@coreoslinux

About Me

CTO/CO-FOUNDER

systems engineer

@brandonphilips

github.com/philips

etcd

/etc
distributed

open source software
failure tolerant
durable
watchable
exposed via HTTP
runtime reconfigurable

Data Store API

-X GET

- Get Wait

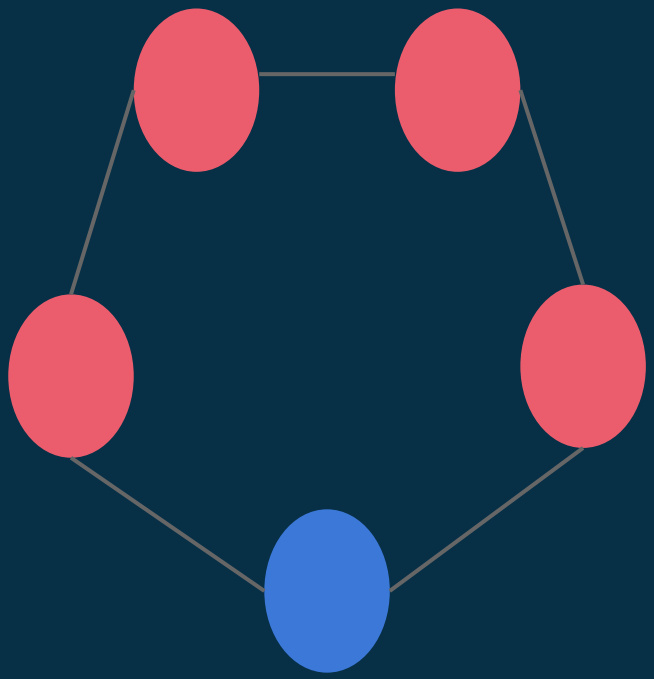
-X PUT

- Put Create CAS

-X DELETE

- Delete CAD

etcd Cluster

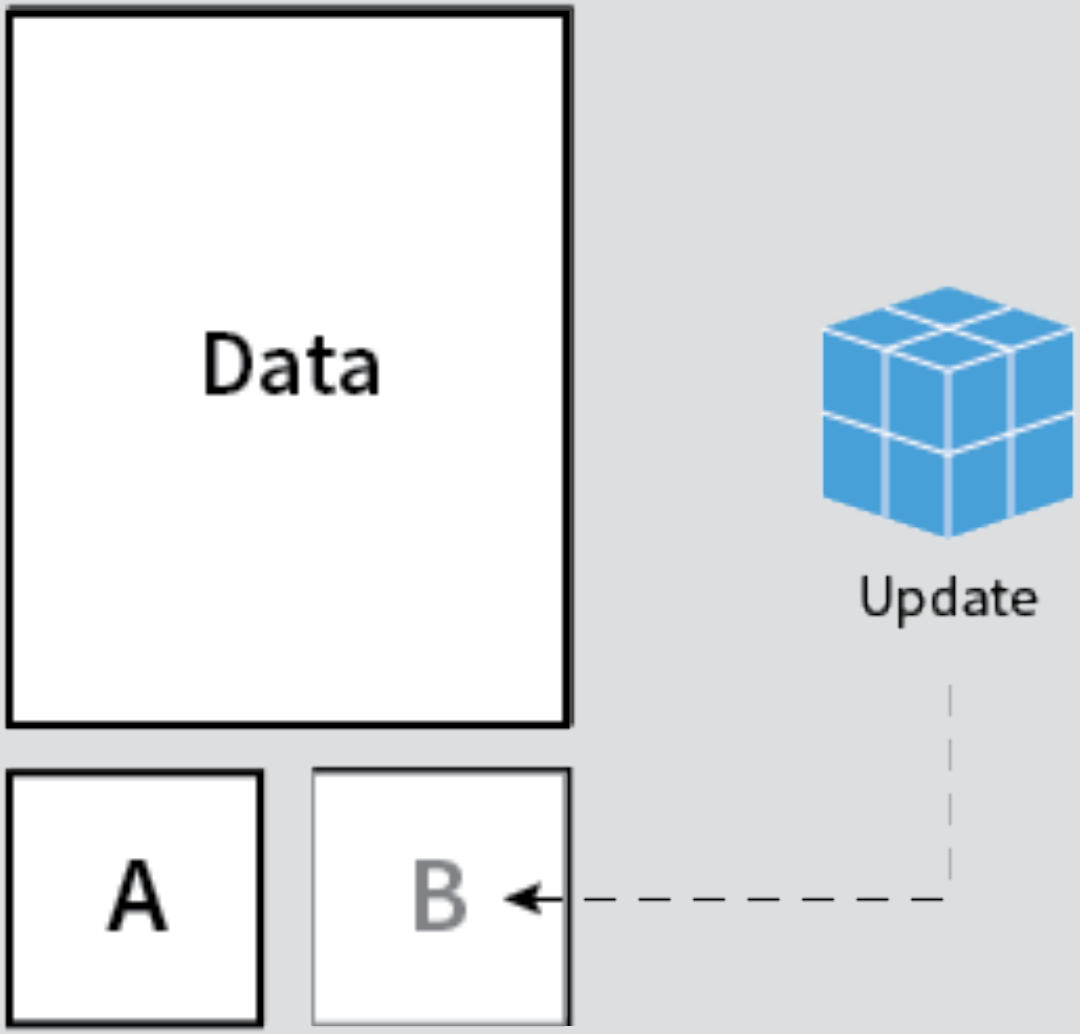


Leader

Follower

Applications

locksmith


Data
Update
A
B


Data
A
B

Cluster Wide Reboot Lock

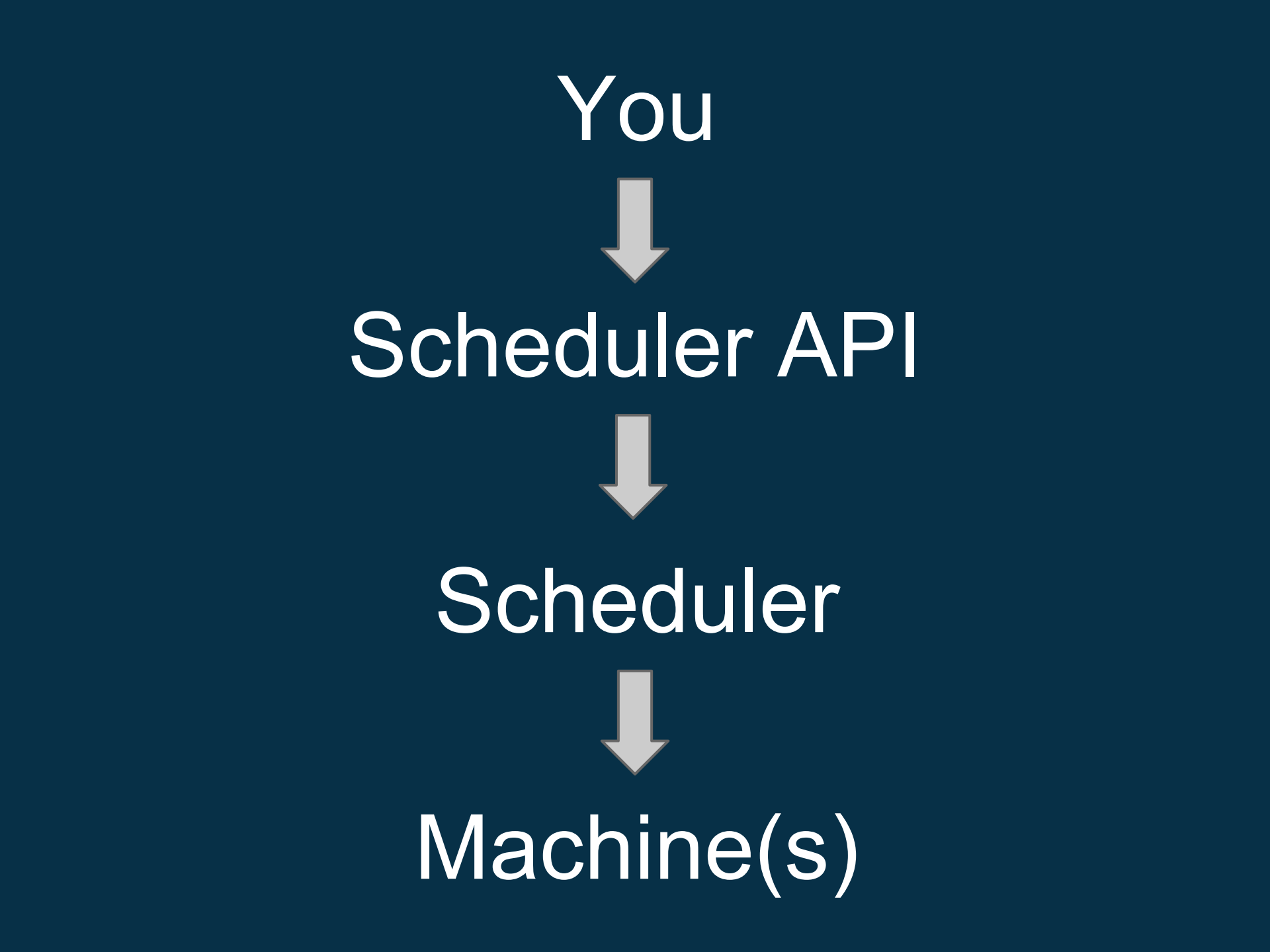
1. Need reboot to reboot? Decrement the semaphore key atomically with etcd.

2. manager.Reboot() and wait...

