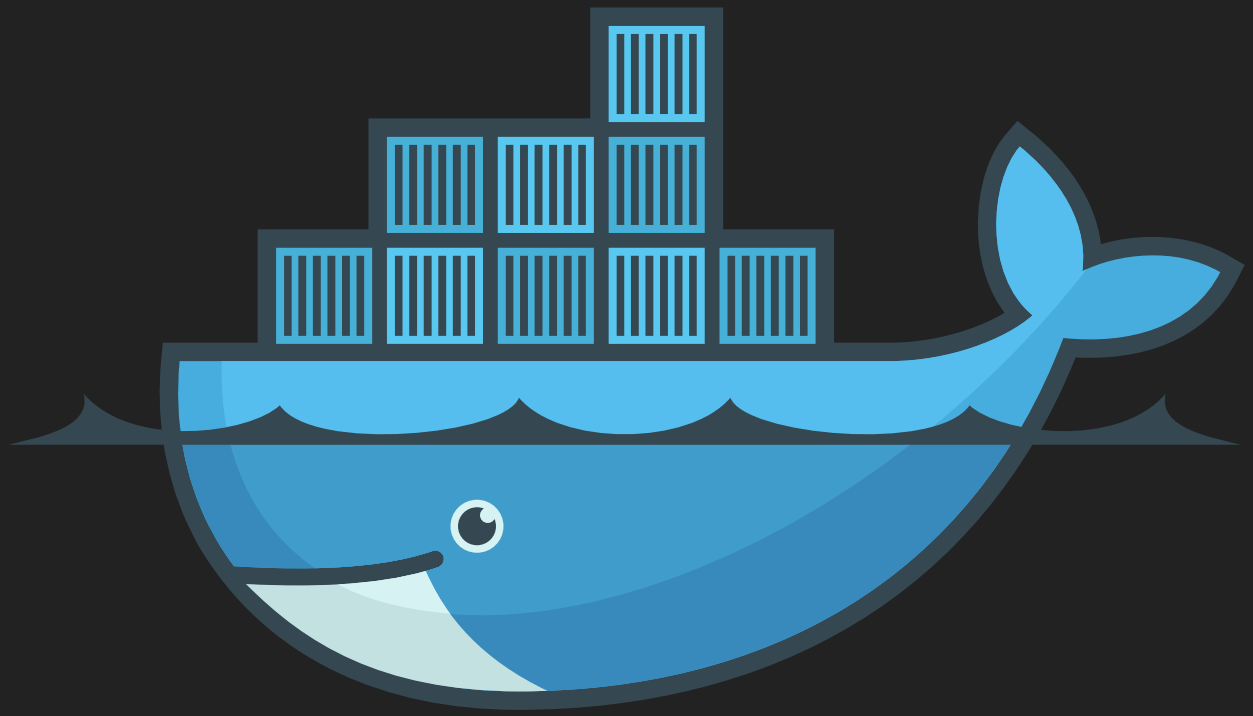
Python on CoreOS

Dan Callahan — @callahad



docker

This is not a talk about Docker

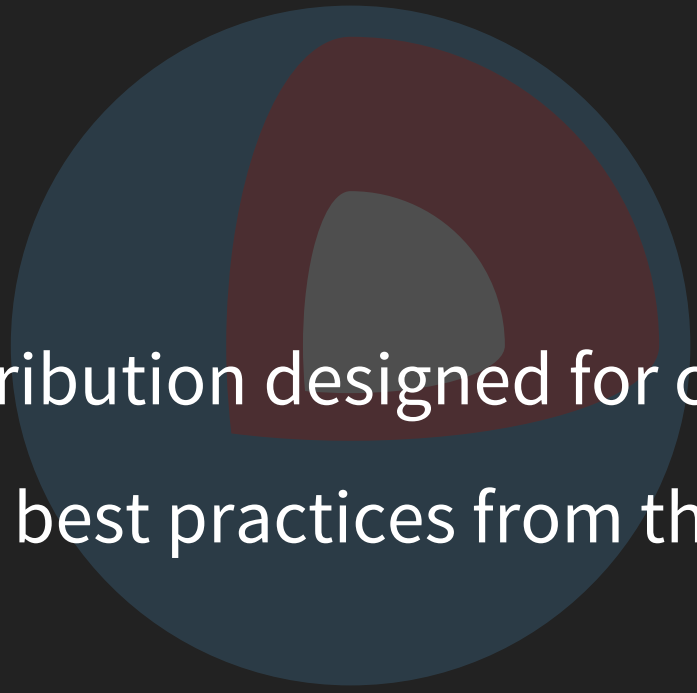
There is one of those tomorrow

docker

This is a talk about servers

And what containerization changes

CoreOS



Linux distribution designed for containers

Preview best practices from the future

CoreOS

What’s your ideal platform?

