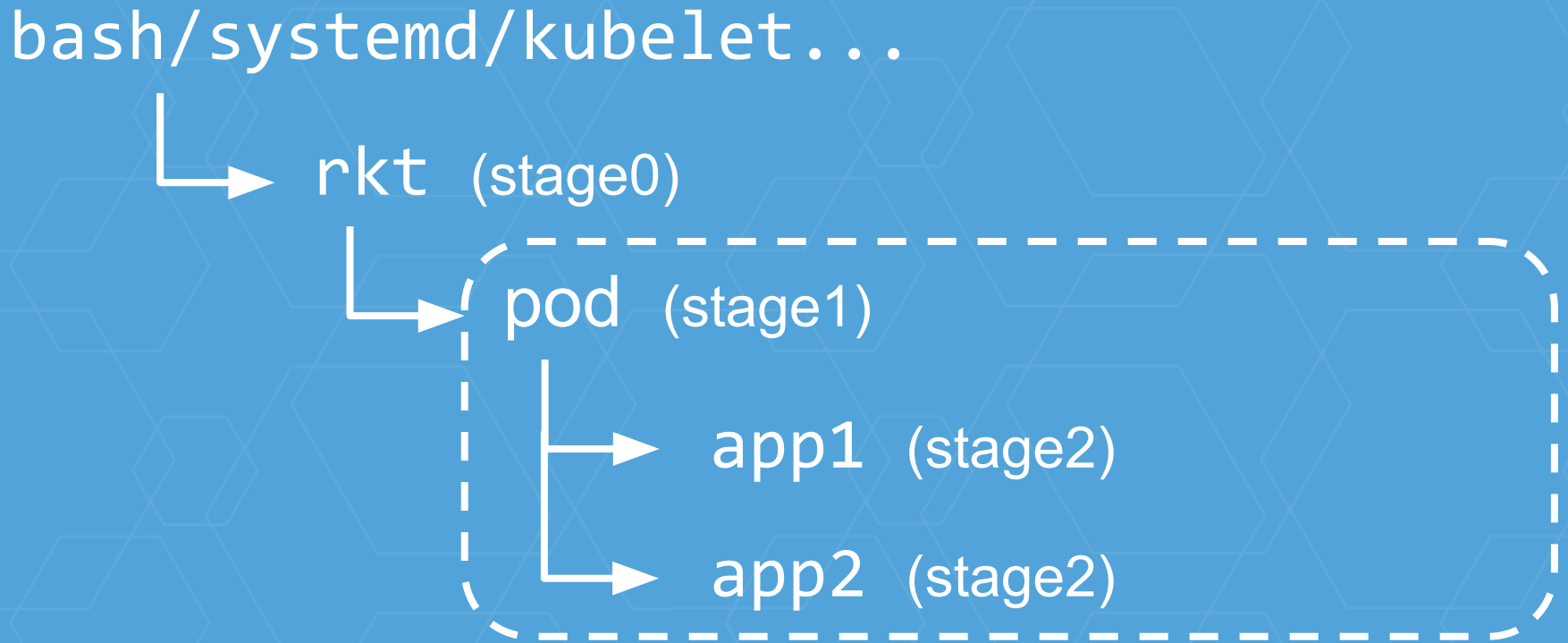
Secure solutions, support plans, training + more

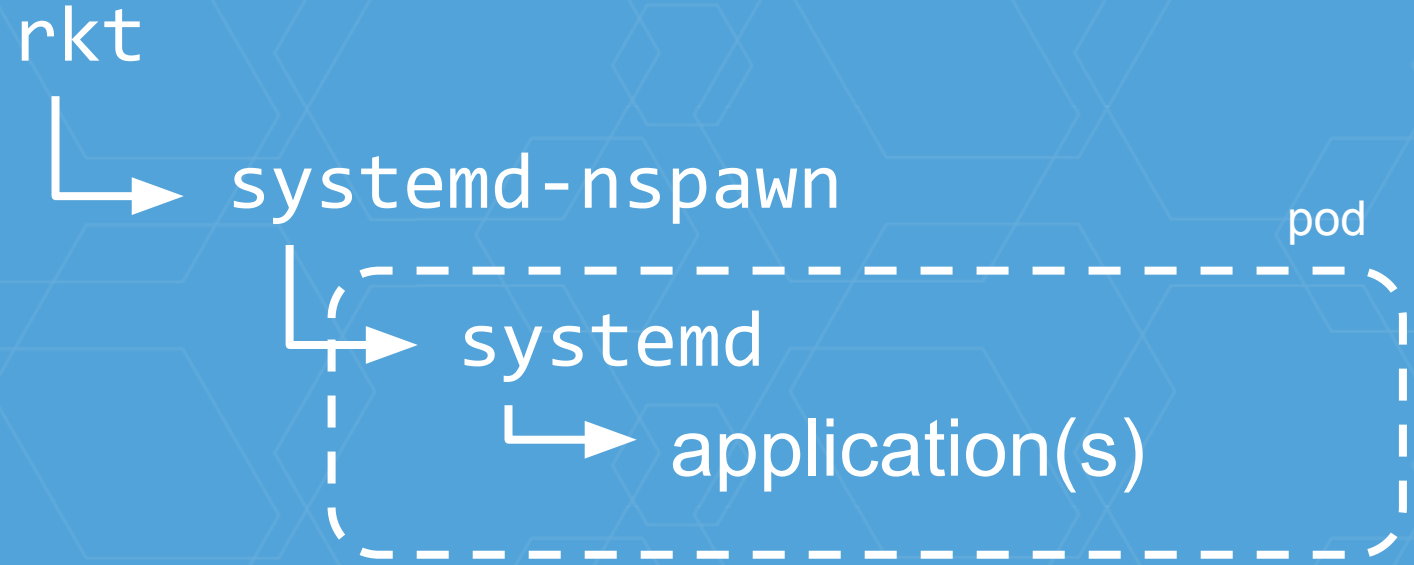
sales@coreos.com - tectonic.com - quay.io