3. After rebooting increment the semaphore key in etcd atomically.

Applications

kubernetes and fleet


You
Scheduler API
Scheduler
Machine(s)

Cluster Work Scheduling

1. Cluster API writes desired work into etcd keyspace.

2. Agents running on individual machines pick up work assigned to them.

3. Agents report where work is running and current status.

Applications

vulcan, confd, dns and distributed git

Example Leader Election

using TTL and atomic operations

PUT /6eadeac2d/f1d2d2f924e98
‘http://10.1.2.3:7001’

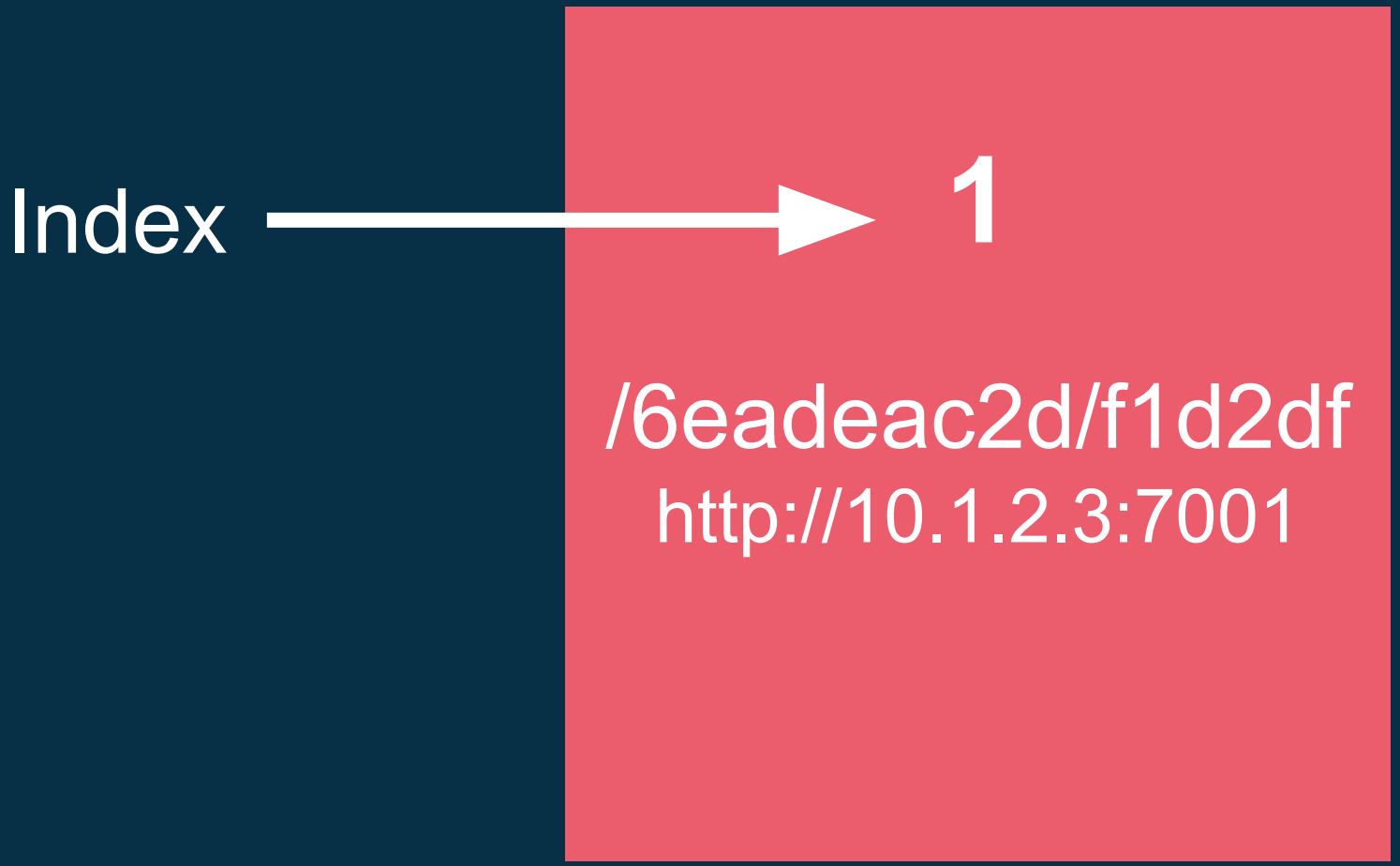
PUT /6eadeac2d/f1d2d2f924e98
‘http://10.1.2.3:7001’

