App Container

github.com/appc

appc-dev@googlegroups.com

Rocket

github.com/coreos/rocket

rocket-dev@googlegroups.com

Jonathan Boulle

github.com/jonboulle
@baronboulle

Core OS

!=

App Container (appc)

github.com/appc
appc-dev@googlegroups.com

appc principles

Simple but efficient

Simple to understand and implement, but eye to optimisations (e.g. aggressive content-based caching)

Secure

Cryptographic image addressing
Image signing and encryption
Container identity

Standards-based

Well-known tools (tar, gzip, gpg, http), extensible with modern technologies (bittorrent, xz)

Composable

Integration with existing init systems and process managers

OS/architecture agnostic

appc components

Image Format

Application Container Image

tarball of rootfs + manifest

uniquely identified by ImageID (hash)

Image Discovery

App name →artefact
example.com/http-server
coreos.com/etcd

Executor

runtime environment
isolators
networking

Metadata Server

`http://$AC_METADATA_URL/acMetadata`

container metadata

container identity (HMAC verification)

appc tooling

$ actool build

rootfs + manifest → ACI

$ actool validate

is this ACI compliant with the spec?

$ actool discover

example.com/app -> https://example.com/releases/app.aci

appc community

cdaylward/libappc

C++ library for working with app containers

(sidenote: mesos)

https://issues.apache.org/jira/browse/MESOS-2162

3ofcoins/jetpack

FreeBSD Jails/ZFS-based executor
(by @mpasternacki)

sgotti/acido

ACI/Rocket utility and testbed

appc/docker2aci

```
docker2aci busybox/latest
docker2aci quay.io/coreos/etcd
```