1. Stays Updated
2. Won’t Break Apps
3. Survives Outages

We need something declarative

> *“Always keep two of these running,*
> *but not on the same machine.”*

We need new technology

1. System Updates
2. Application Isolation
3. Clustering
4. Task Distribution

Technology in CoreOS

1. FastPatch (Updates)
2. Docker / rkt (Containers)
3. Etcd (Consensus)
4. Fleet / Kubernetes (Scheduling)

All Free / Open Source Software

System Updates

(FastPatch)

Staying up-to-date is key to good security

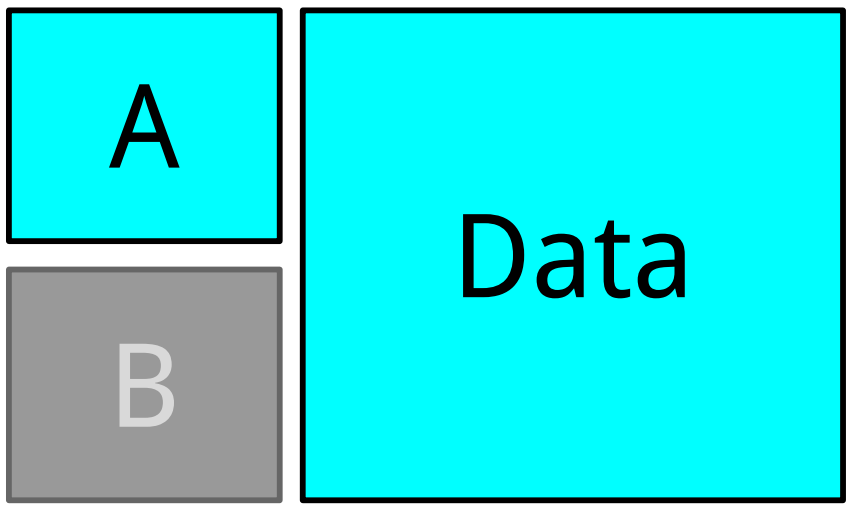