Entry

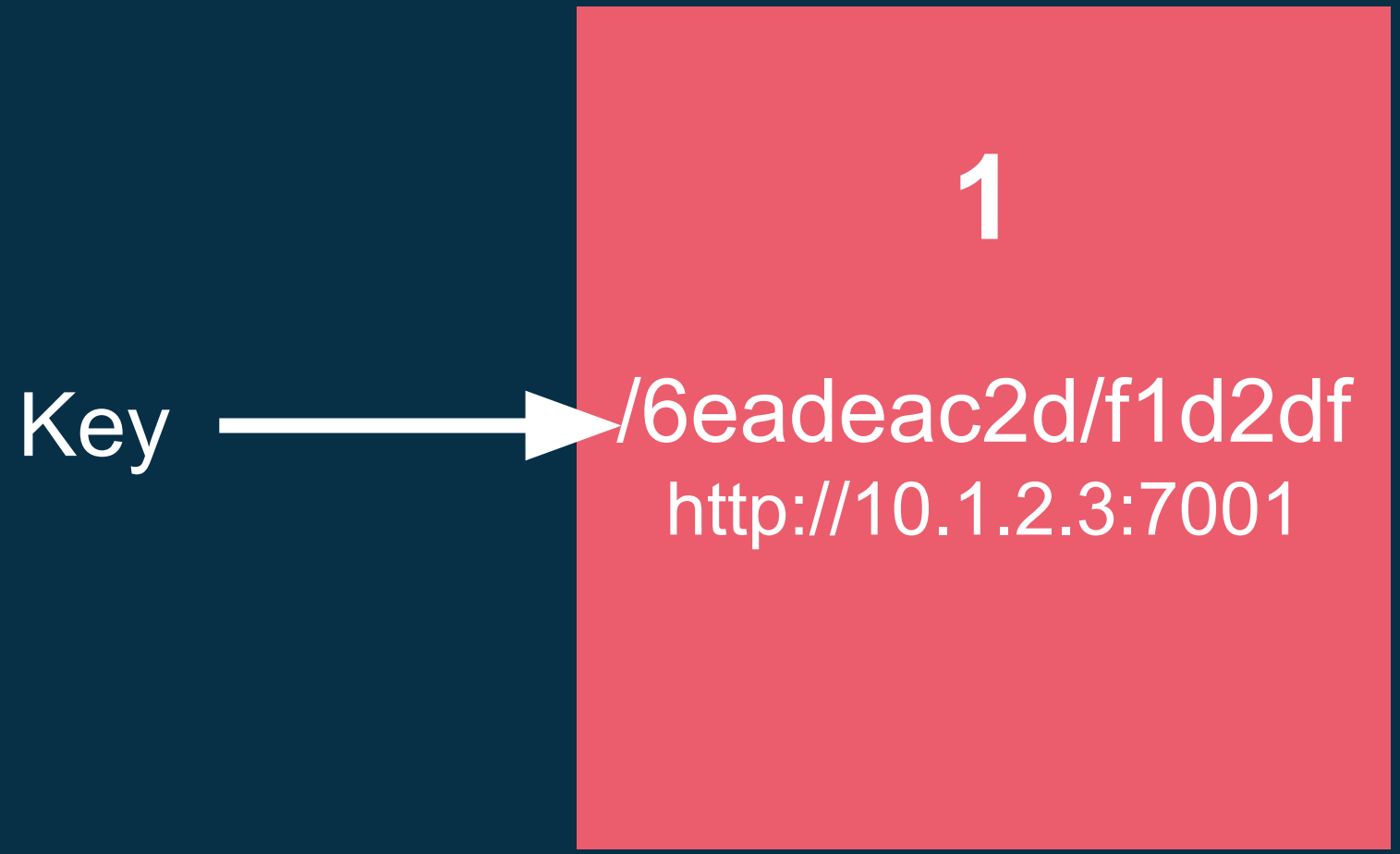
1

/6eadeac2d/f1d2df
http://10.1.2.3:7001

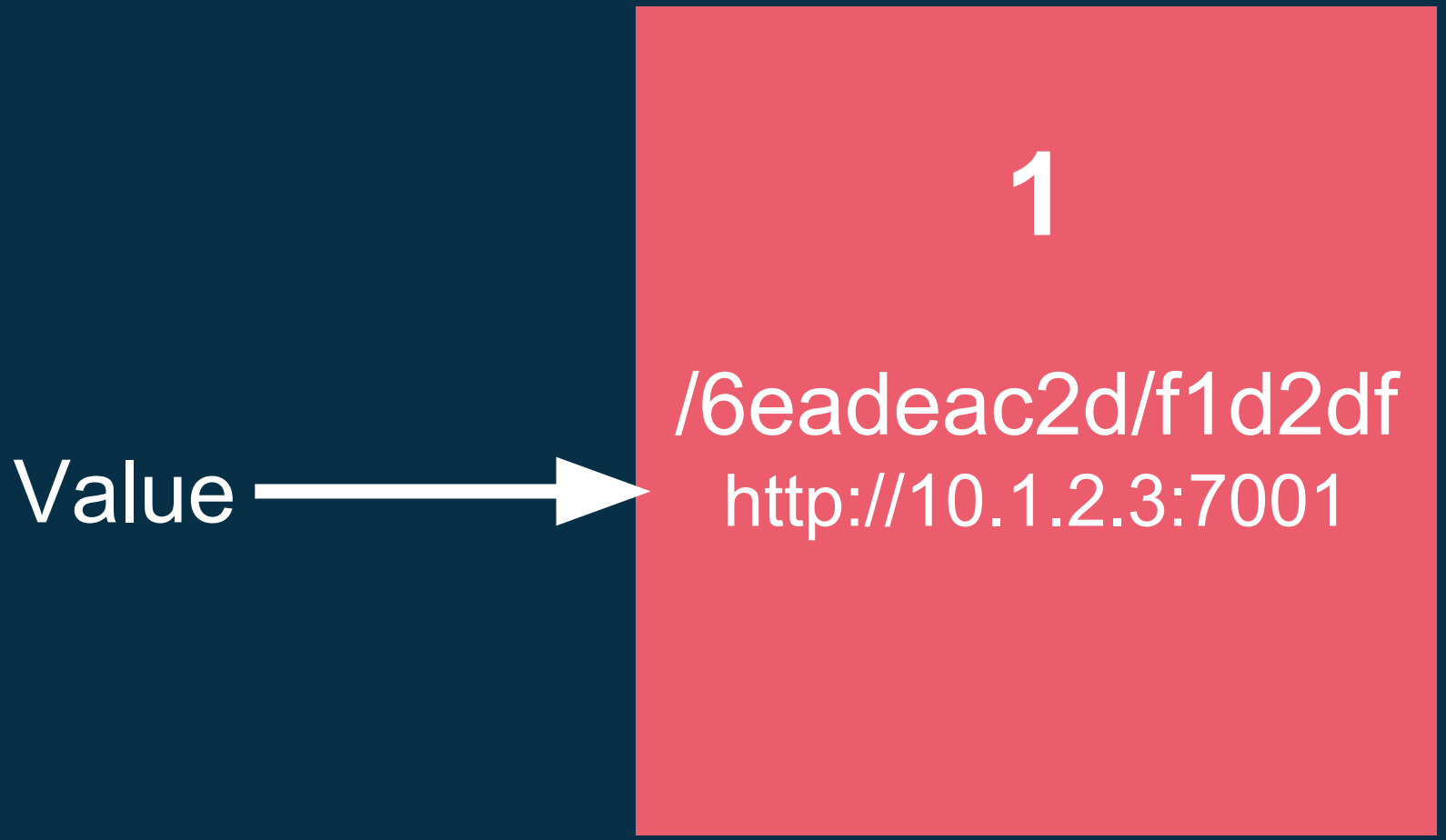
PUT /6eadeac2d/f1d2d2f924e98
‘http://10.1.2.3:7001’



PUT /6eadeac2d/f1d2d2f924e98
‘http://10.1.2.3:7001’

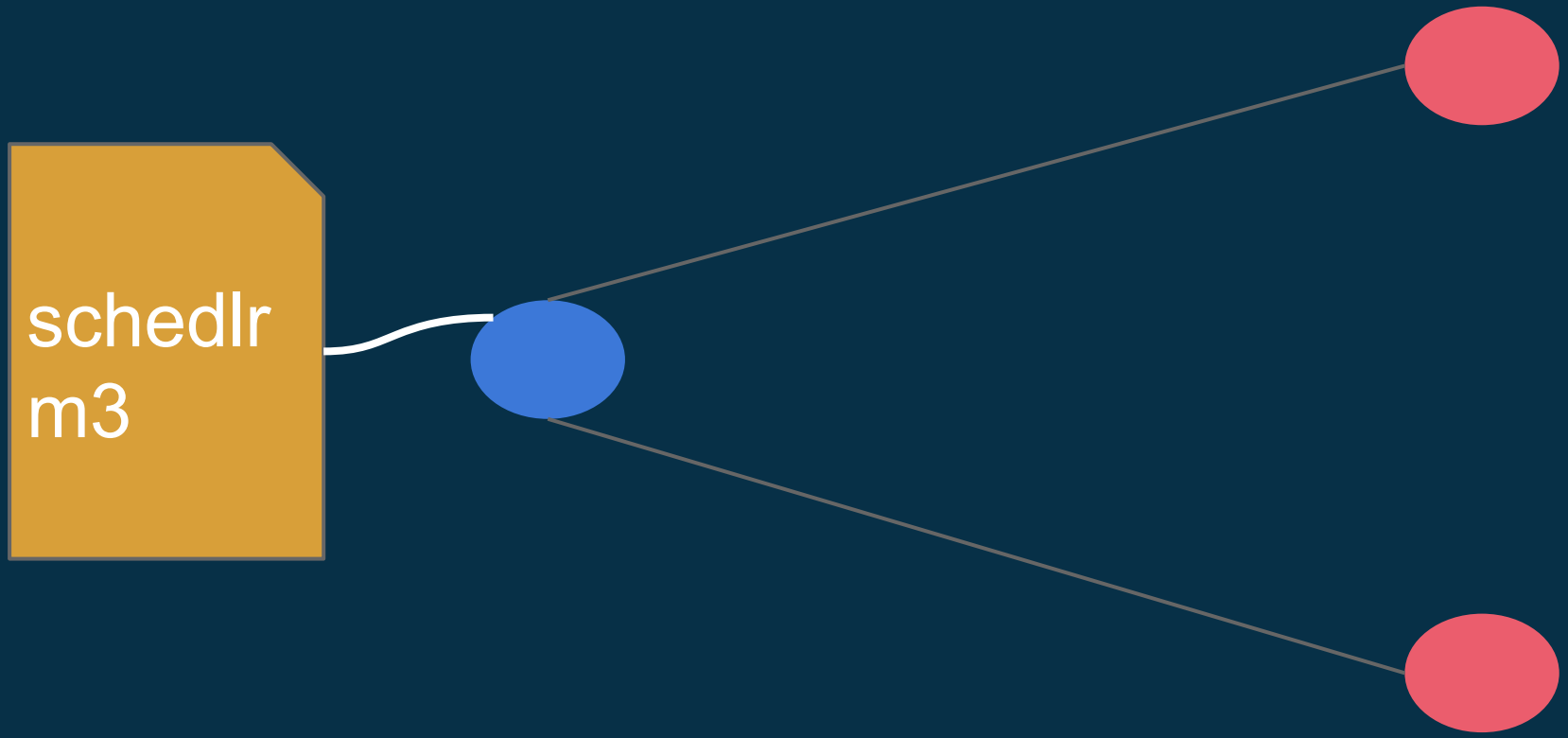


PUT /6eadeac2d/f1d2d2f924e98
‘http://10.1.2.3:7001’