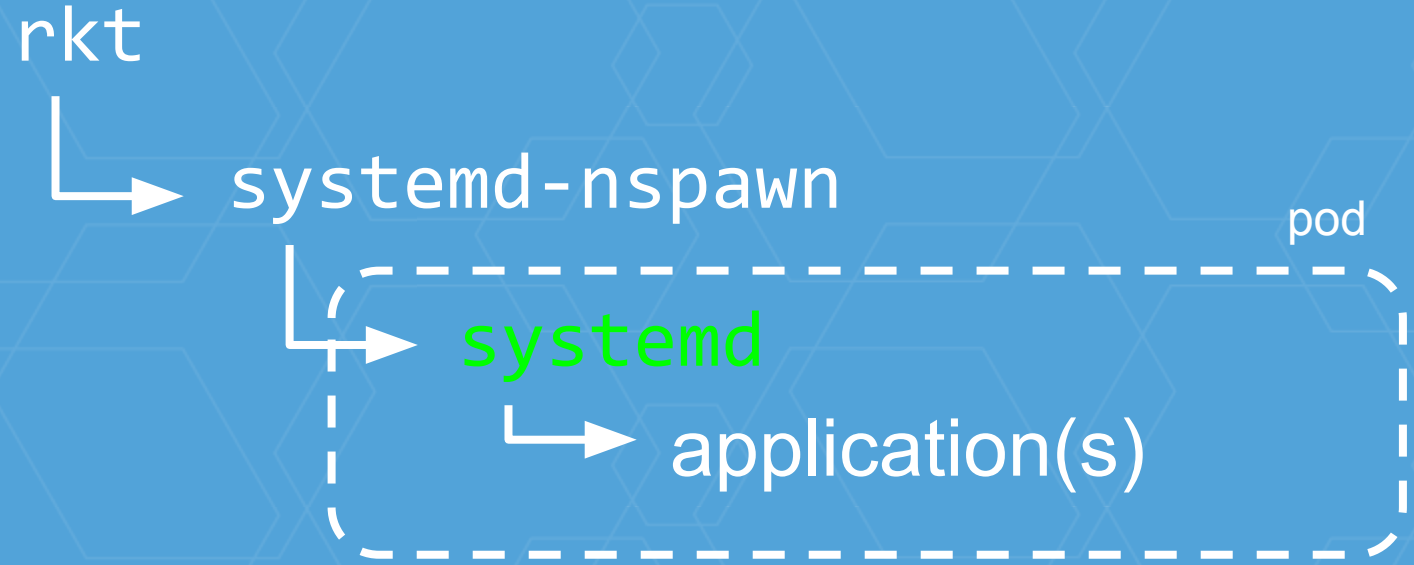
TECTONIC QUAY

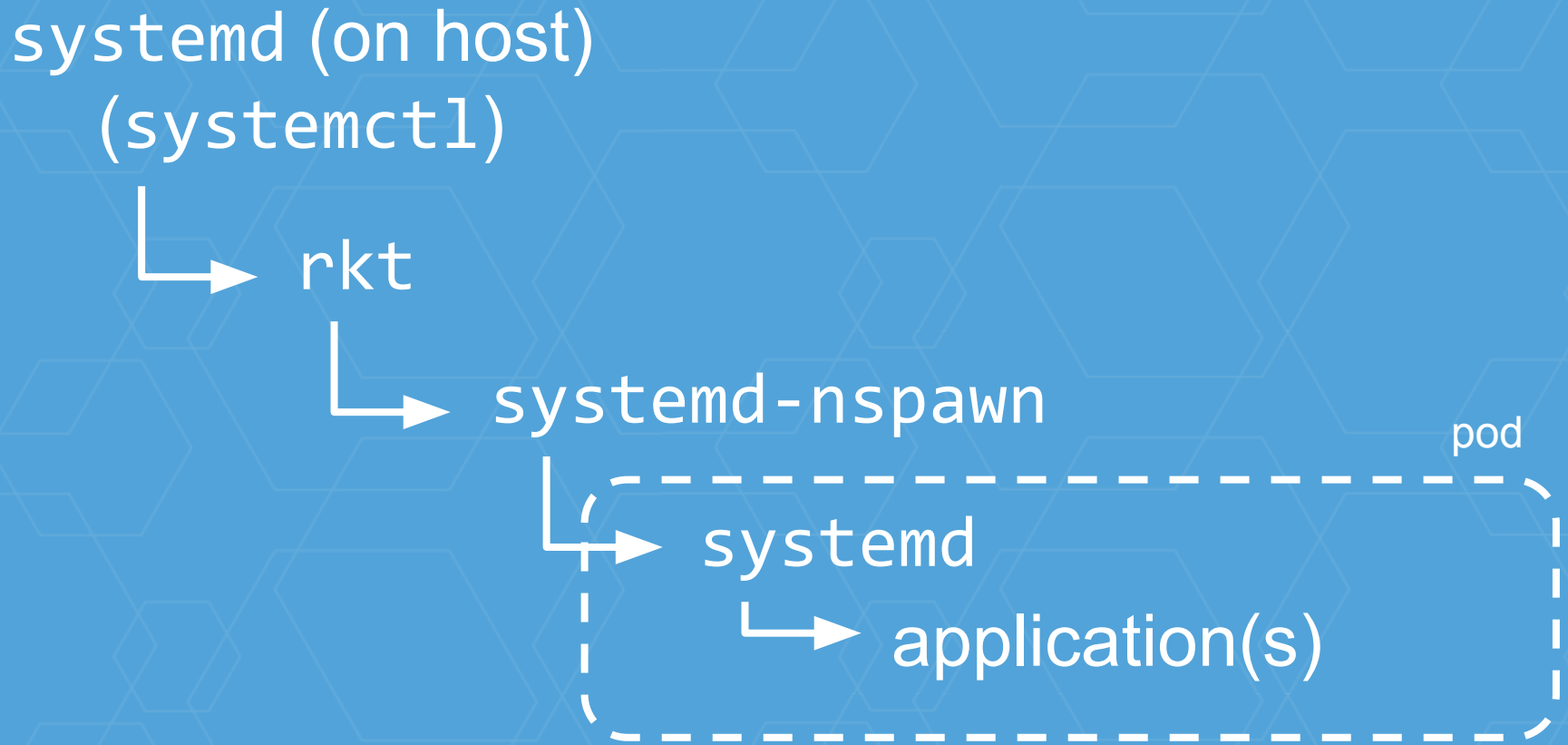
We're hiring in all departments! Email: careers@coreos.com Positions: coreos.com/ careers