Browser-like Update Channels

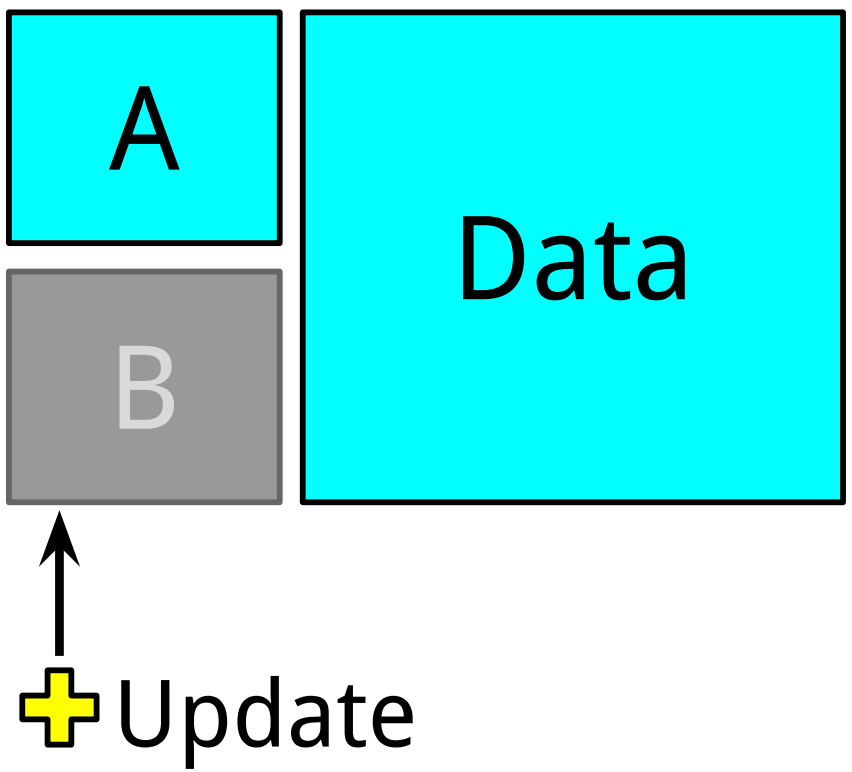
Alpha → Beta → Stable

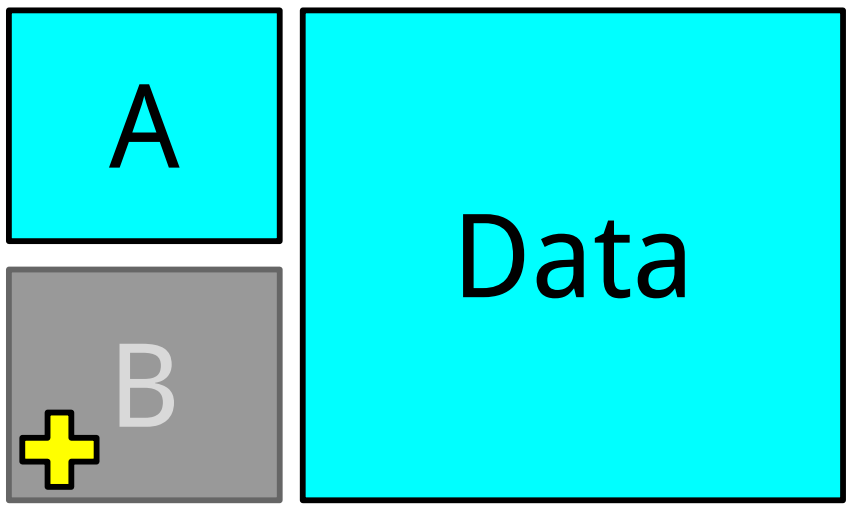
Opportunistically downloaded

Applied on next start

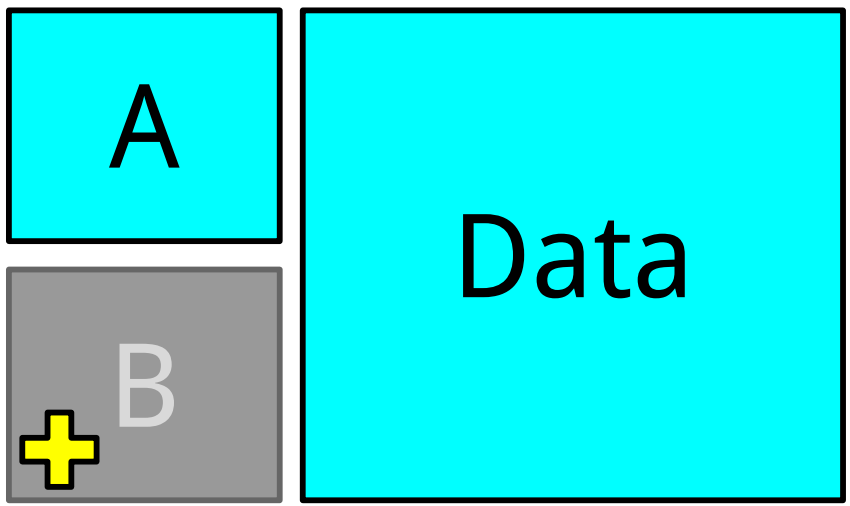
Whole-system Updates

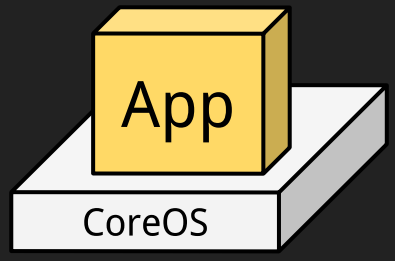

A
B
Data


A
Data
B
Update

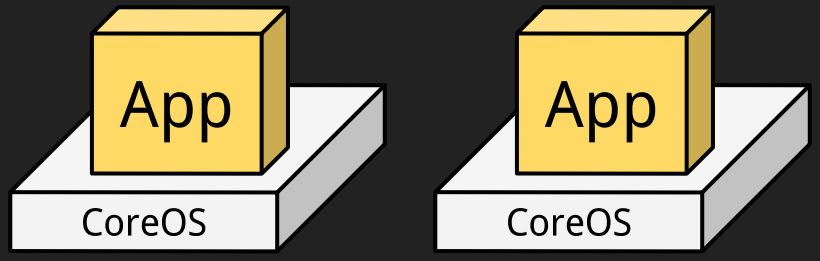

A
B
Data


A
B
Data


A
B
Data


App
CoreOS

My server is rebooting on its own,
how do I keep my app online?


App
CoreOS
App
CoreOS

Both my servers reboot at the same time, how do I keep my app online?