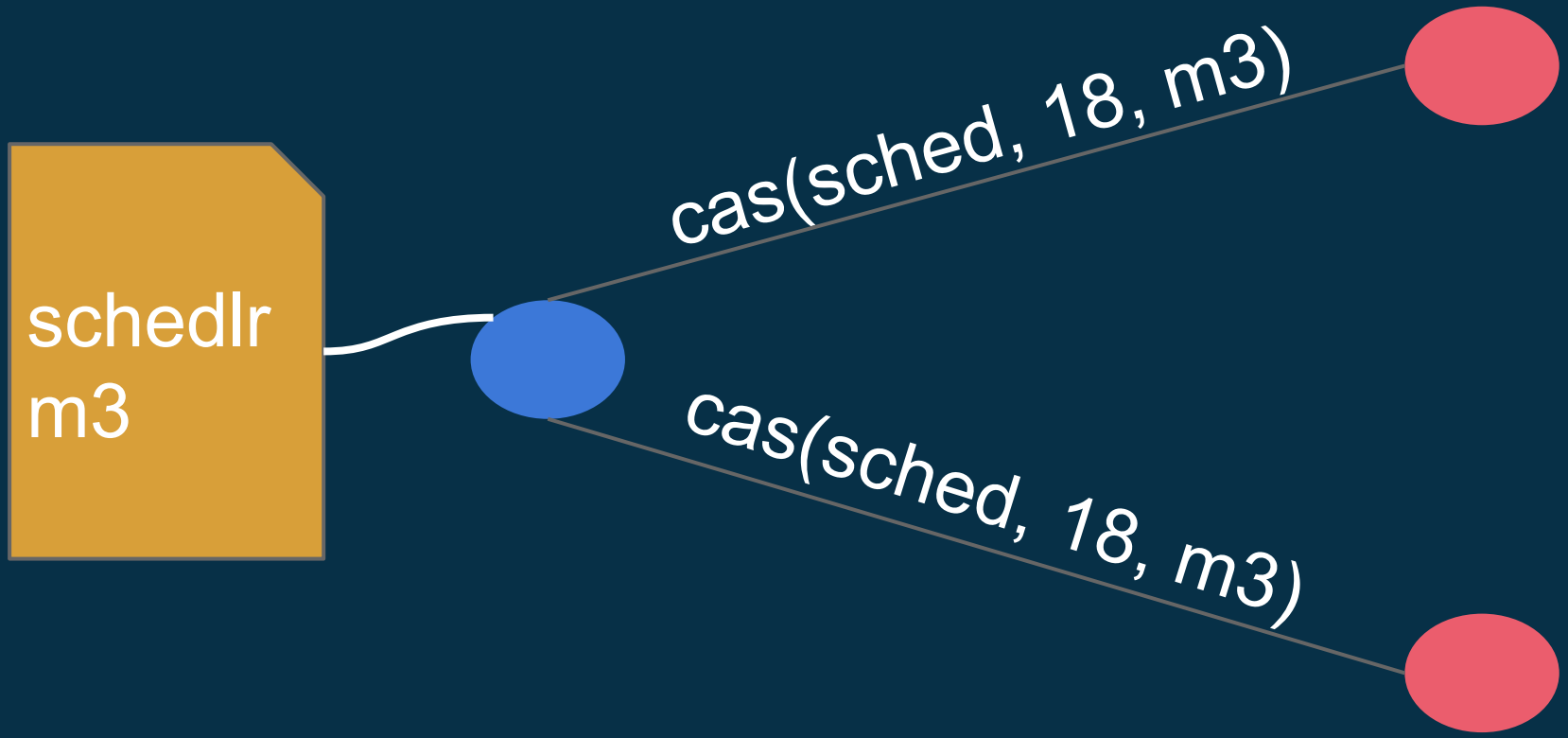


Idx	Key	Value	Expiration Time
18	sched	m3	Sept 18 2:11:30

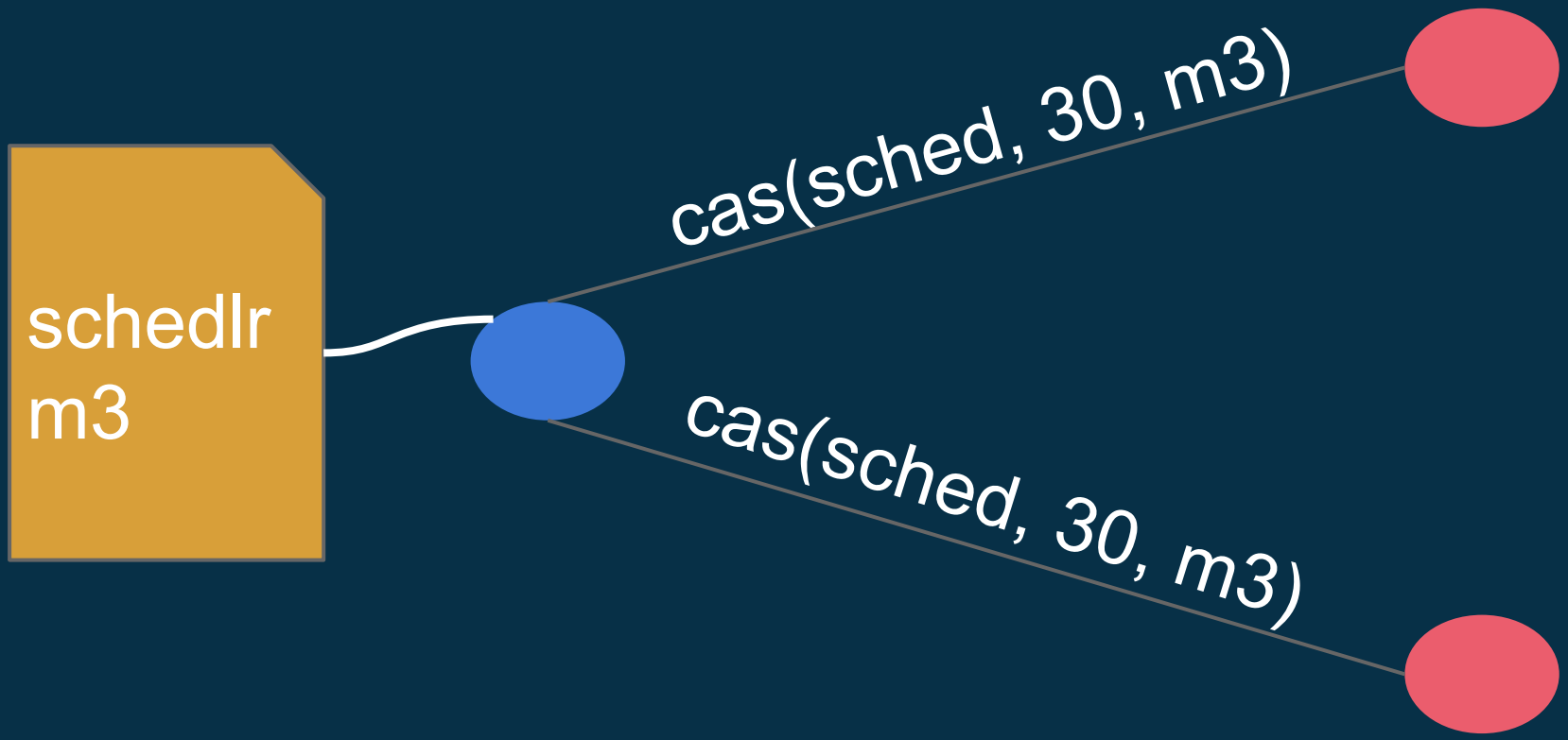
Idx	Key	Value	Expiration Time
18	sched	m3	Sept 18 2:11:30



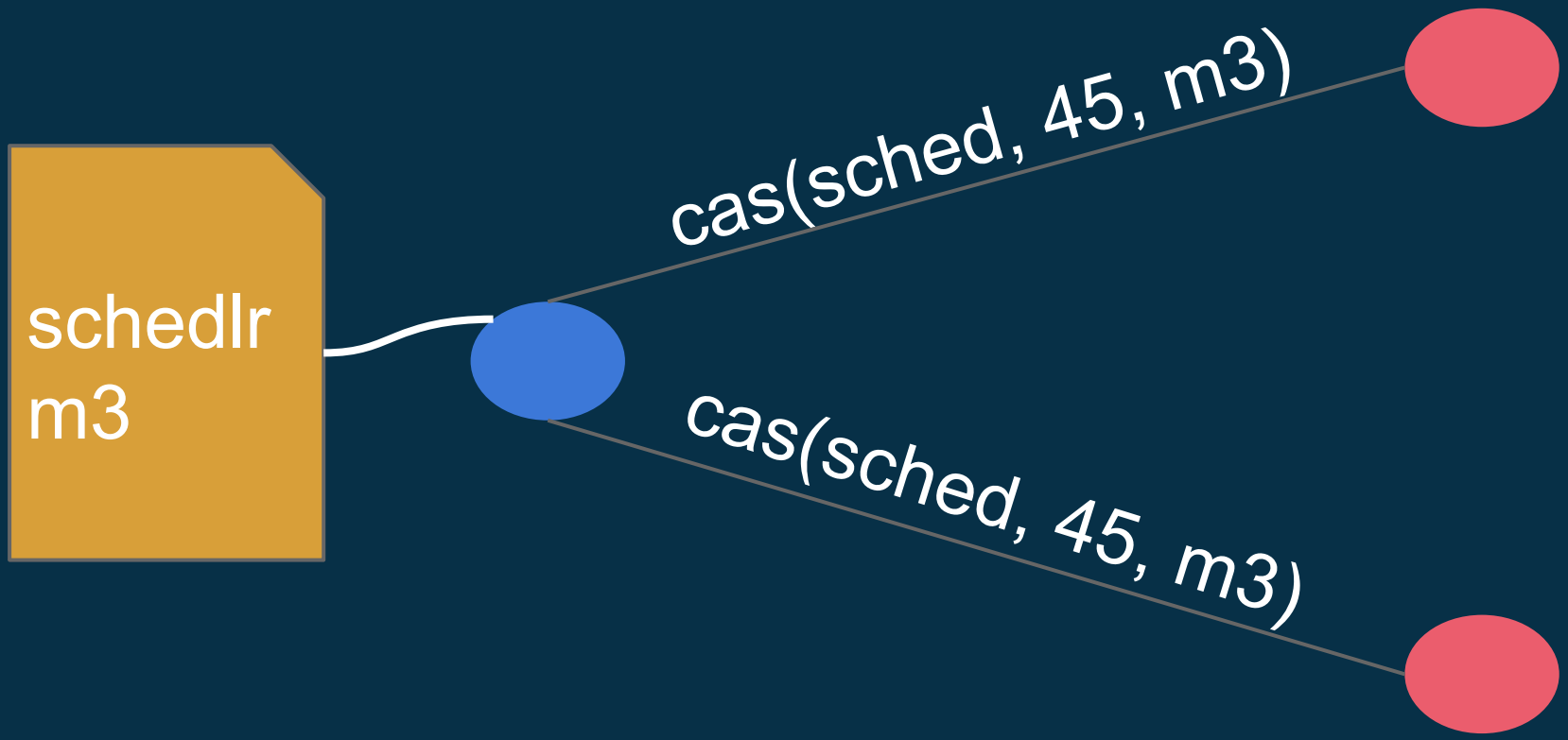
Idx	Key	Value	Expiration Time
18	sched	m3	Sept 18 2:11:30