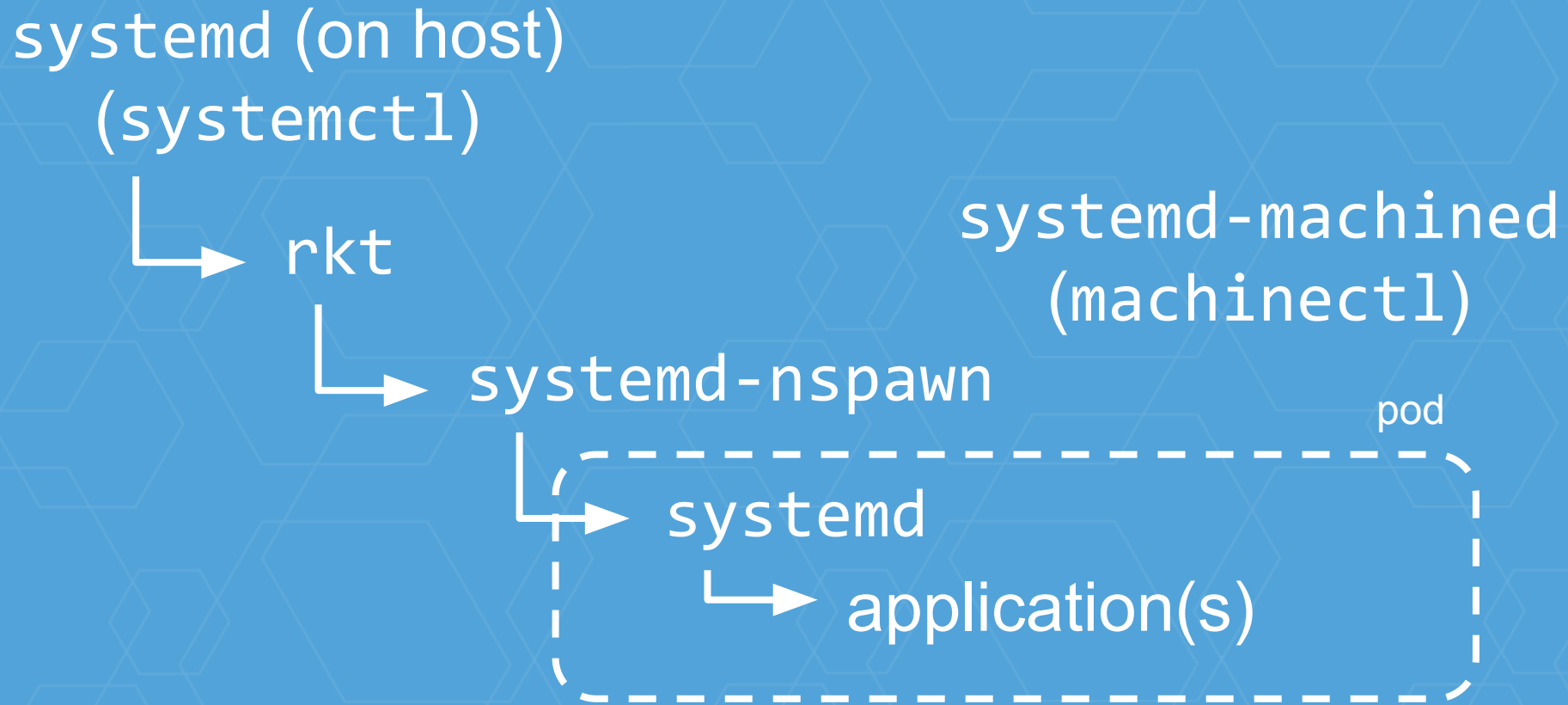
Extra slides

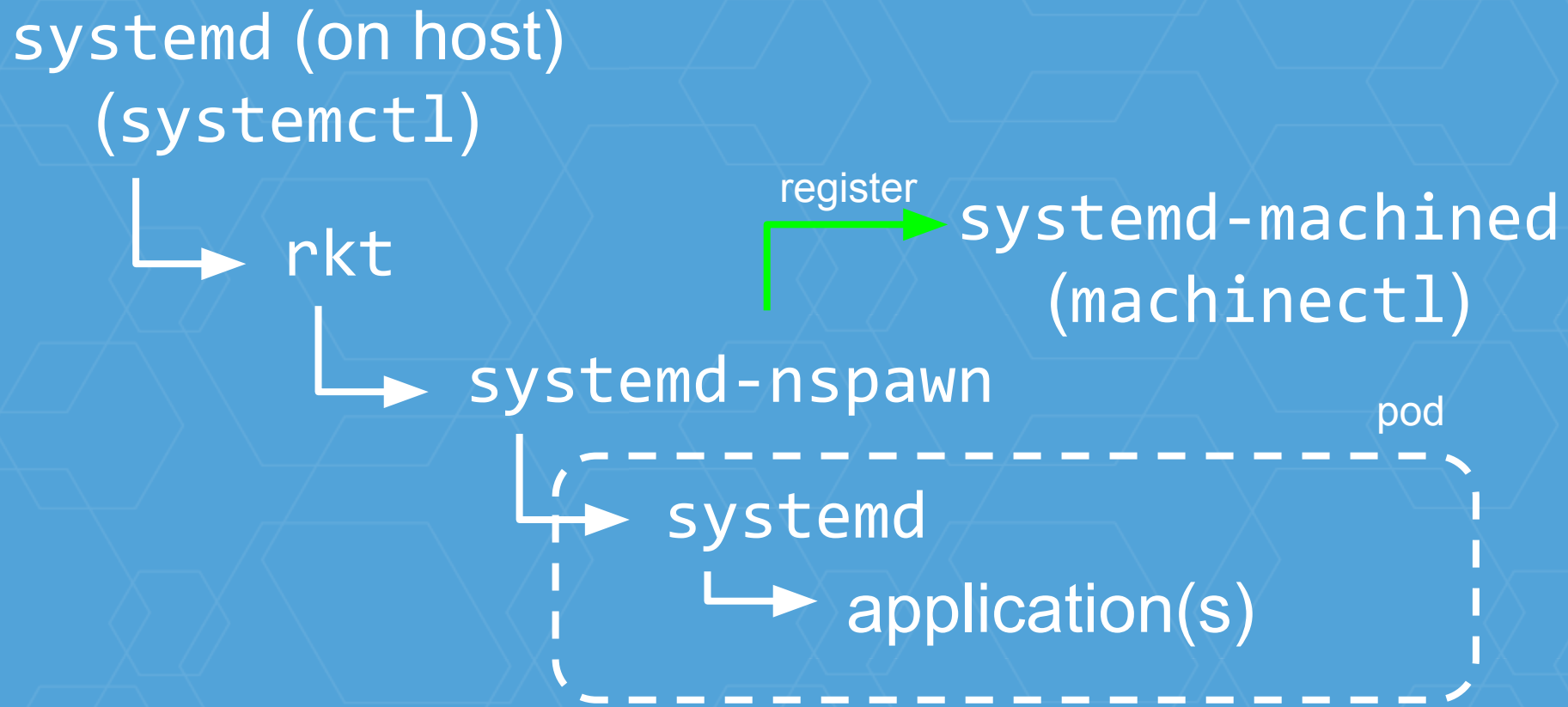

bash/systemd/kubelet...
rkt (stage0)
pod (stage1)
app1 (stage2)
app2 (stage2)

