CoreOS

About Me

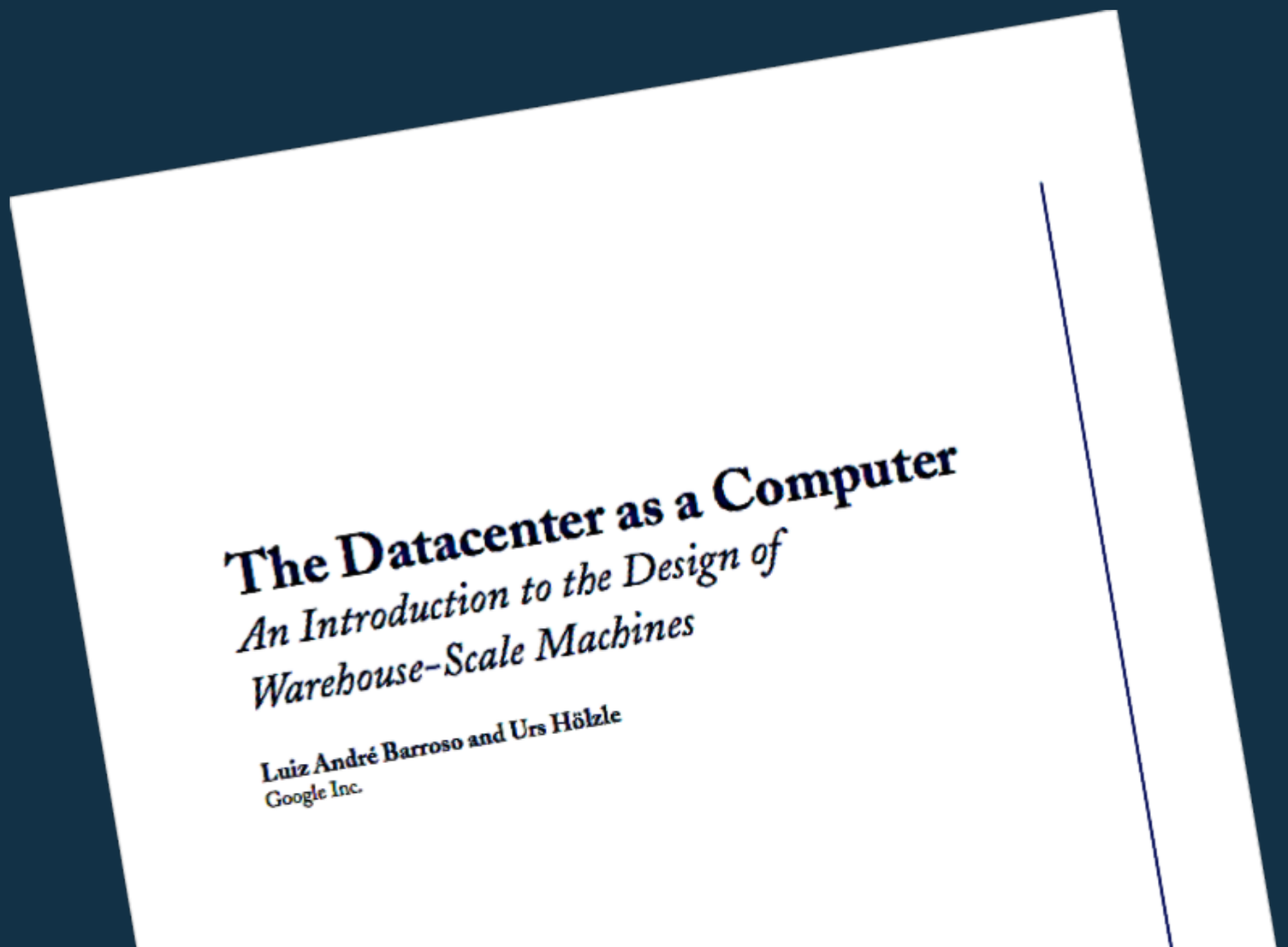
CTO/CO-FOUNDER

systems engineer

@brandonphilips

github.com/philips

Why build CoreOS?