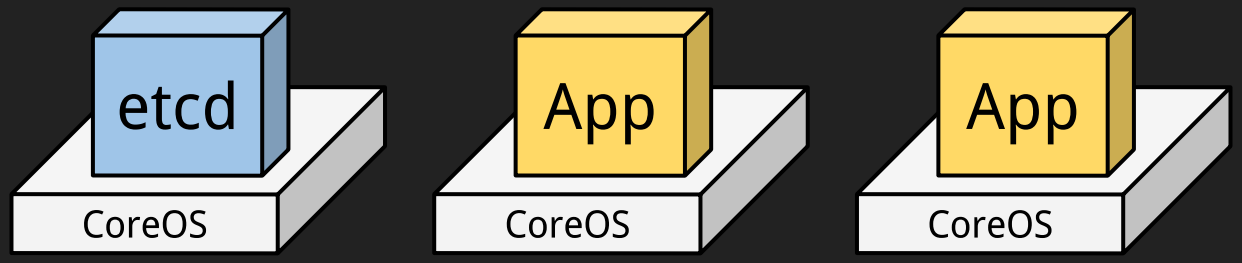
Consensus

(etcd)

etcd

Key-value store

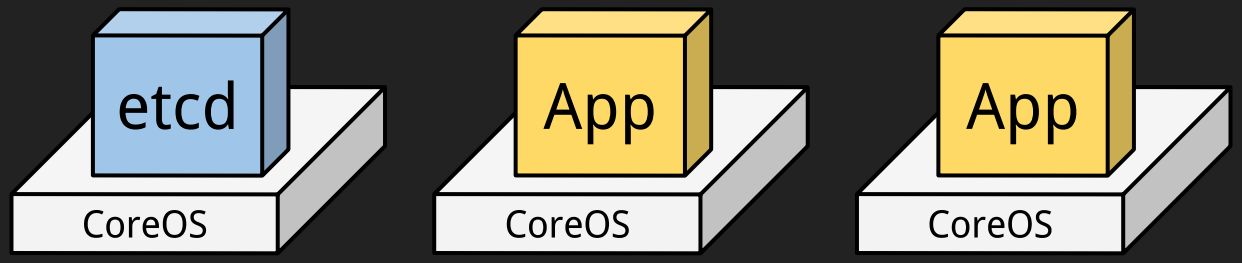
Centralized place to store cluster metadata


etcd
CoreOS
App
CoreOS
App
CoreOS

locksmith

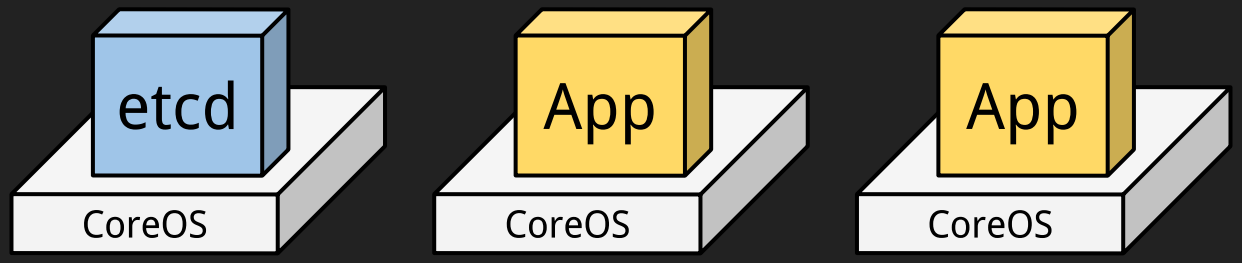
Must acquire a lock from etcd before rebooting