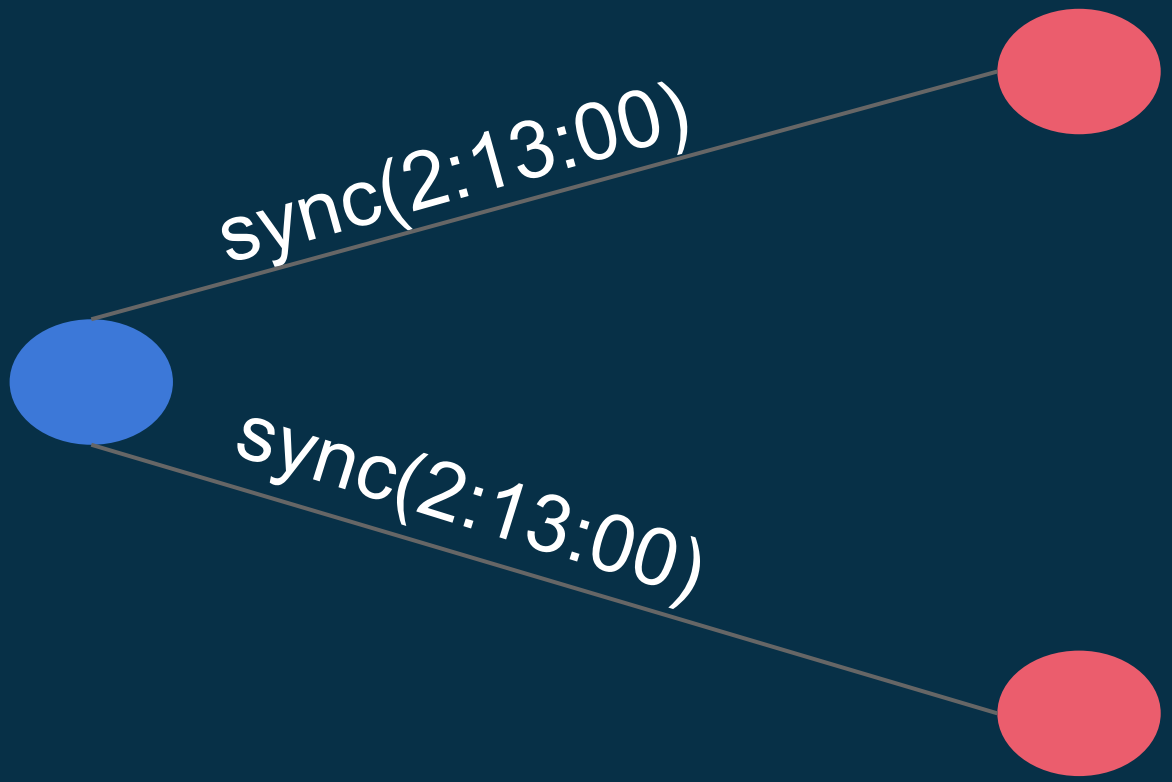
Idx	Key	Value	Expiration Time
30	sched	m3	Sept 18 2:12:50



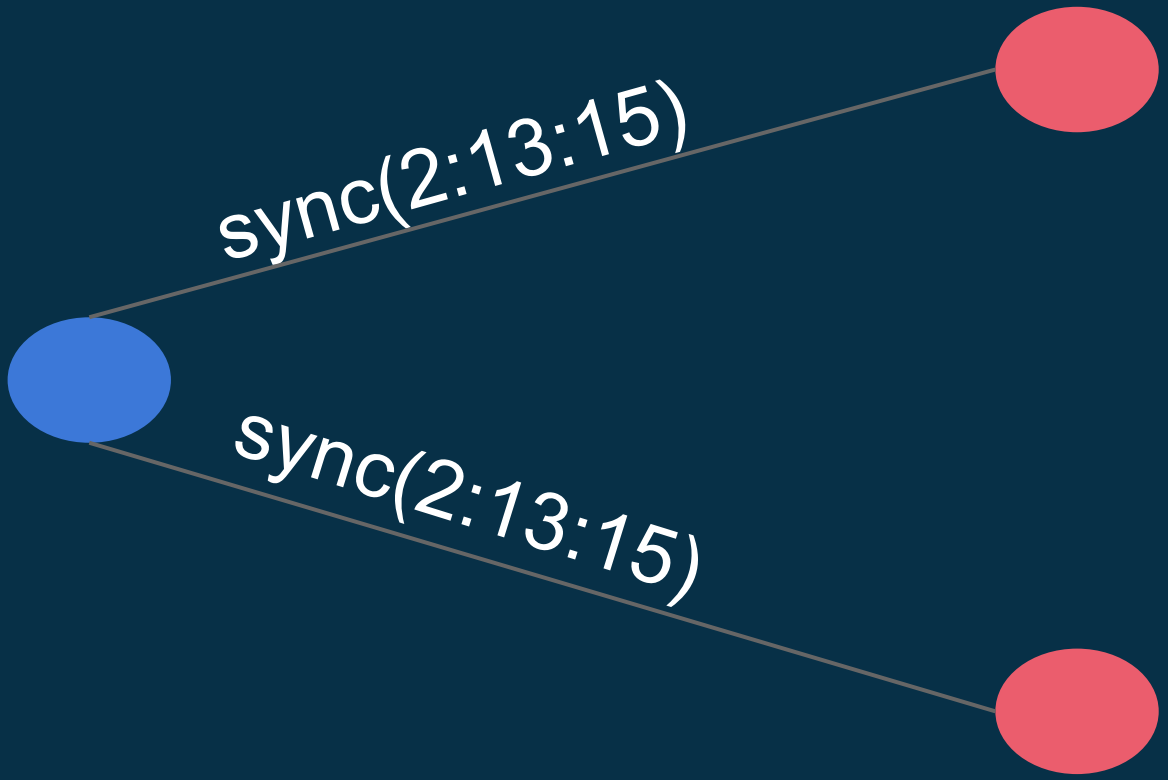
Idx	Key	Value	Expiration Time
45	sched	m3	Sept 18 2:13:30



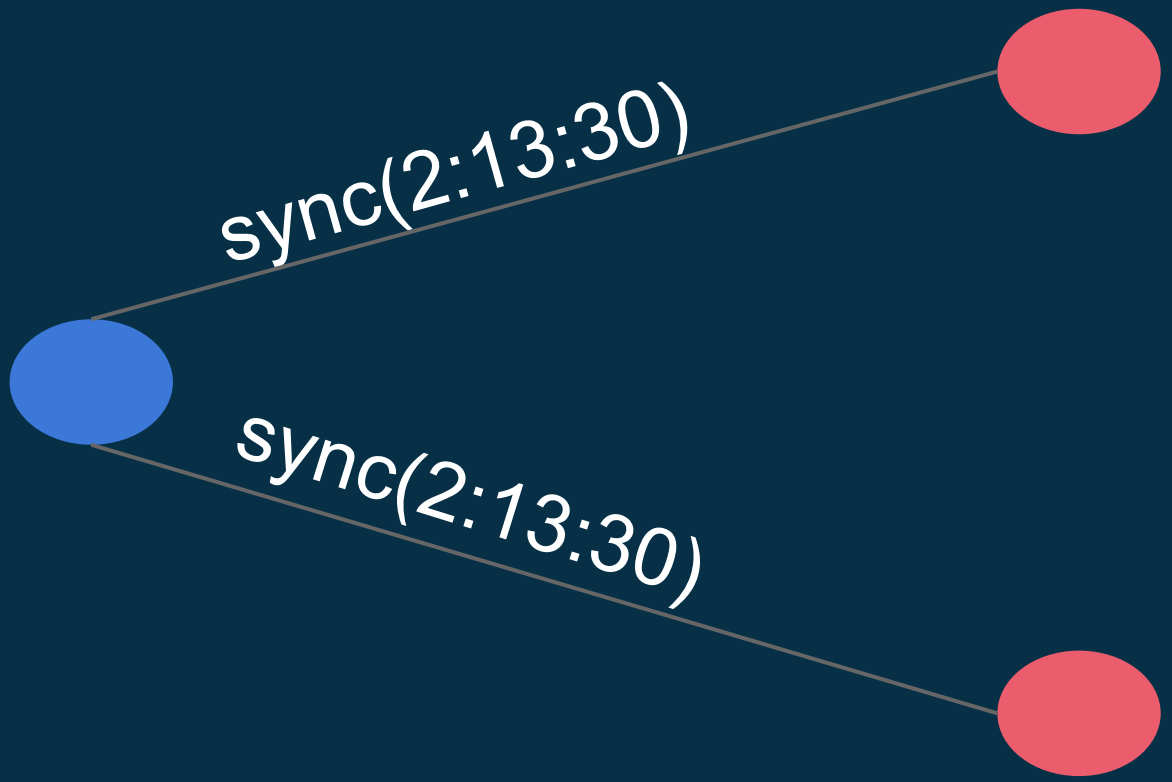
Idx	Key	Value	Expiration Time
45	sched	m3	Sept 18 2:13:30



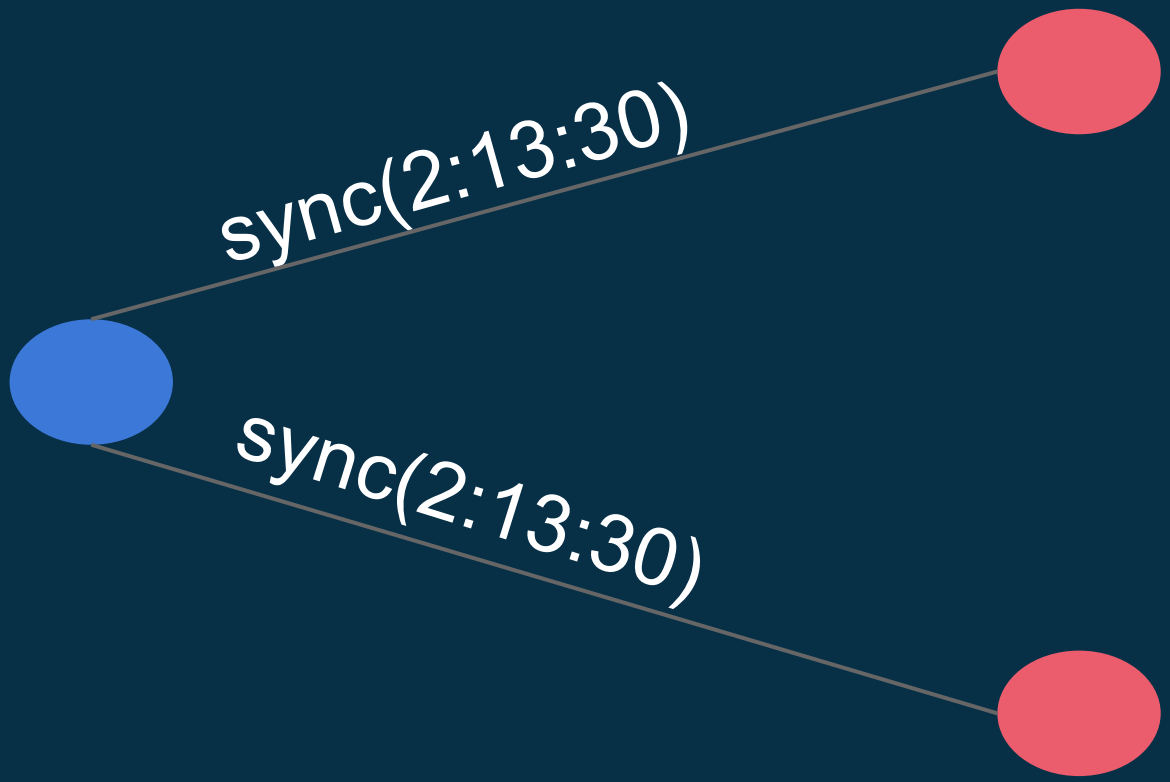
Idx	Key	Value	Expiration Time
45	sched	m3	Sept 18 2:13:30