The Datacenter as a Computer

An Introduction to the Design of Warehouse-Scale Machines

Luiz André Barroso and Urs Hölzle
Google Inc.

containers

run and isolate apps

containers

what is it exactly?

libc

python

django

app.py

$ /usr/bin/python run app.py

libc
python
django
app.py

example.com/myapp

libc
python
django
app.py

$ container fetch example.com/myapp

$ container run example.com/myapp

pid ns

isolated pid 1

user ns

isolated uid 0

network ns

isolated netdev

mount ns

isolated /

cgroups

manage resources

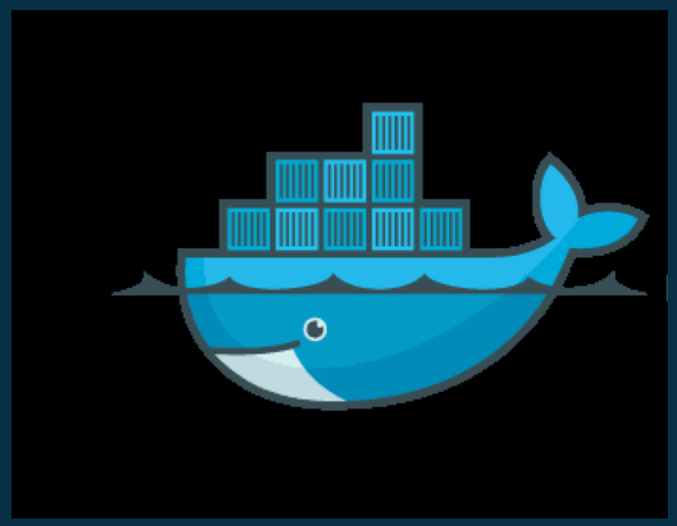
cgroups

count resources

cAdvisor

cgroups

limit resources



docker engine

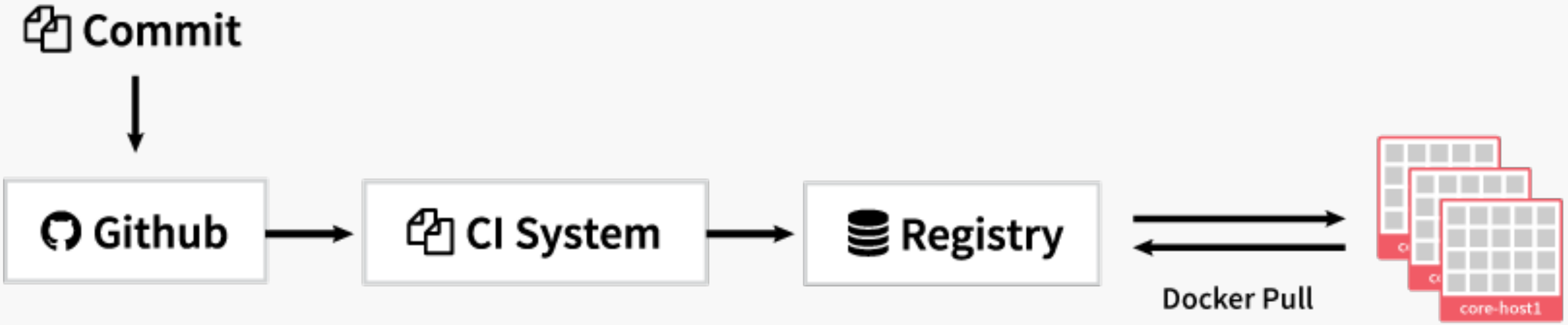
Rocket

google lmctfy
cloud foundry garden
mesos containers

lxc
systemd-nspawn

containers

how are they created?


Commit
Github
CI System
Registry
Docker Pull
core-host1

containers

super-powers

App independence from the OS.

System to get container to the server.

Resource isolation between apps.

OS

reduced API contracts

kernel
systemd
etcd
ssh
docker

python
java
nginx
mysql
openssl

distro distro distro

kernel
systemd
etcd
ssh
docker

o distro distro distro distro distro distro distro distro distr

python
java
nginx
mysql
openssl

app

kernel
systemd
etcd
ssh
docker

o distro distro distro

manual

updates

automatic

updates

automatic updates

auto updates

atomic with rollback

Data

Update

A

B

Data

A

B

OS

super-powers

Opportunity for automatic updates.

Consistent set of software across hosts.