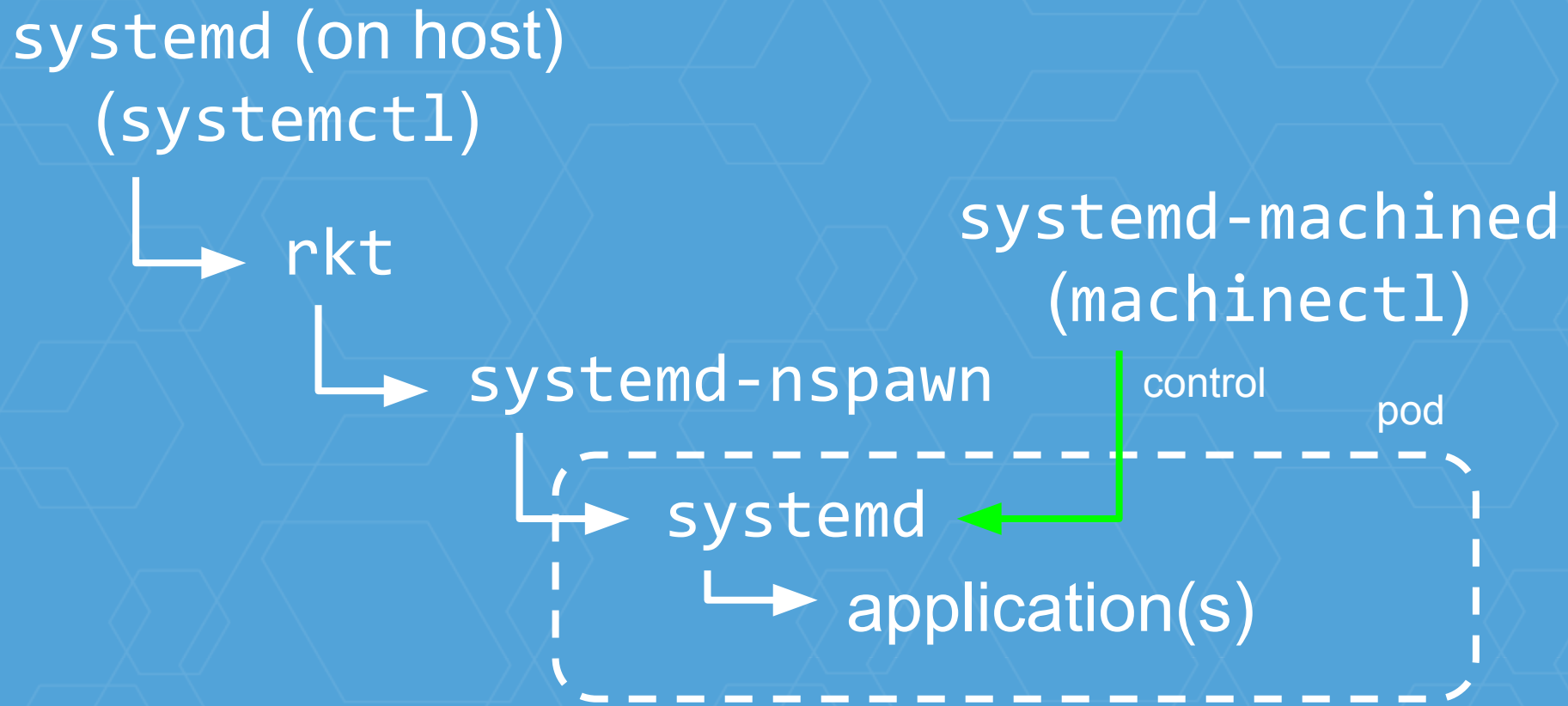

rkt
systemd-nspawn
pod
systemd
application(s)