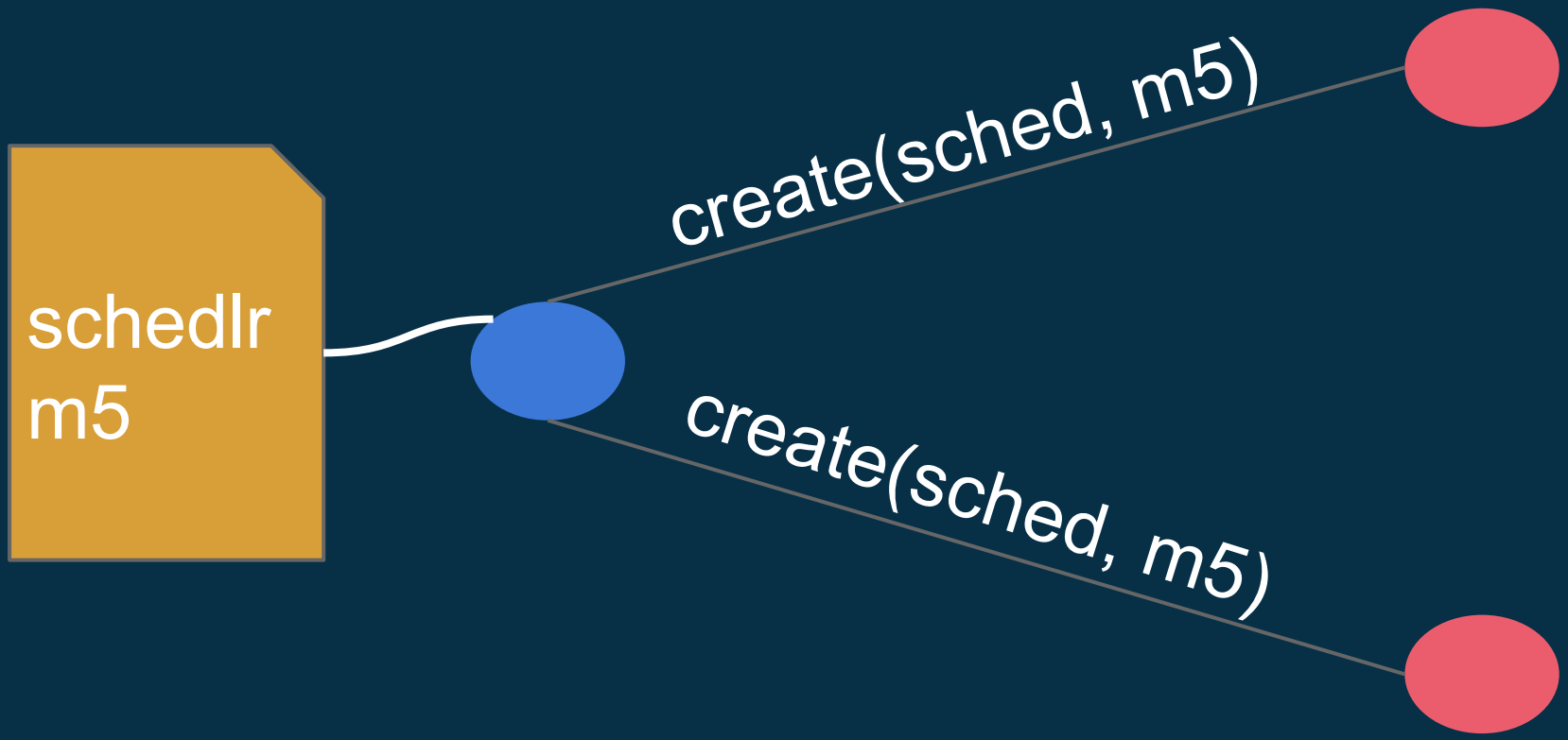
Idx	Key	Value	Expiration Time
45	sched	m3	Sept 18 2:13:30



Idx	Key	Value	Expiration Time



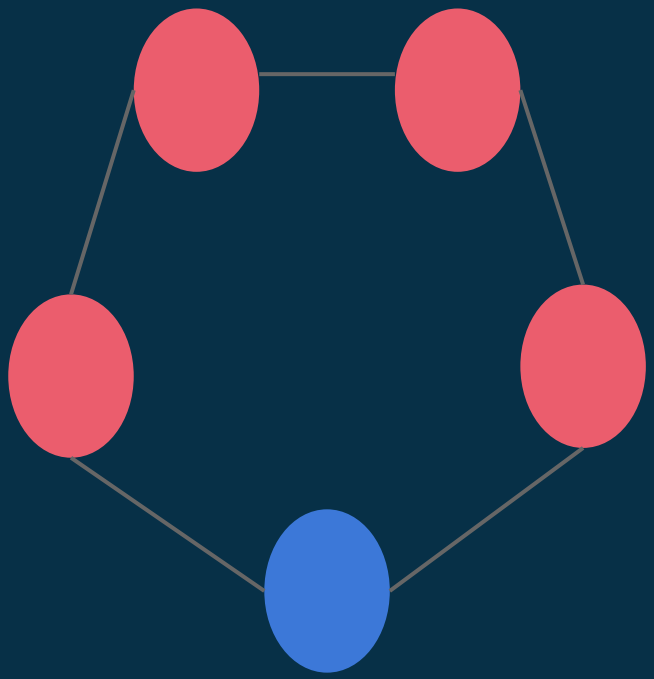
Idx	Key	Value	Expiration Time
50	sched	m5	Sept 18 2:13:35