Base OS independent from app.

clustering

design for host failure

etcd

/etc

distributed

open source software

sequentially consistent

exposed via HTTP

runtime reconfigurable

-X GET

Get Wait

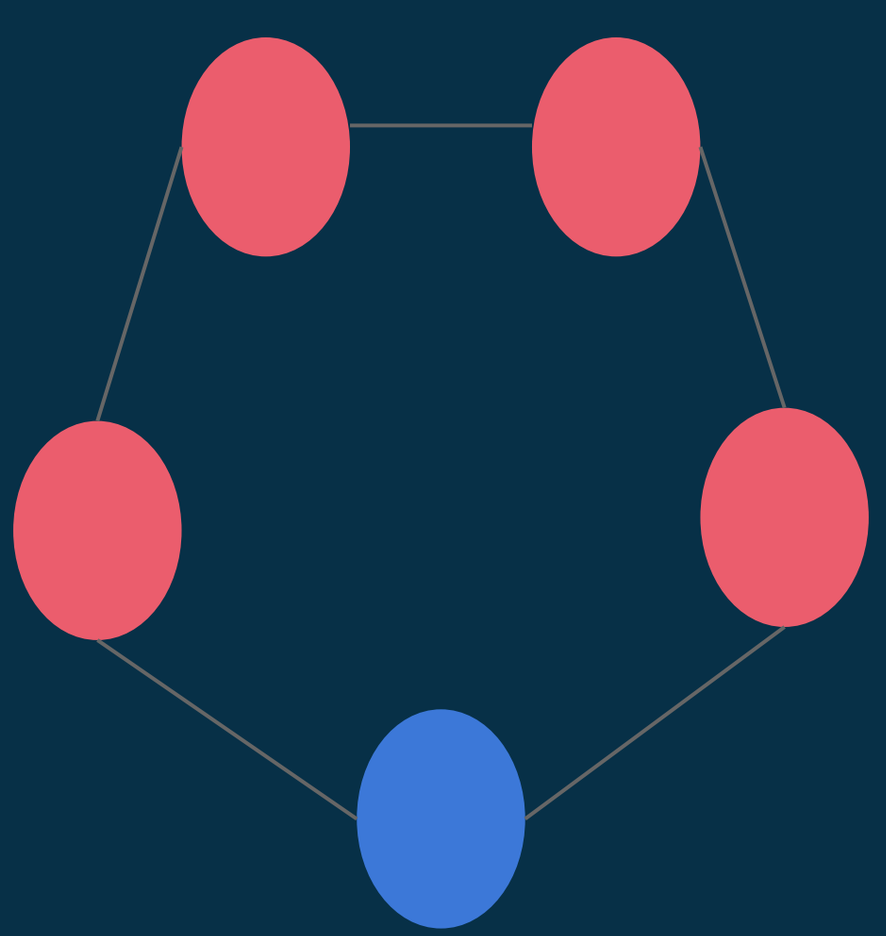
-X PUT

Put Create CAS

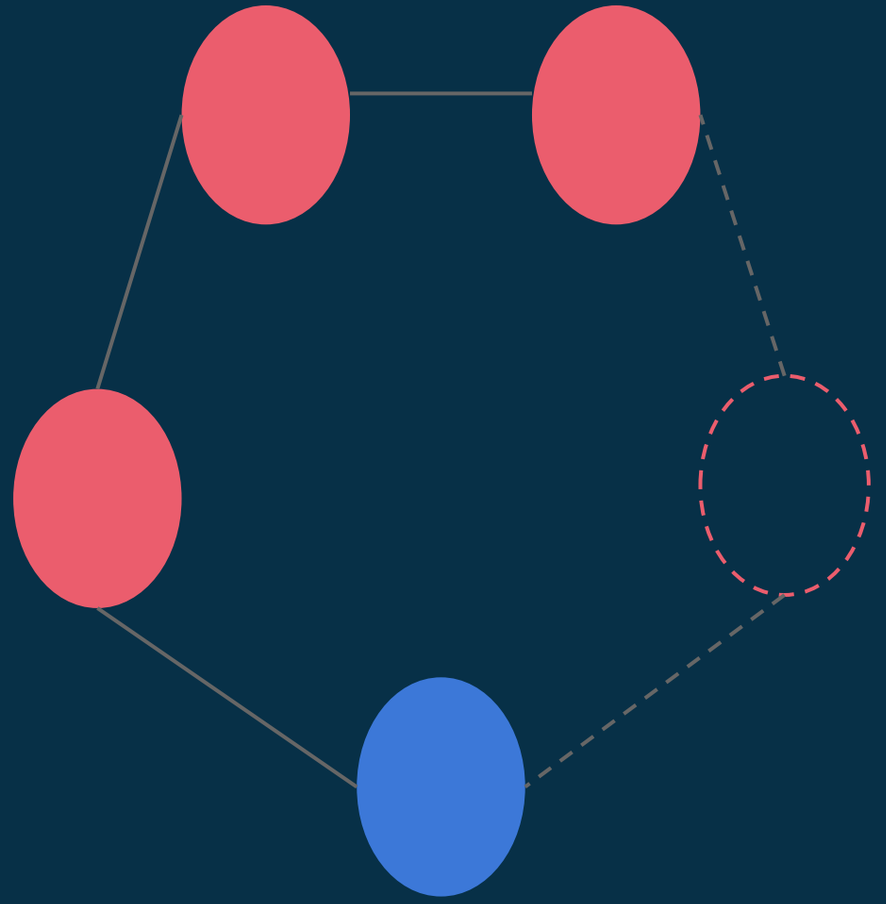
-X DELETE

Delete CAD