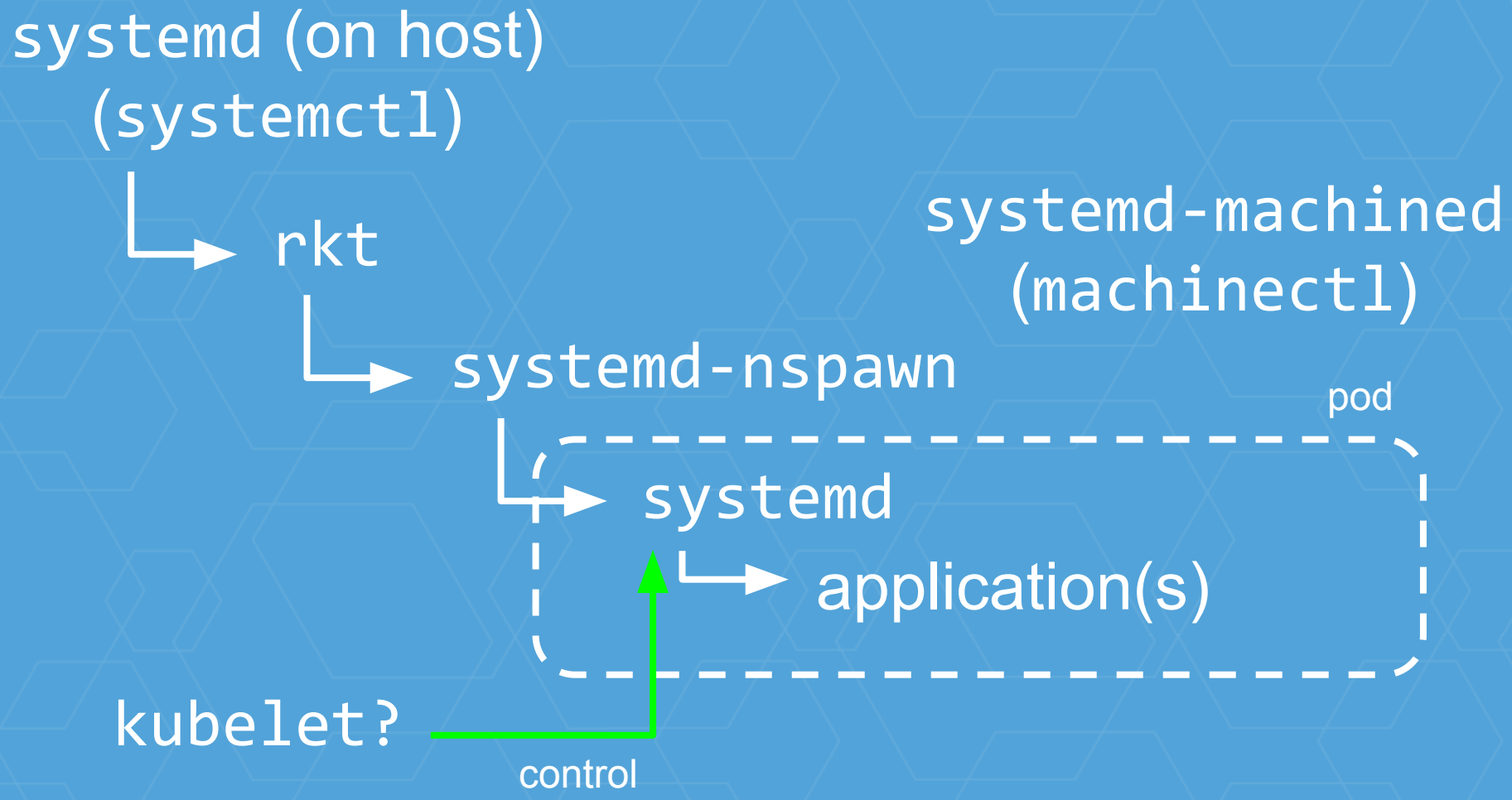

rkt
systemd-nspawn
pod
systemd
application(s)