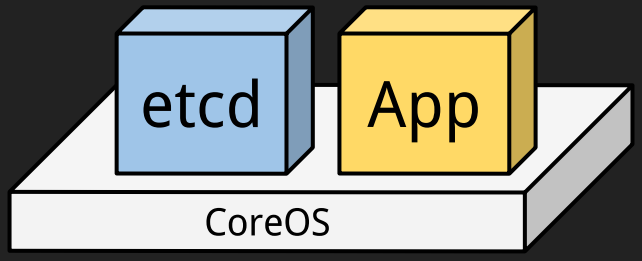
Release lock after successful boot

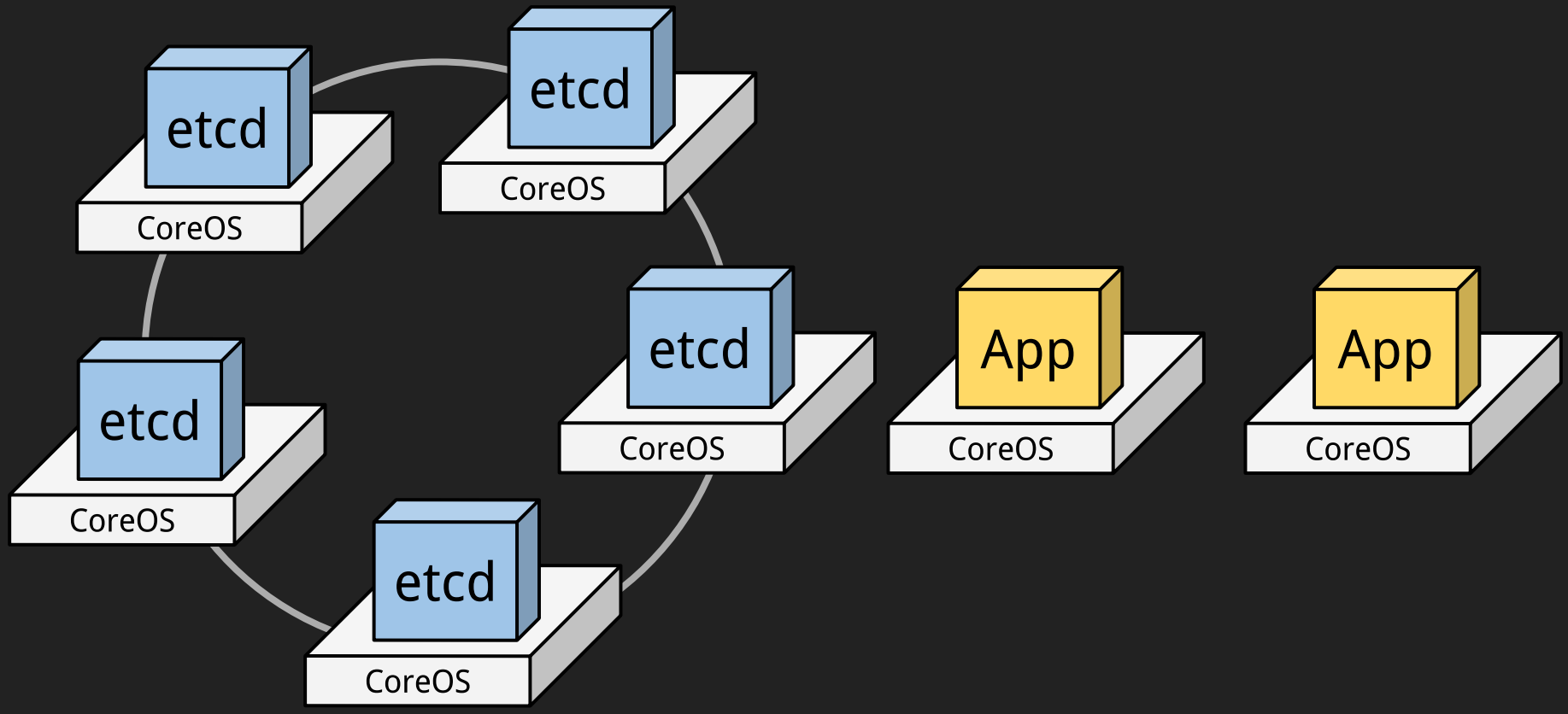