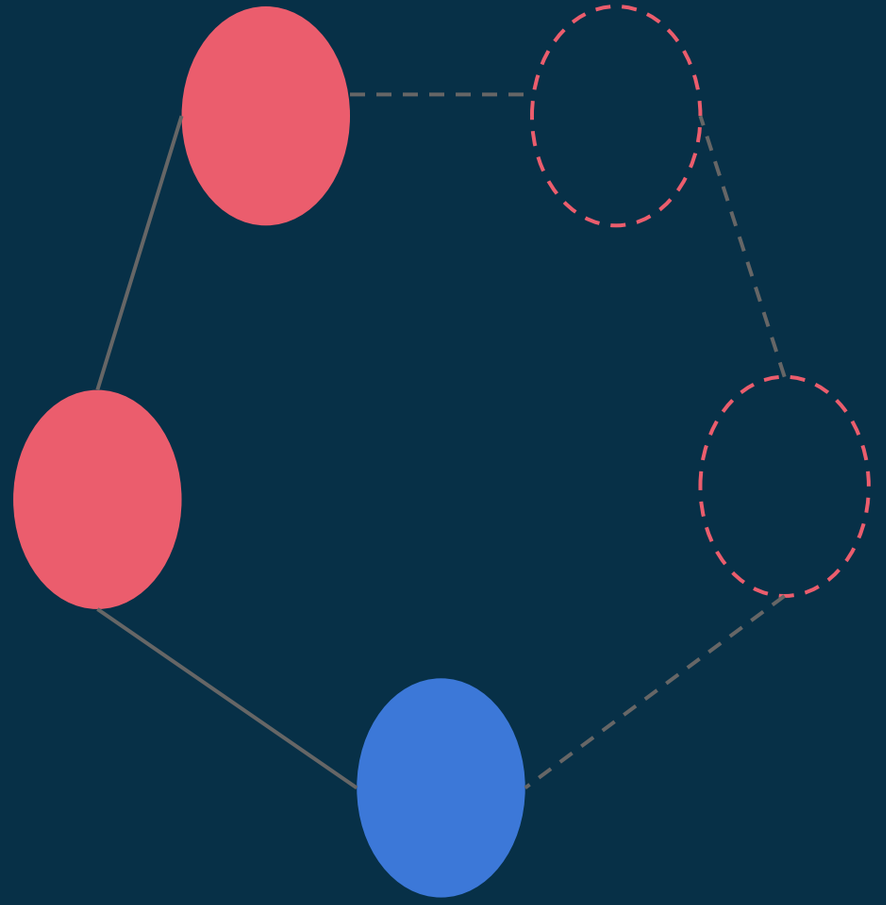
Available



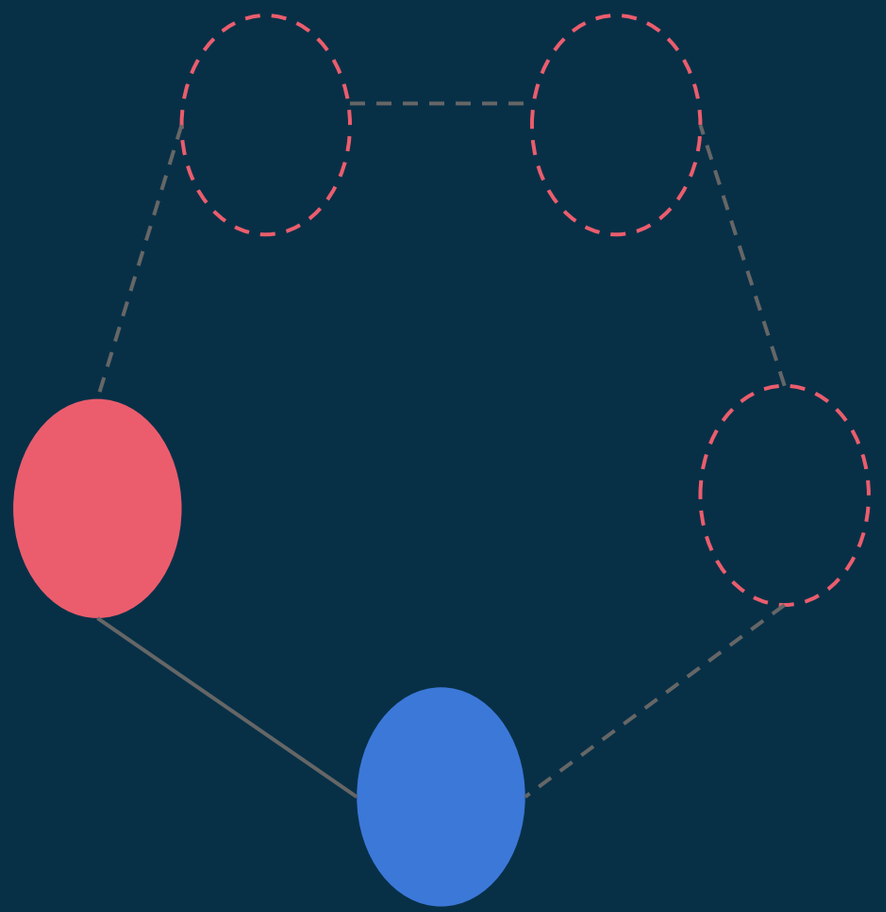
Available



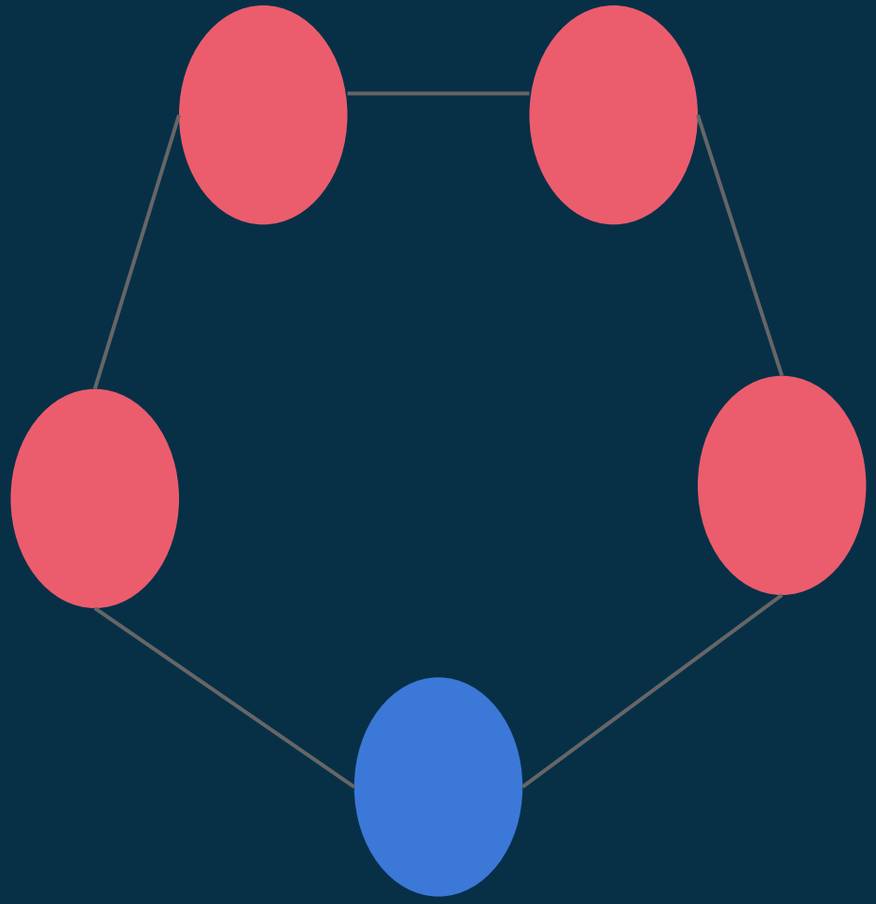
Available



Unavailable



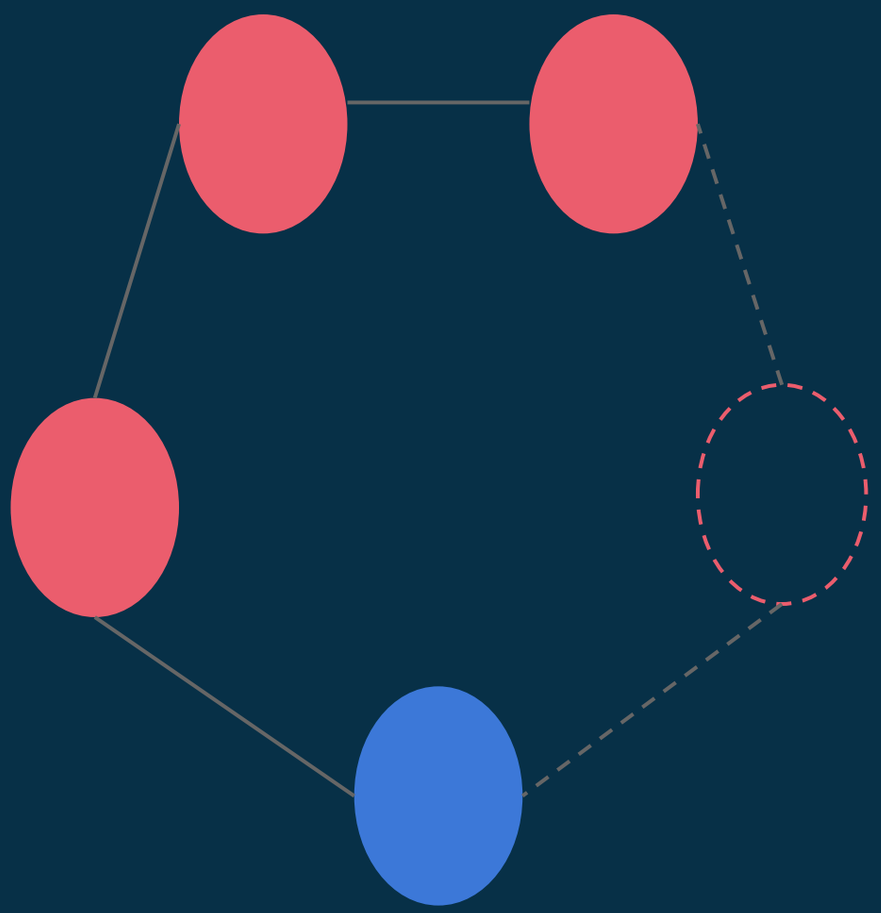
Available

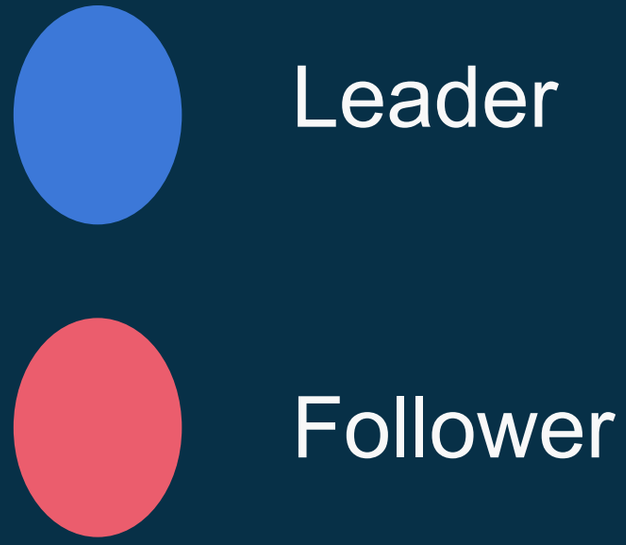


Leader

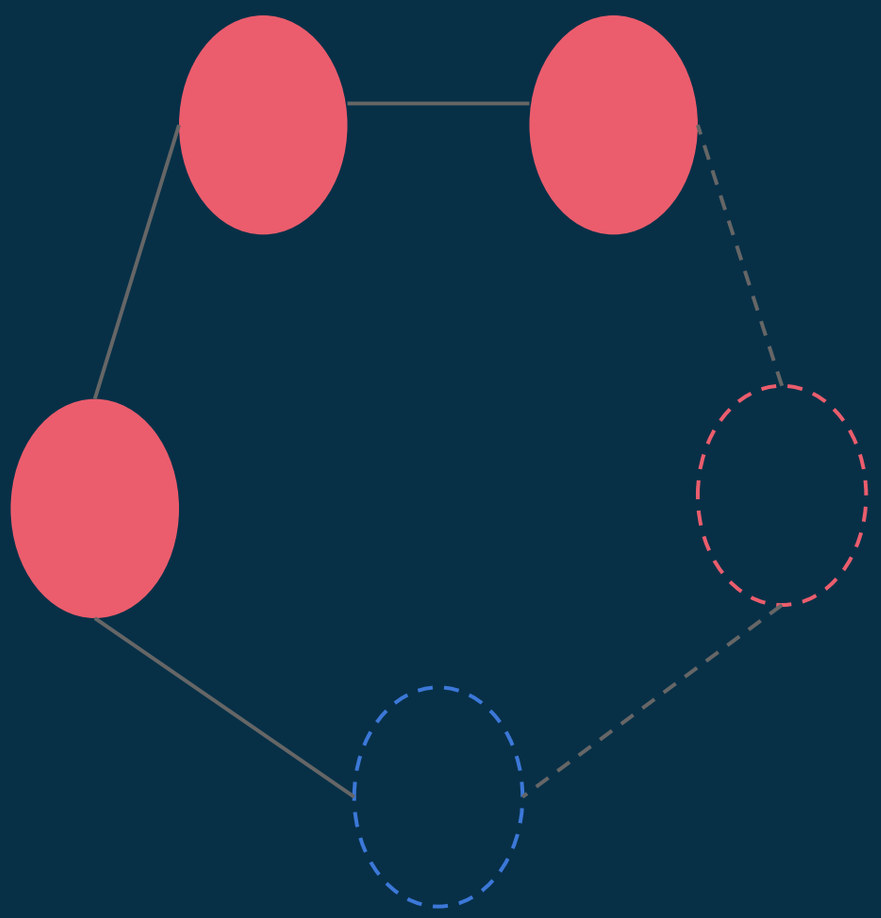
Follower

Available