etcd basics

clusters and bootstrapping

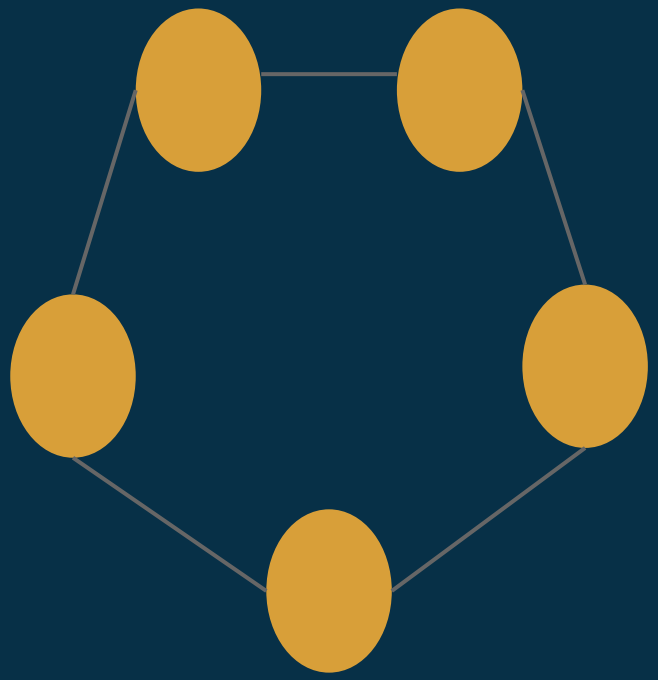
etcd Cluster



Leader

Follower

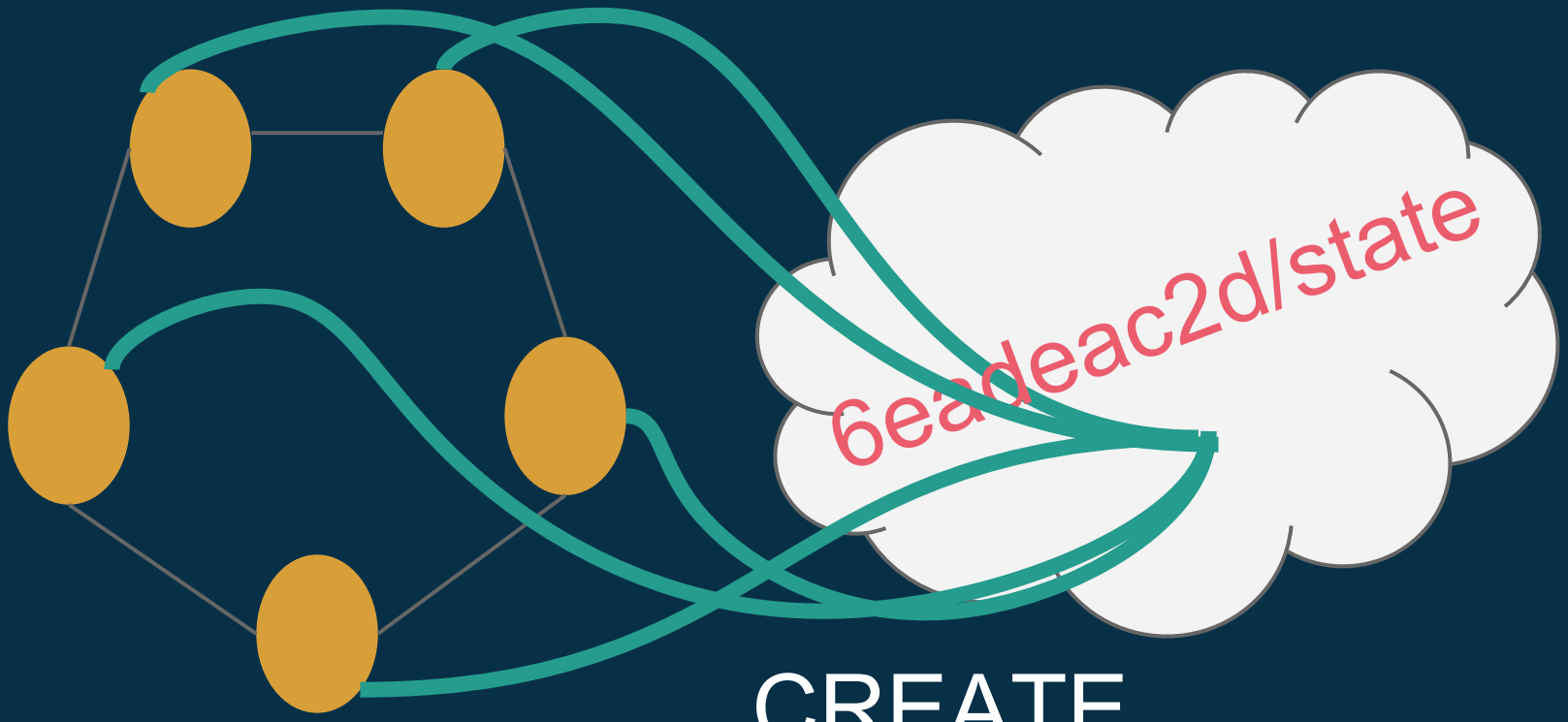
bootstrapping



Candidate

GET discovery.etcd.io/new

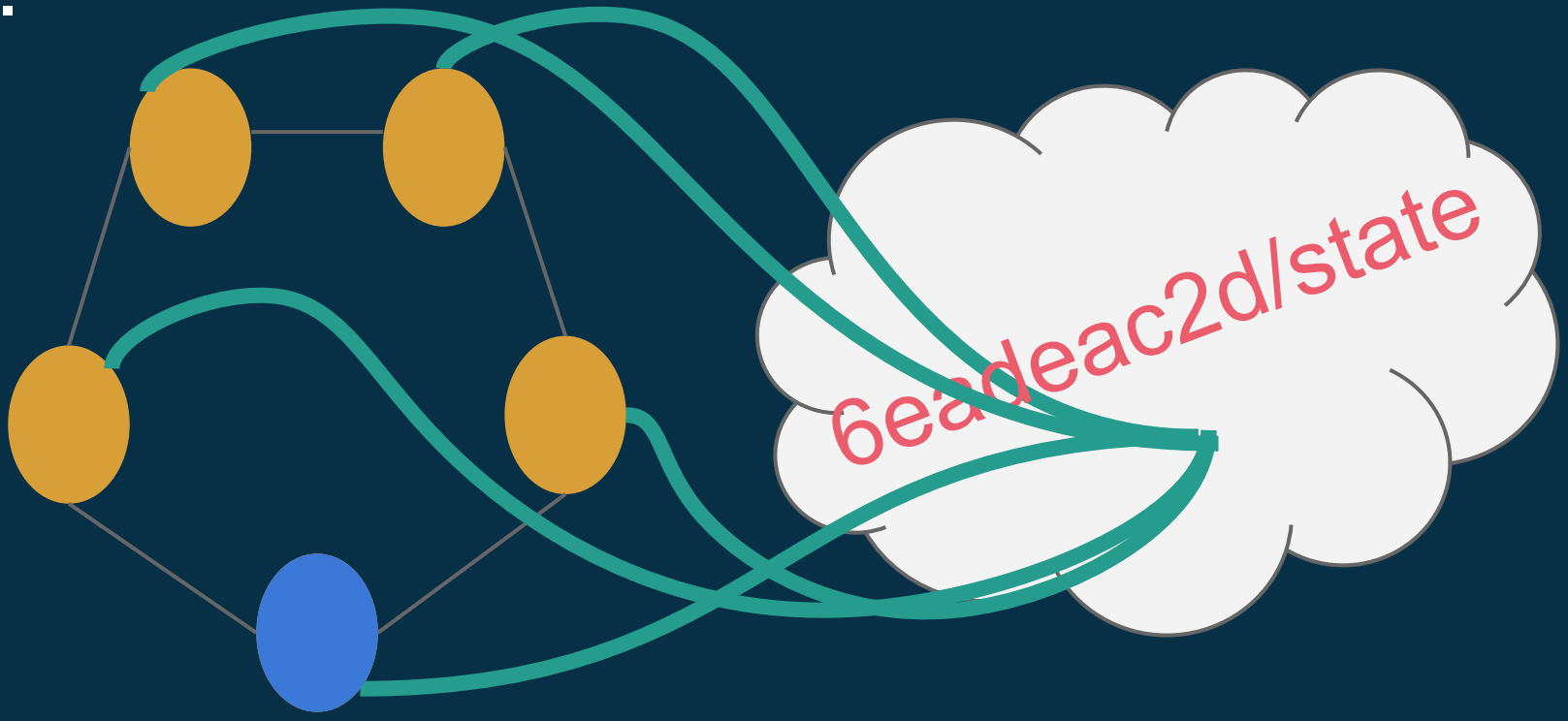
discovery.etcd.io/6eadeac2



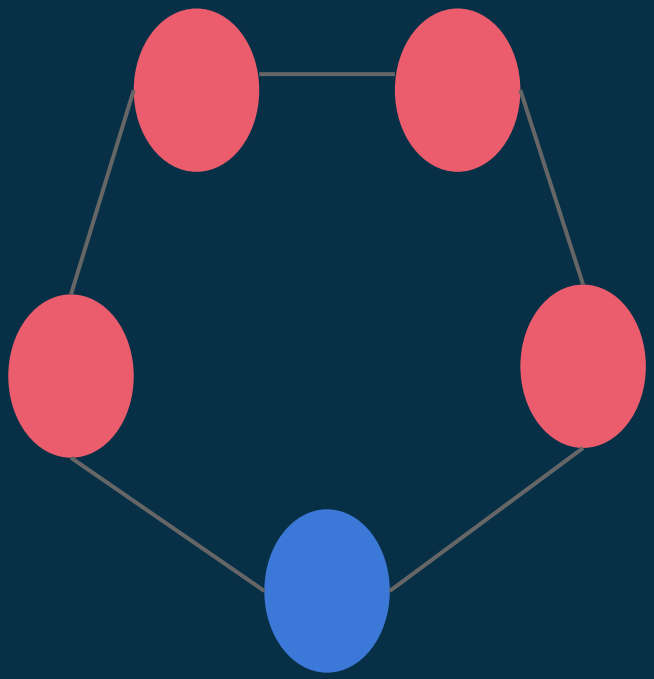
CREATE

Key	Value	Index
state	started	5890
n0	10.0.2.1	5891
n1	10.0.2.4	5898

...