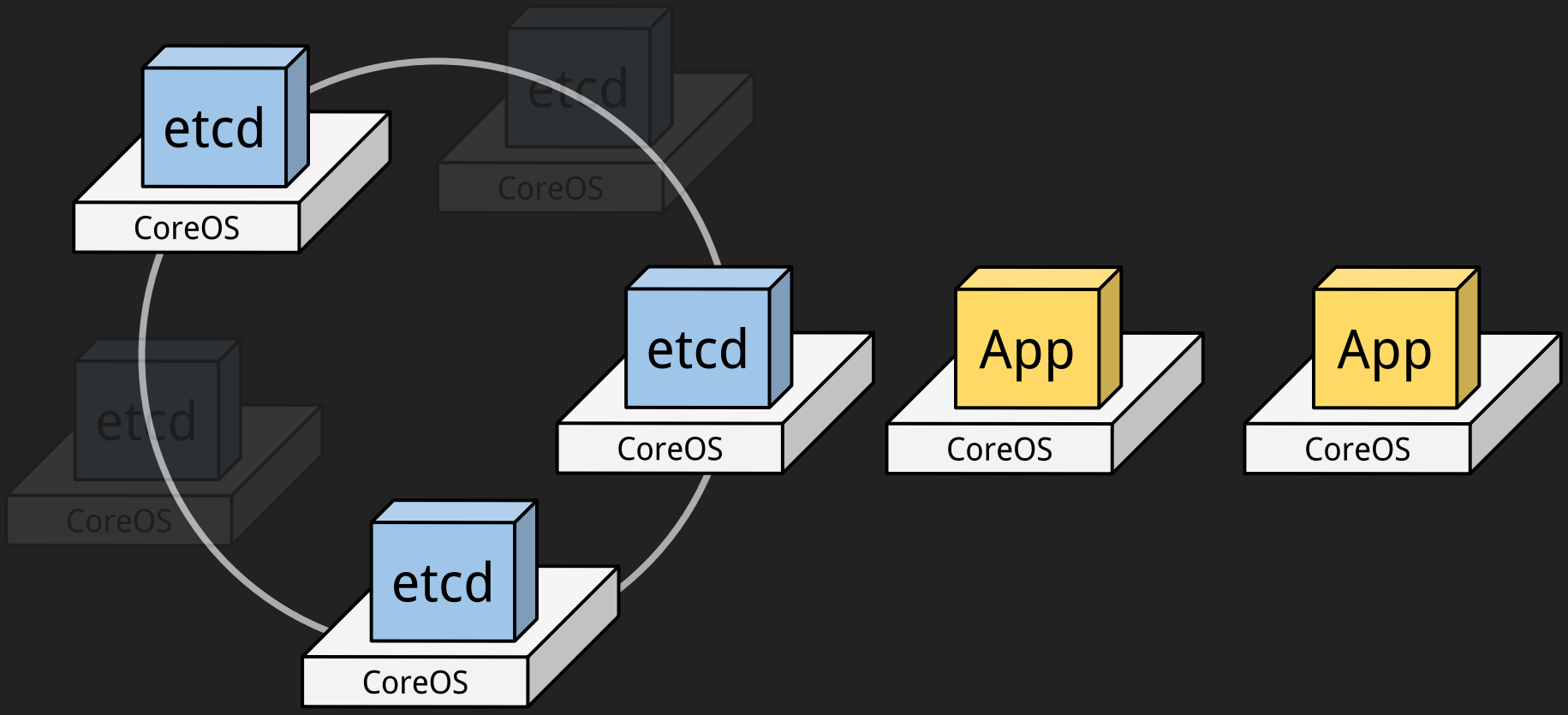
Demo

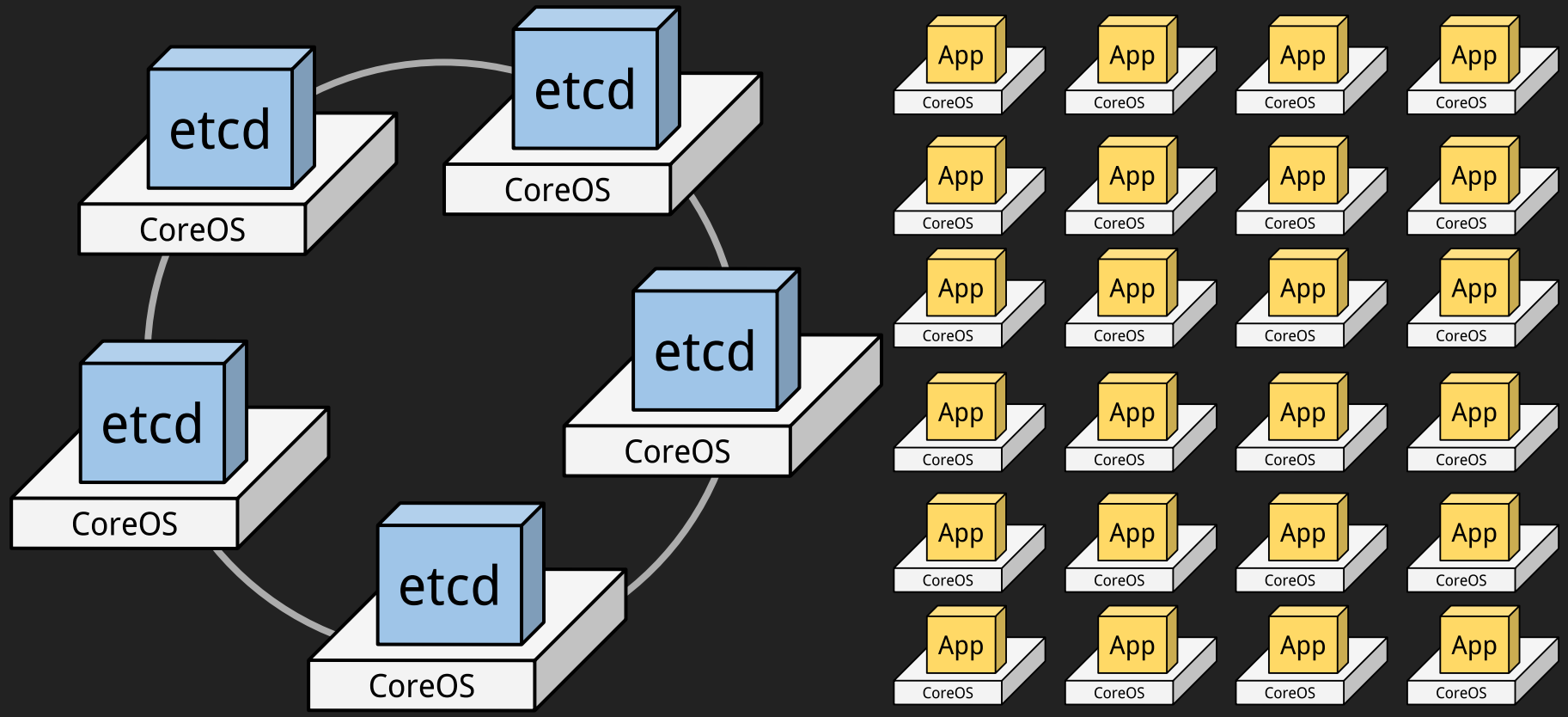
My etcd server is rebooting on its own, how do I keep my app online?