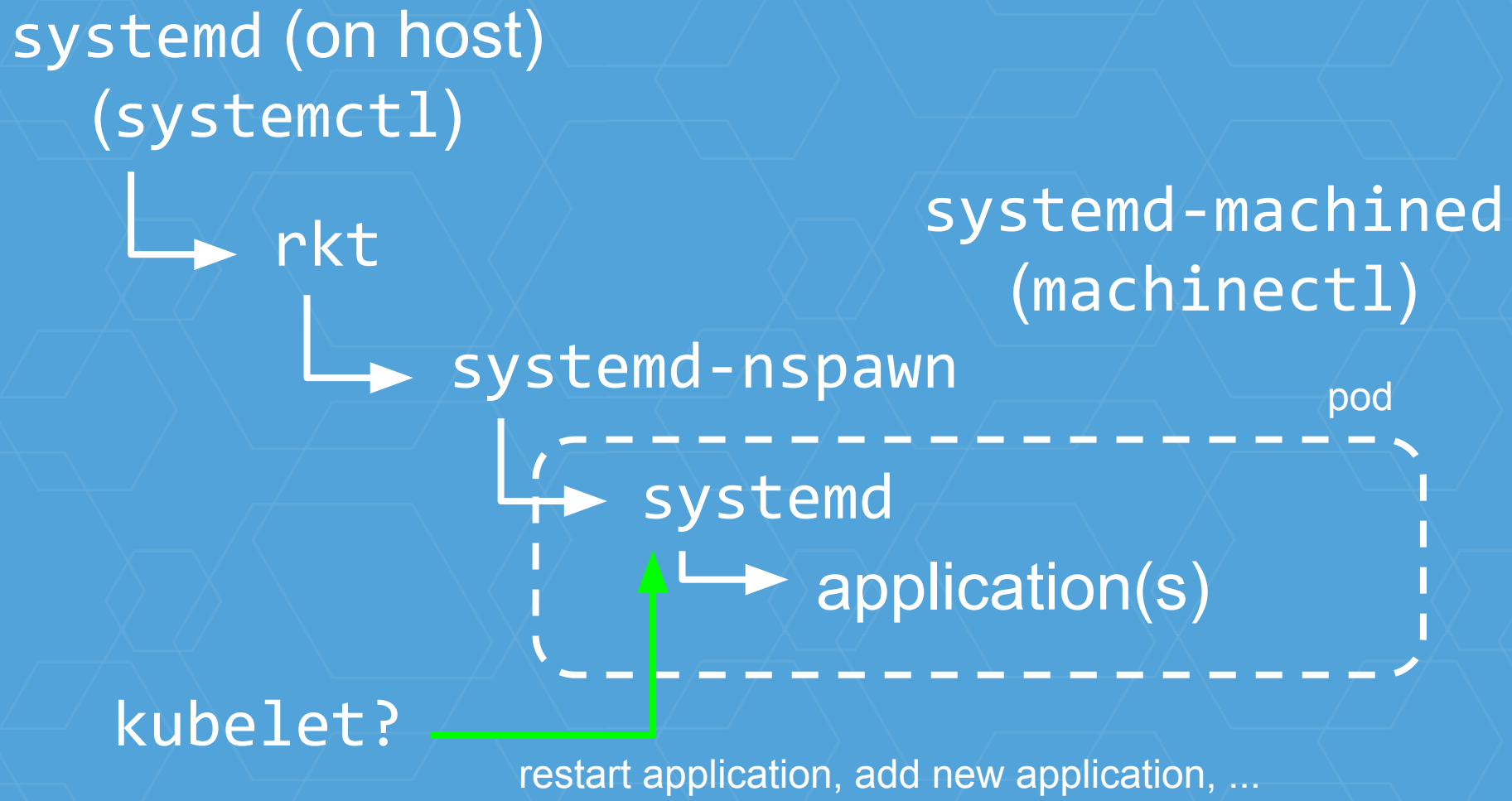

systemd (on host)
(systemctl)
rkt
systemd-nspawn
pod
systemd
application(s)


systemd (on host)
(systemctl)
rkt
systemd-machined
(machinectl)
systemd-nspawn
pod
systemd
application(s)


systemd (on host)
(systemctl)
rkt
register
systemd-machined
(machinectl)
systemd-nspawn
pod
systemd
application(s)


systemd (on host)
(systemctl)
rkt
systemd-machined
(machinectl)
