App Container

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

github.com/coreos/rocket
rocket-dev@googlegroups.com
App Container (appc)
github.com/appc
appc-dev@googlegroups.com
appc != Rocket
App Container Spec  *noun*

A new, open specification for running applications in containers
Containers?!
Application Containers

self-contained
portable
decoupled from operating system
appc principles

Why are we doing this?
Open

Independent GitHub organisation
Contributions from Cloud Foundry, Mesosphere, Google, Red Hat
(and many others!)
Simple but efficient

Simple to understand and implement, but eye to optimisation (e.g. 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

Integrate with existing systems
Non-prescriptive about build workflows
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

HTTPS + HTML
Executor

grouped applications
runtime environment
isolators
networking
Metadata Service

http://$AC_METADATA_URL/acMetadata
container metadata
container identity (HMAC verification)
apppc 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
ACE validator

is this executor compliant with the spec?

$EXECUTOR run ace_validator.aci
apcc community
cdaylward/libappcc

C++ library for working with app containers
cdaylward/nosecone

C++ executor for running app containers
(sidenote: mesos)

https://issues.apache.org/jira/browse/MESOS-2162
3ofcoins/jetpack

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

ACI toolkit (build ACIs from ACIs)
appc/docker2aci

docker2aci busybox/latest
docker2aci quay.io/coreos/etcd
ap pc status

Stabilising

v0.3.0+git

TODO: pods, isolators
implementation of appc
discovery
executor
metadata service
golang + Linux

self-contained
init system agnostic
CLI only

no daemon

apps run directly under spawning process
systemd → rkt → application
upstart → rkt → application
Rocket internals

modular architecture
execution divided into stages
stage0 → stage1 → stage2
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 v0.3.2+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 --prefix coreos.com/etcd
rkt trust --root https://foo.com/key.asc
stage1 as ACI

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

rkt run docker://redis:latest
Rocket

Crash course!
rocket v0.4.0+

what's coming?
networking

"it's complicated"
host systemd integration

$ machinectl list
$ machinectl terminate
developer environments

interactive containers
filesystem diffs → new ACI
Kubernetes

github.com/GoogleCloudPlatform/kubernetes/issues/2725

http://goo.gl/kJTj96
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