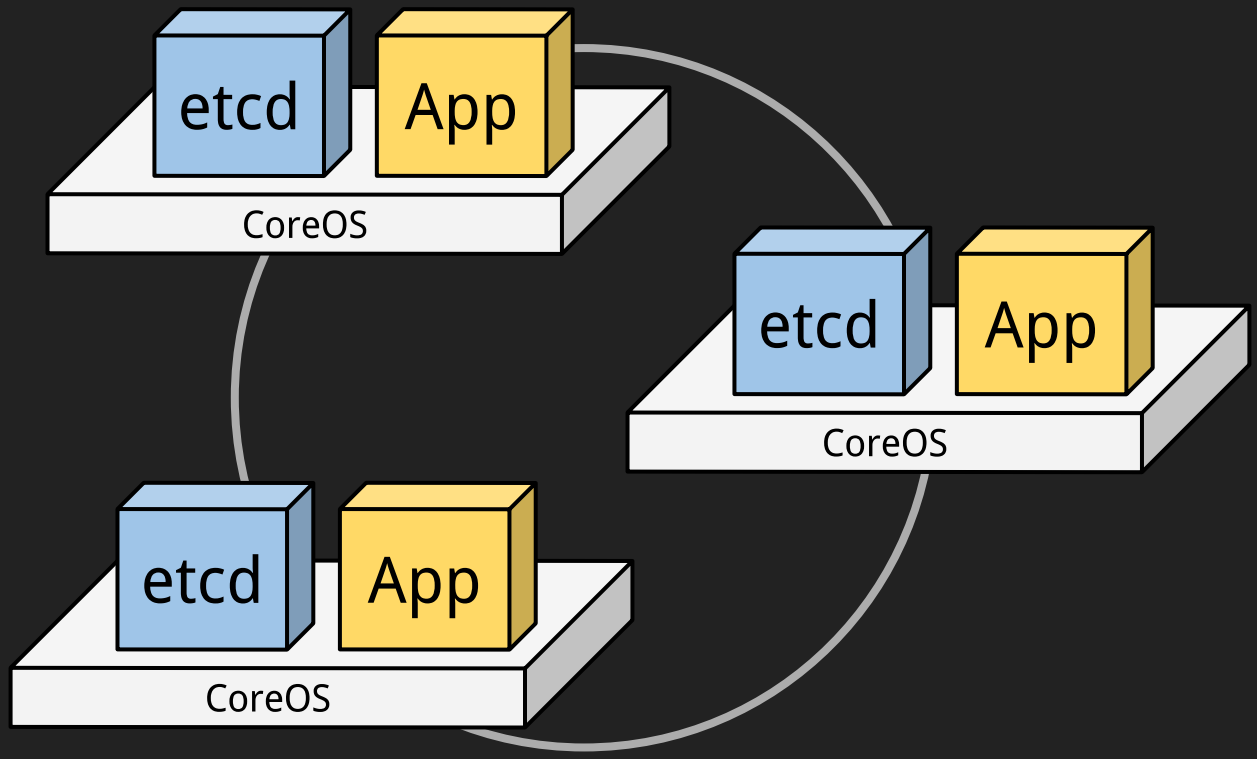

etcd
CoreOS
App
CoreOS
App
CoreOS


etcd
App
CoreOS


etcd
CoreOS
etcd
CoreOS
etcd
CoreOS
etcd
CoreOS
etcd
CoreOS
App
CoreOS
App
CoreOS


etcd
CoreOS
etcd
CoreOS
etcd
CoreOS
App
CoreOS
App
CoreOS
etcd
CoreOS
etcd
CoreOS


etcd
CoreOS
etcd
CoreOS
etcd
CoreOS
etcd
CoreOS
etcd
CoreOS
App
CoreOS


etcd
App
CoreOS
etcd
App
CoreOS
etcd
App
CoreOS

Etcd is Reusable

Google Kubernetes

Pivotal CloudFoundry

Mailgun Vulcand

Containerization

(Docker / rkt)