systemd-nspawn
control
pod
systemd
application(s)


systemd (on host)
(systemctl)
rkt
systemd-machined
(machinectl)
systemd-nspawn
pod
systemd
application(s)
kubelet?
control


systemd (on host)
(systemctl)
rkt
systemd-machined
(machinectl)
systemd-nspawn
pod
systemd
application(s)
kubelet?
restart application, add new application, ...

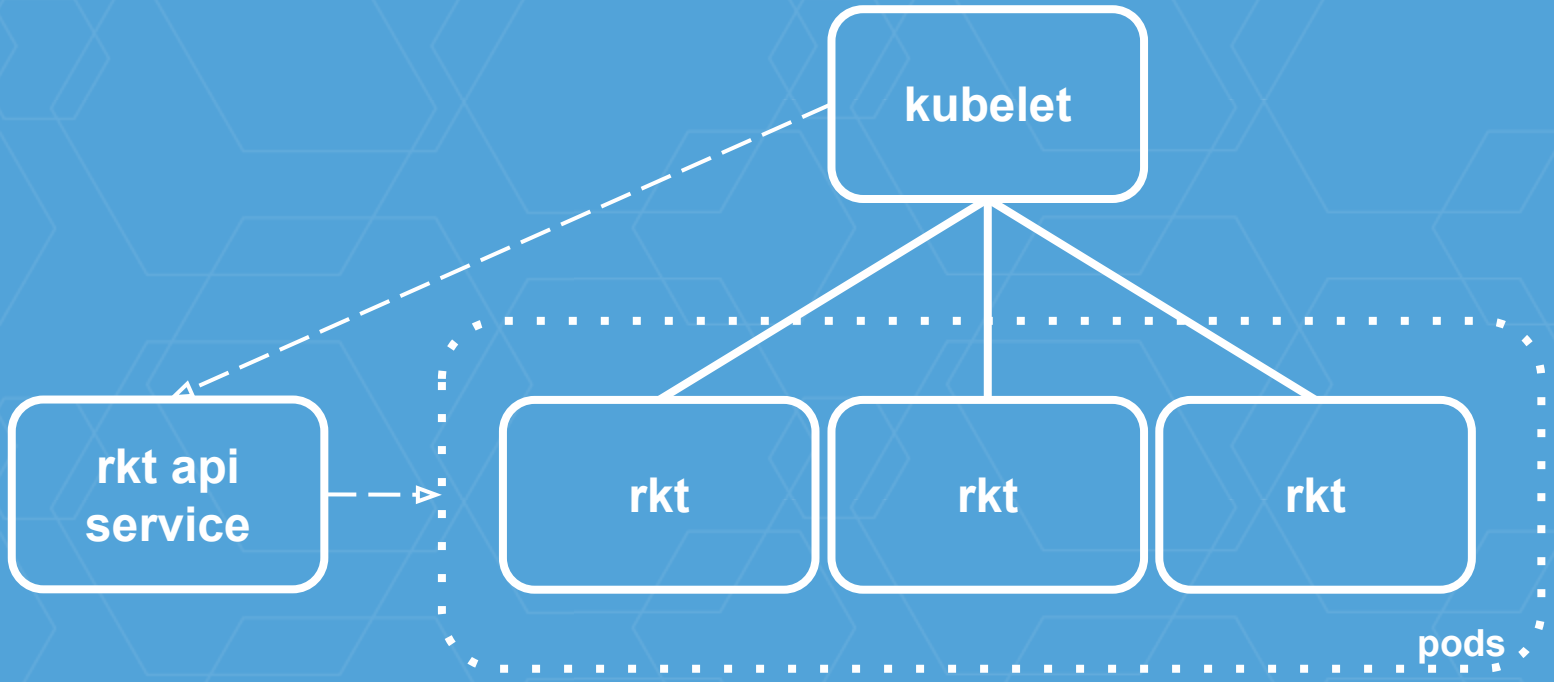
Kubelet + Docker (future)