appc status

Stabilising
v0.2.0+git
TODO: pods, isolators, best practices

Rocket

github.com/coreos/rocket

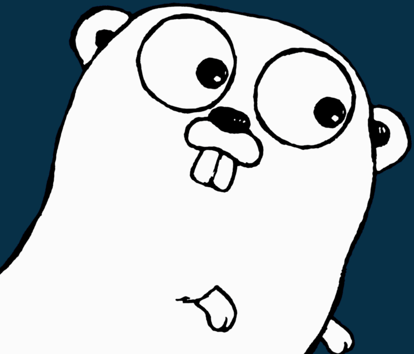
rocket-dev@googlegroups.com

appc implementation

discovery
executor
metadata service

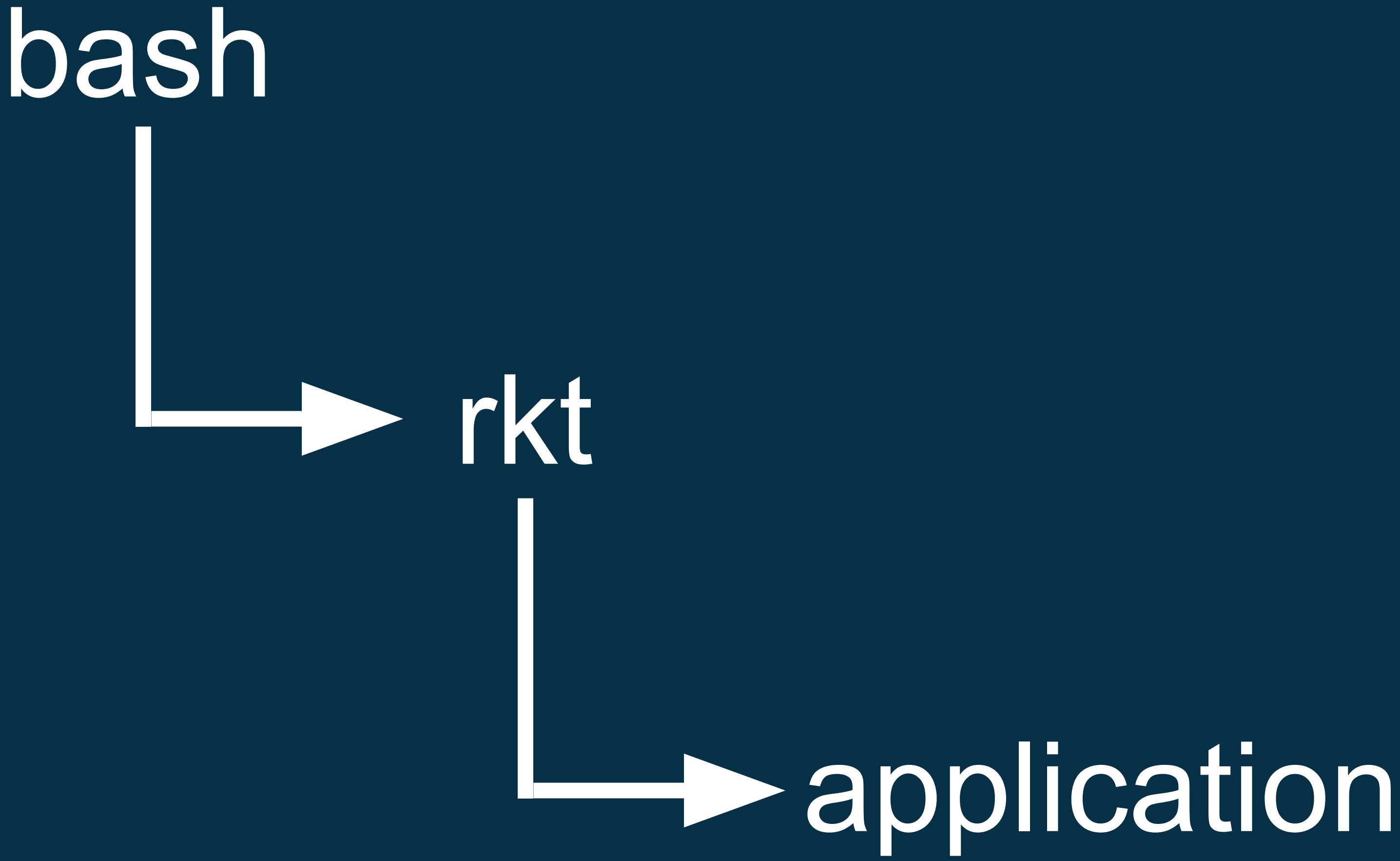
golang + Linux

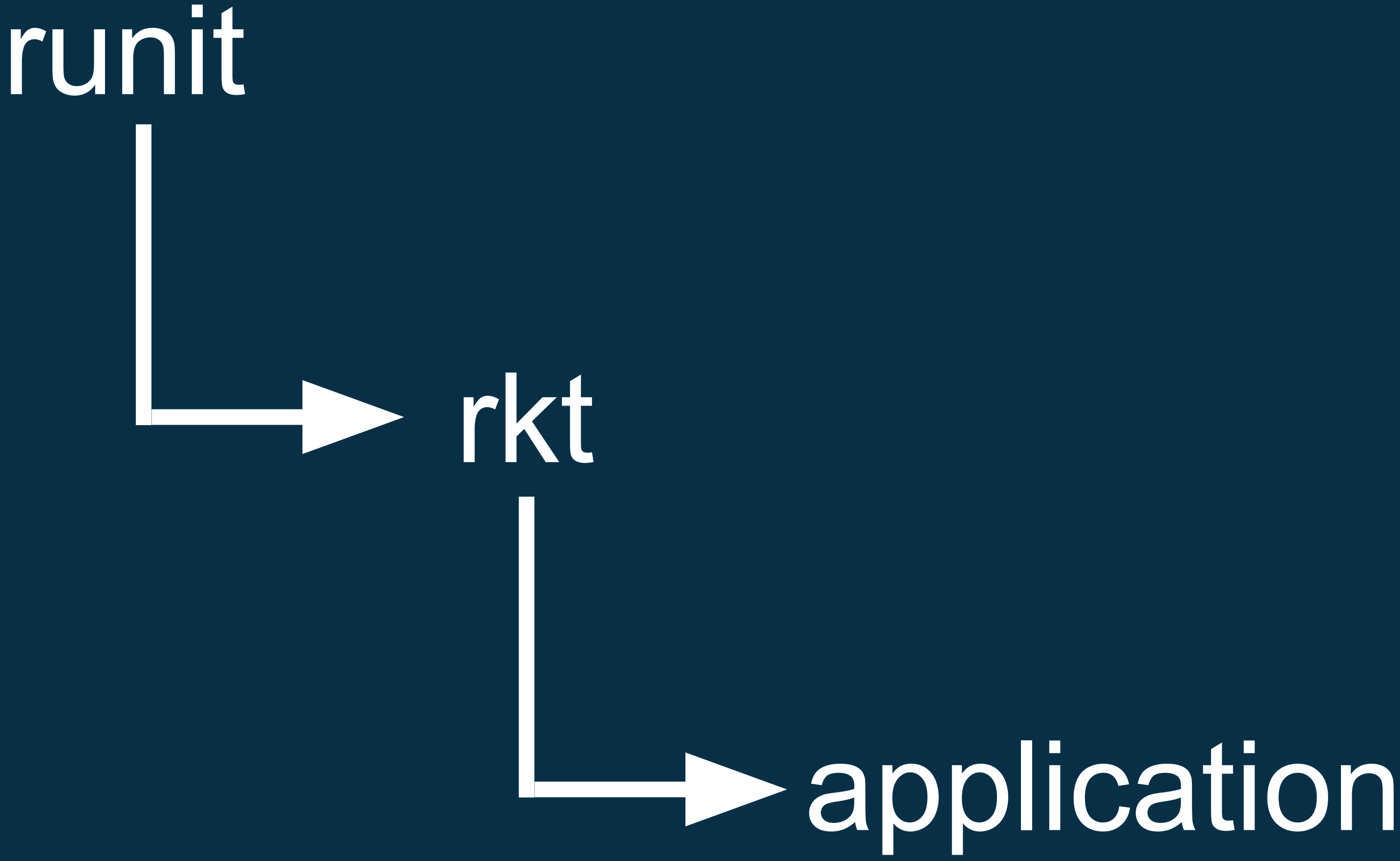
self-contained
init system agnostic

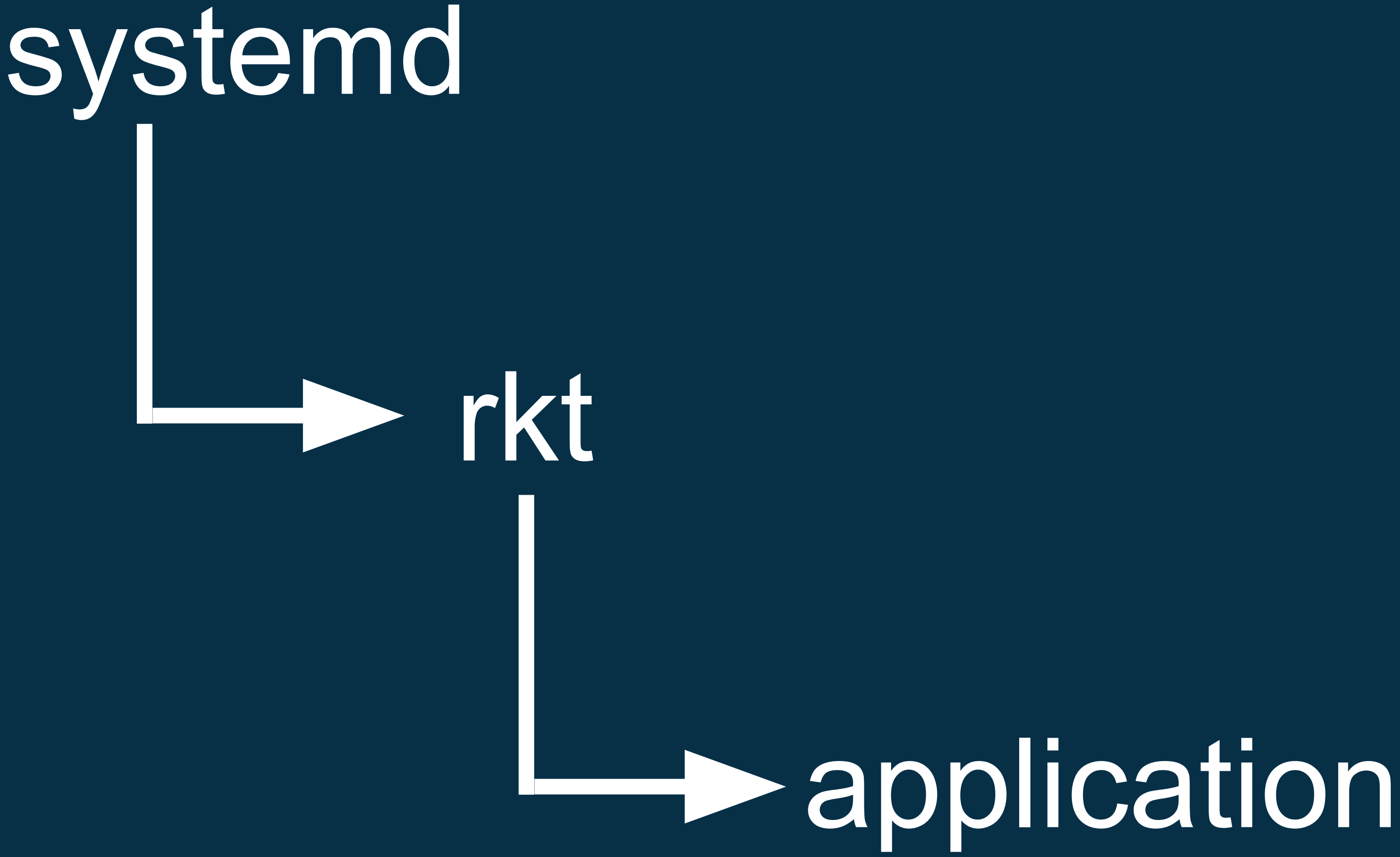


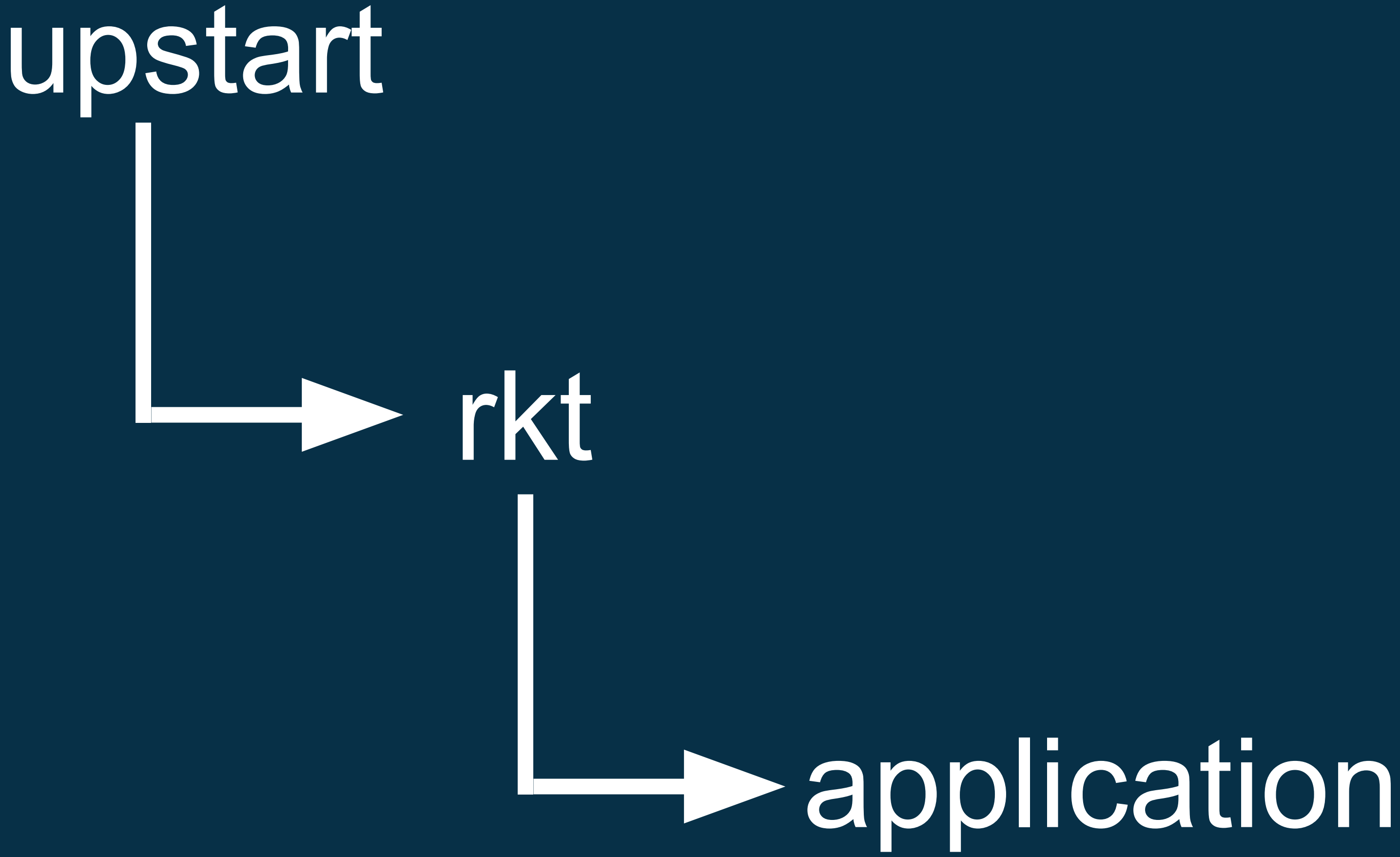
CLI only

no daemon
apps run directly under spawning process


bash
rkt
application