CoreOS is Minimal

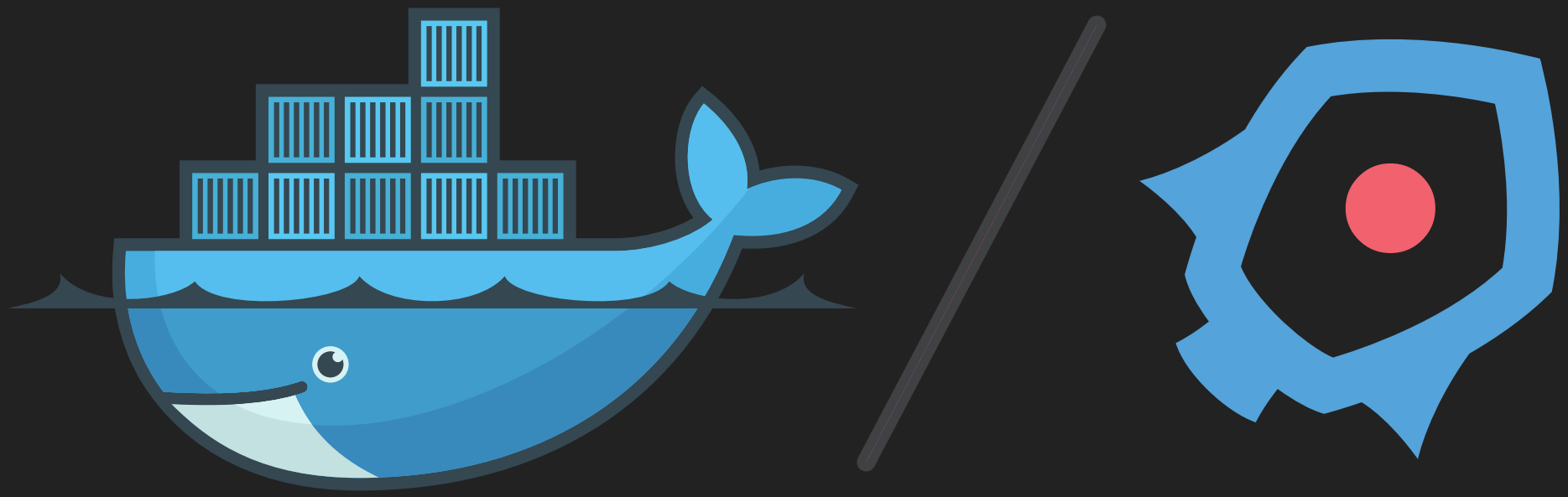
140 MB compressed

No Python, Perl, Ruby, or JavaScript

No package manager

No compiler

How do you run anything?



Containers

CoreOS includes Docker and rkt

Demo

Scheduling

(Fleet / Kubernetes)

Cluster-level init

“Always keep two of these running,
but not on the same machine.”

Schedulers

CoreOS includes Fleet, supports Kubernetes

Both independent components

Both built on etcd

Fleet

Clustered interface for systemd

Systemd Unit Files

```
[Unit]
Description=My App
After=docker.service
Requires=docker.service

[Service]
ExecStartPre=-/usr/bin/docker kill my-app-%i
ExecStartPre=-/usr/bin/docker rm my-app-%i
ExecStart=/usr/bin/docker run -rm --name my-app-%i -p 80:8080 callaha
ExecStop=/usr/bin/docker stop my-app-%i

[X-Fleet]
Conflicts=my-app@*.service
```

X-Fleet attributes

- `Conflicts`
- `MachineOf`
- `MachineID`
- `MachineMetadata`
- `Global`

Demo

Design Considerations

Minimize state

Build “Twelve-Factor Apps”

What about Databases?
Load balancers?

We did it!

We built a platform that is

self-updating, self-organizing, and self-healing.

We used

1. An OS with automatic, atomic, whole-system updates.
2. Portable, isolated containers for our applications.
3. Multiple servers in a coordinated cluster.
4. A scheduler to distribute jobs across machines.

Now it’s your turn!

Many supported platforms

- Local VMs (Vagrant)
- Azure, EC2, GCE, RackSpace
- DigitalOcean

$40 credit on DigitalOcean:
“SAMMYLOVESPYCON”

Questions?

dcallahan@mozilla.com

@callahad

github.com/callahad/pycon2015-coreos

“SAMMYLOVESPYCON”