All about rkt: Containers and Kubernetes at CoreOS

Josh Wood

DocOps • CoreOS

@joshixisjosh9 | j@coreos.com | github.com/joshix
CoreOS runs the world’s containers

We’re hiring: careers@coreos.com

<table>
<thead>
<tr>
<th>OPEN SOURCE</th>
<th>ENTERPRISE</th>
</tr>
</thead>
<tbody>
<tr>
<td>90+ Projects on GitHub, 1,000+ Contributors</td>
<td>Support plans, training and more</td>
</tr>
<tr>
<td>container linux</td>
<td>TECTONIC</td>
</tr>
<tr>
<td>rkt</td>
<td>QUAY</td>
</tr>
<tr>
<td>etcd</td>
<td></td>
</tr>
<tr>
<td>coreos.com</td>
<td><a href="mailto:sales@coreos.com">sales@coreos.com</a></td>
</tr>
</tbody>
</table>
What’s it all about?

- Decouple the Application from the OS
  - Then you can upgrade them both -- independently
  - Containers: distribution and execution
- Automate OS upgrades - stay secure
- Orchestrate the result as a unified resource
  - Apps evolve -- are continuously deployed and scaled
- Democratize access to utility computing
  - #GIFEE
A CLI for running app containers on Linux.

Focuses on:
- Security
- Modularity
- 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
rkt

A CLI for running app containers on Linux.

Focuses on:
- Not reinventing the wheel:
  - Systemd - init
  - Overlayfs
  - CNI networking
A CLI for running app containers on Linux.

Security:
- Signed images
- GPG detached sigs (ACI)
- DTC integration with TPM
rkt
A CLI for running app containers on Linux.

Modularity: External
- “Fits in”
- Systemd or other init
- CNI and plugins
rkt

A CLI for running app containers on Linux.

Modularity: Internal

- Stages of execution
- Fly, cgroups/ns, KVM vm
  - SAME CONTAINER
rkt
A CLI for running app containers on Linux.

Standards/Compatibility:
● Appc ACI format & sigs
● **rkt runs Docker images**
  ○ OCI support as develops
rkt run: default stage1

- Isolates containers with the linux container primitives (cgroups, ns), systemd-nspawn
- Container apps in a machine slice PID namespace
- Manage with standard init tools: systemd
- Network isolation
rkt run: KVM isolation

- Isolates containers with the Linux KVM hypervisor
- Container apps in a machine slice PID namespace
- Manage with standard init tools: systemd
- Network isolation
rkt fly

- Leverages the packaging, discovery, distribution, and validation features of rkt/containers
- Reduced isolation for privileged components
- chroot file system isolation only
- Has access to host-level mount, network, PID namespaces
- Method for infra bootstrap in CoreOS Linux
rkt run: your stage1

- stage1 can be replaced with custom implementations for security, performance, architecture, ...
- KVM stage1 originated with Intel ClearContainers project and has seen at least two alternate external implementations
$ rkt run quay.io/josh_wood/caddy
rkt: using image from local store for image name coreos.com/rkt/stage1-coreos:0.15.0
rkt: using image from local store for image name quay.io/josh_wood/caddy
[ 1161.330635] caddy[4]: Activating privacy features... done.
[ 1161.333482] caddy[4]: :2015
$
Kubernetes

Cluster-level container orchestration with #GIFEE baked in.

Handles:
- Scheduling/Upgrades
- Failure recovery
- Scaling
What is rkt in Kubernetes?

- “Rktnetes” was a nickname for the work in both rkt and kubernetes
- rkt is container execution engine, runs cluster work on nodes
- Add configuration to declare a node uses the rkt engine, or that a pod executes with rkt
What is rkt in Kubernetes?
Why rkt in Kubernetes?

- Ensure cleanliness and modularity of the critical interface between the orchestrator and the execution engine
- Spur innovation through community effects
- In short: standards and interfaces
Why rkt in Kubernetes?

- Obtain unique rkt features
- Externally modular: Refine runtime interface: CRI
- Internally modular: Pluggable “stage1” isolation environments
- Run pods as software-isolated (cgroups, ns)
- Run pods as VMs with hypervisor isolation
What’s it all about?

● Decouple the Application from the OS
  ○ Then you can upgrade them both, and each
  ○ Containers: distribution and execution

● Automate OS upgrades

● Orchestrate the result as a unified resource
  ○ Apps evolve -- are continuously deployed and scaled

● Democratize access to utility computing
  ○ #GIFEE
Markers

- CRI - Kubernetes Container Runtime Interface
- CNI as Kubernetes network plugin model
- Docker refactor: runc, containerd
- Appc -> OCI: Standard for container images
- Ocid, et al: Let 1000 runtimes bloom?
  - ocid: Inherits runc: Pro and Con
See also:

- coreos.com/rkt
- github.com/opencontainers/image-spec
- kubernetes.io/docs/getting-started-guides/rkt/
- blog.kubernetes.io/2016/07/rktnetes-brings-rkt-container-engine-to-Kubernetes.html
See also:

- speakerdeck.com/joshix (these slides)
Thank you!

We’re hiring!  Email: careers@coreos.com  Positions: coreos.com/ careers