runit
rkt
application


systemd
rkt
application


upstart
rkt
application

Rocket internals

modular architecture

execution divided into *stages*

stage0

rkt binary
discover, retrieve application images
set up container filesystems

stage1

execution environment for apps
container *rootfs* + *init* binary
app process management, cgroups,
metadata service

stage2

actual app execution

rocket v0.1.0

first version (announcement)
somewhat limited..

rkt fetch

rkt fetch https://example.com/my_app.aci

rkt fetch coreos.com/etcd:v2.0.0.rc1

simple CAS on disk

rkt run

```
rkt run coreos.com/etcd:v2.0.0-rc.1
rkt run ./my-app.aci
rkt run sha512-fcdf125873...
```

rocket v0.2.0+git

what's new?

new commands!

rkt enter
rkt list
rkt status
rkt gc
rkt trust

rkt enter, list

enter the namespaces of an application
list containers on the system

rkt status, rkt gc

file-based locking (flock)
mark-and-sweep gc (time based)

rkt trust

easily manage public ACI signing keys

```
rkt trust coreos.com
rkt trust --prefix foo.com https://foo.com/key
```

stage1 from ACI

no more go-bindata
swappable execution environments
distribution packaging friendly!

Rocket

Crash course!

rocket v0.3.0+

what's coming?

networking

"it's complicated"

networking

IP-per-container

extensible plugin-based system

http://goo.gl/lQA9PB

kubernetes

github.com/GoogleCloudPlatform/kubernetes/issues/2725

App Container

+

Rocket

get involved!

GitHub: "help wanted" label

Questions?

Credits

- SpaceX Falcon 9 Landing by Elon Musk
- Golang gopher by Renee French, licensed under CC BY 3.0
- Tux by Larry Ewing, Simon Budig and Anja Gerwinski