- **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

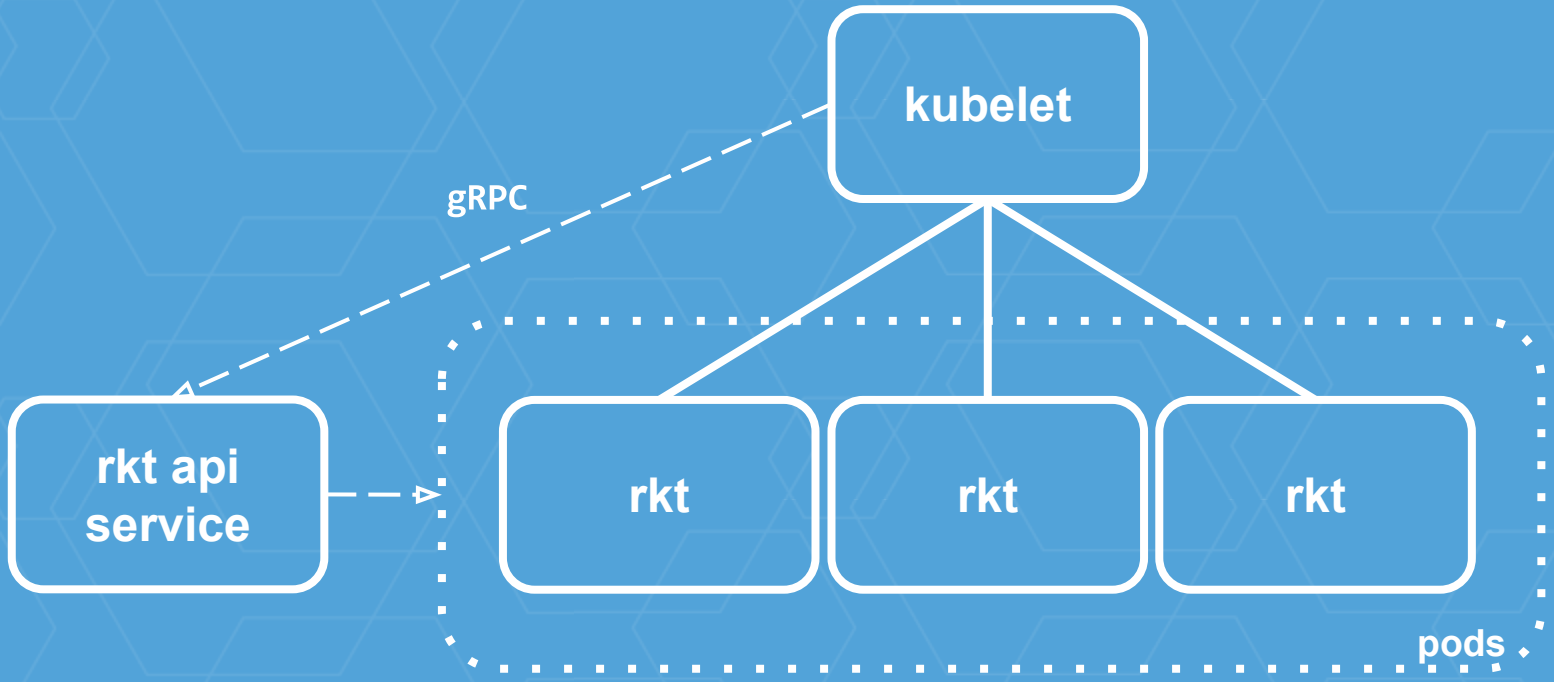
Kubelet + Docker (1.11+ with containerd)

- `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 - 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

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/0UIiE43yAwAJ
- 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