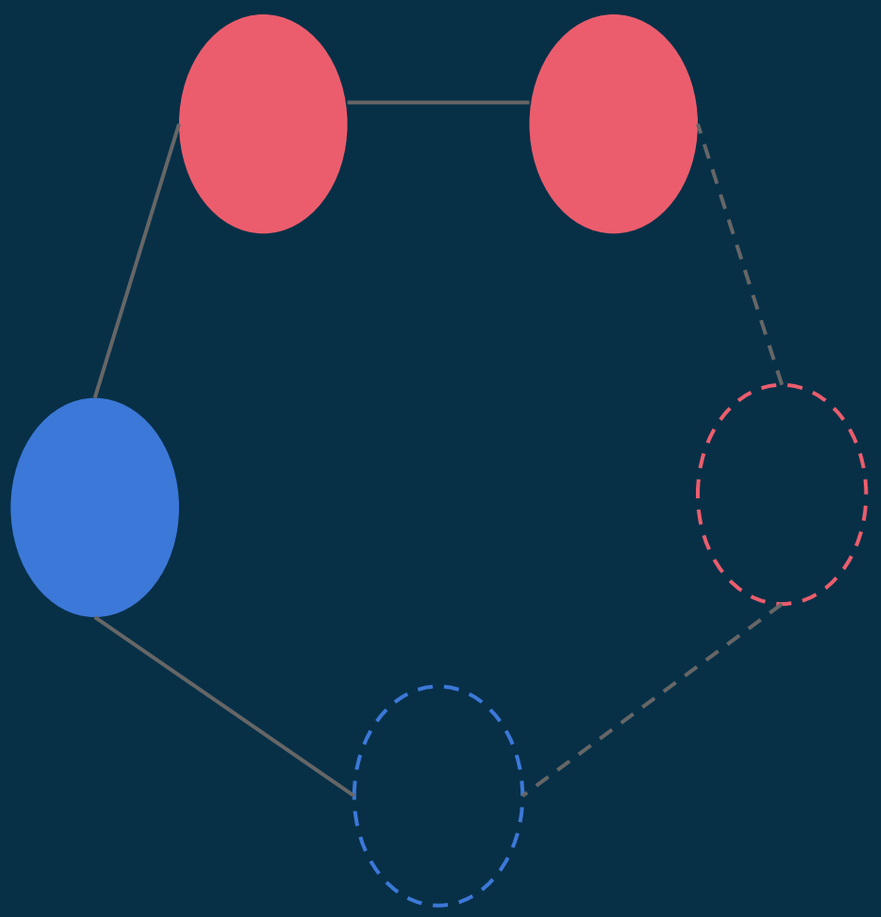
Temporarily Unavailable

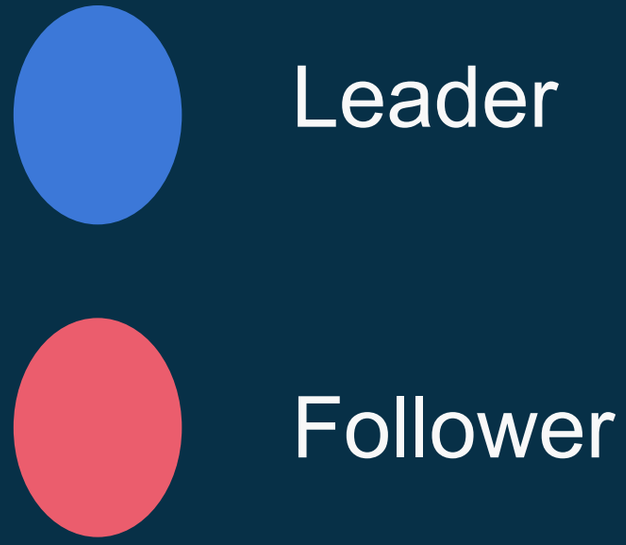


Leader

Follower

Available





etcd

super-powers

Share configuration data across hosts.

Resilient to host failures.

Designed for consistency across hosts.

scheduling

getting work to servers

You

Scheduler API

Scheduler

Machine(s)

```
$ cat foo.service
[Service]
ExecStart=/usr/bin/sleep 500
```

```
$ fleetctl start foo.service
Job foo.service launched on
e1cd2bcd.../172.17.8.101
```

```
while true {
  todo = diff(desState, curState)
  schedule(todo)
}
```

```
while true {
  todo = diff(desState, curState)
  schedule(todo)
}
```

```
while true {
  todo = diff(desState, curState)
  schedule(todo)
}
```

```
while true {
  todo = diff(desState, curState)
  schedule(todo)
}
```

job scheduling

fleet

mesos

kubernetes

swarm

coordination

locksmith

scheduling

super-powers

Think about app capacity first.

Take advantage of compute resources.

Build for resilience to host failure.

service discovery

skydns, discoverd, confd

service discovery

magic proxies

OS

Containers

Cluster Configuration

Job Scheduling

Service Discovery

Wednesday 6:00pm AKL Continuous Delivery Meetup.
CoreOS: An Introduction

Thursday 6:00 PM Go AKL Meetup
etcd (and maybe rocket)

Friday 10:40am LCA
CoreOS Tutorial

CoreOS