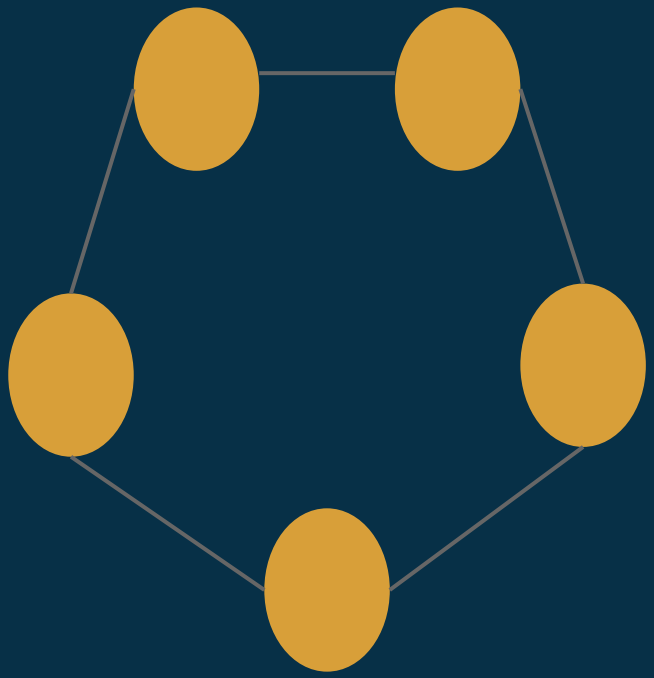


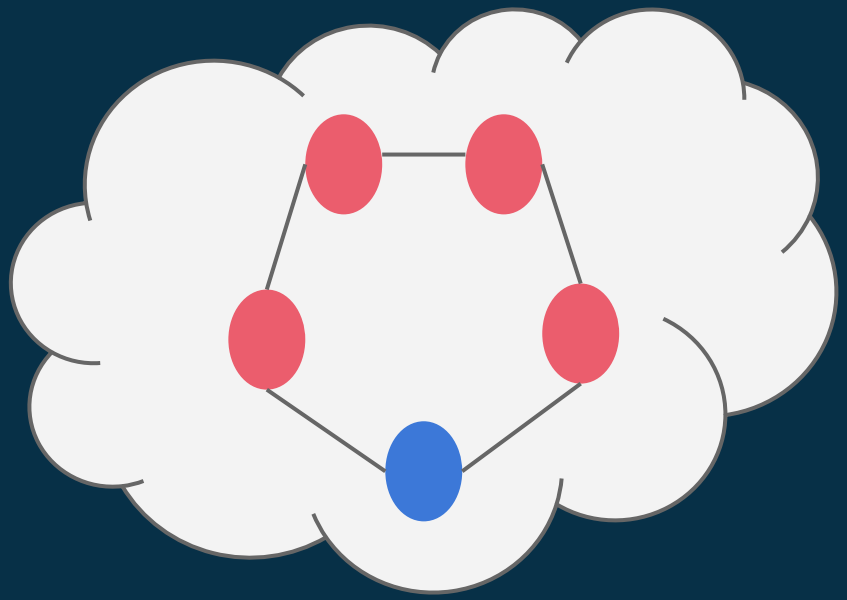
bootstrapped

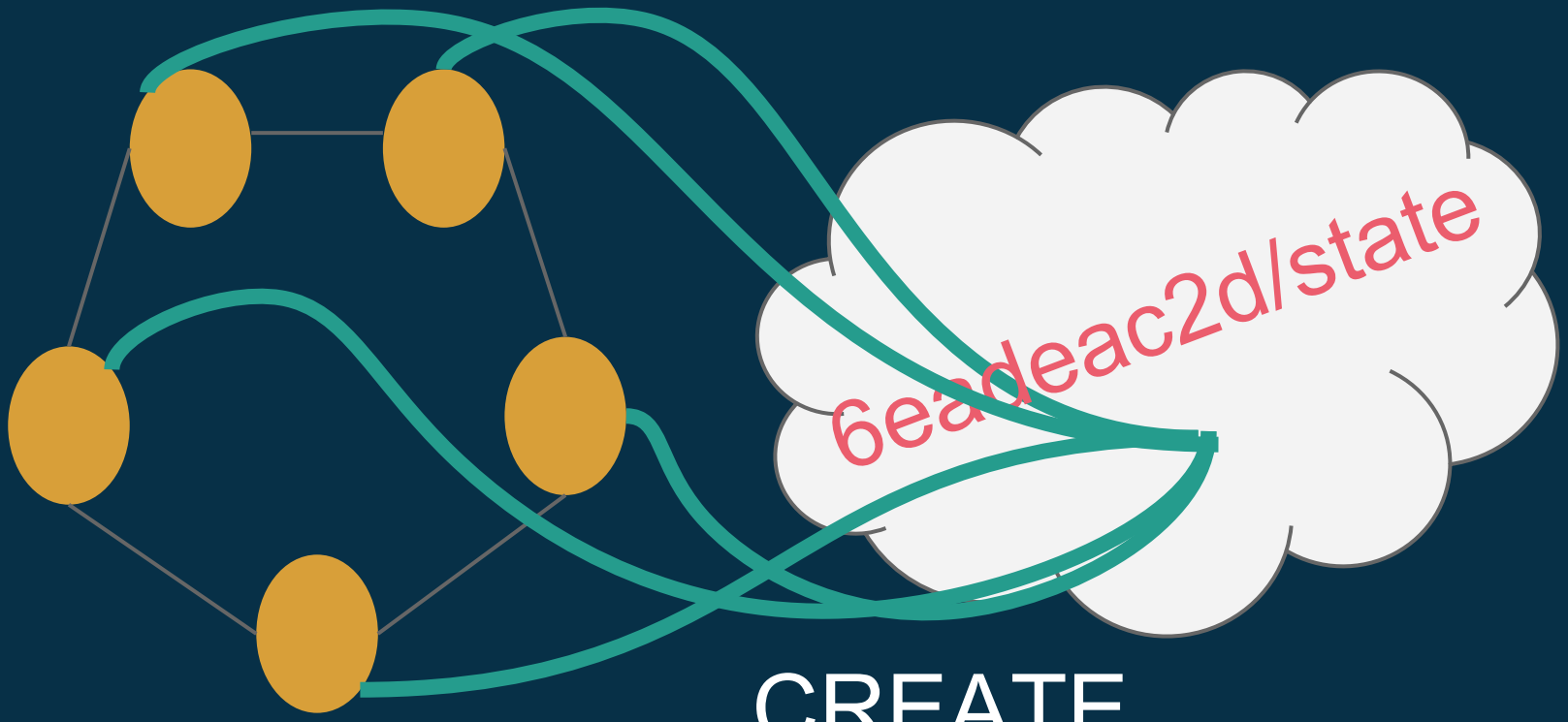


Leader

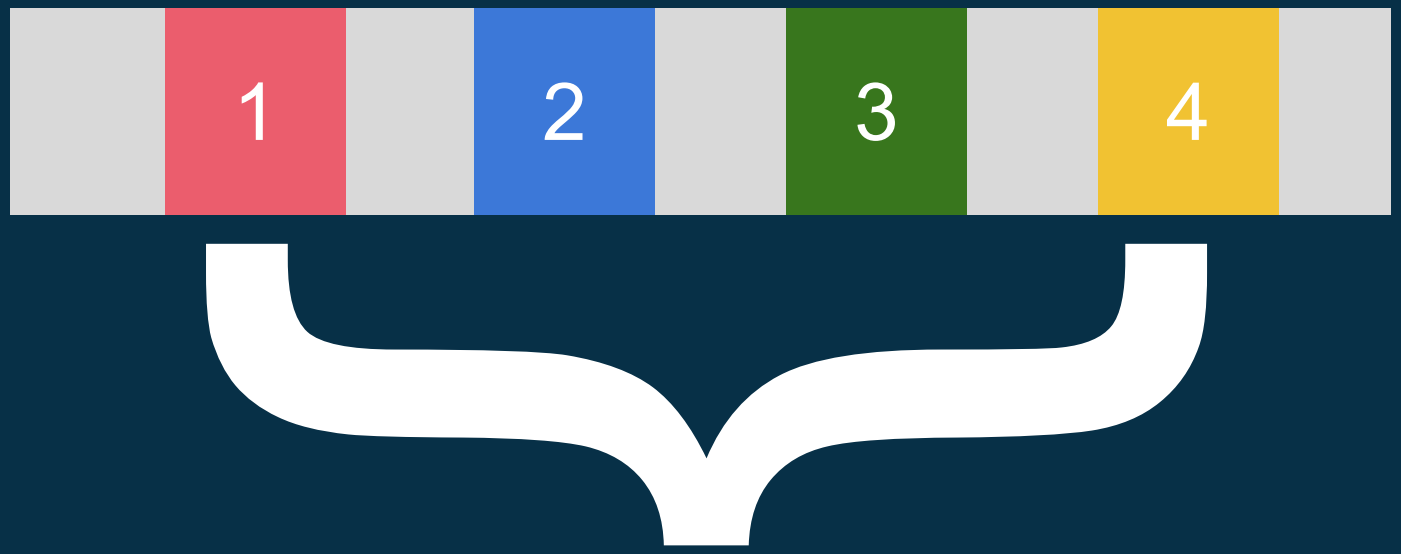
Follower



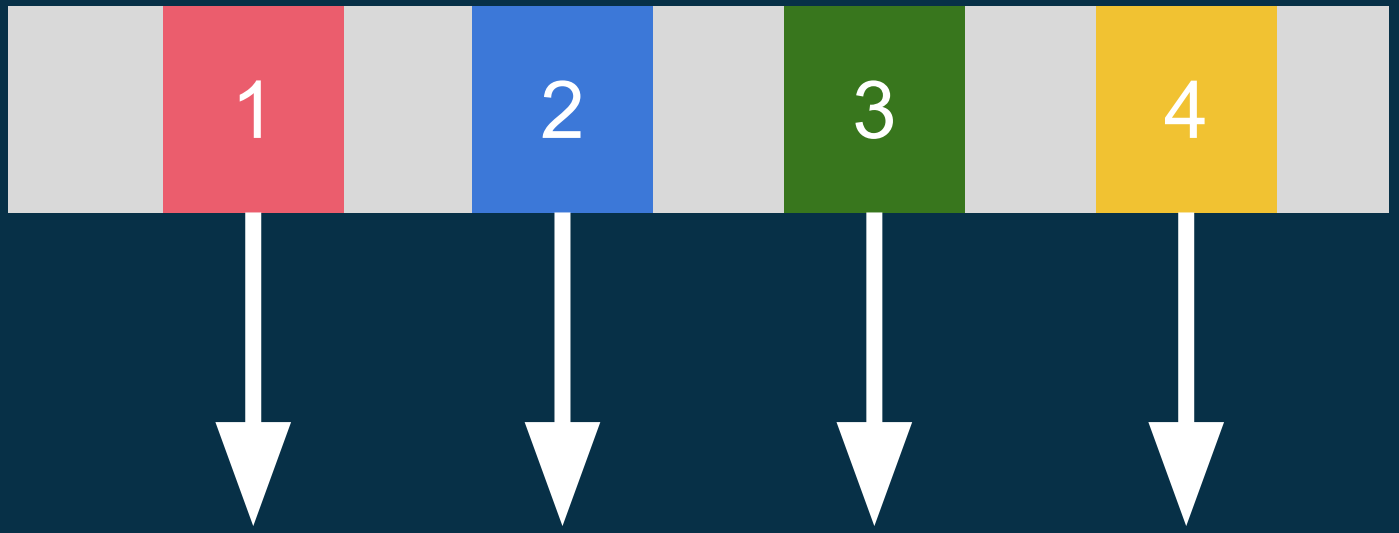




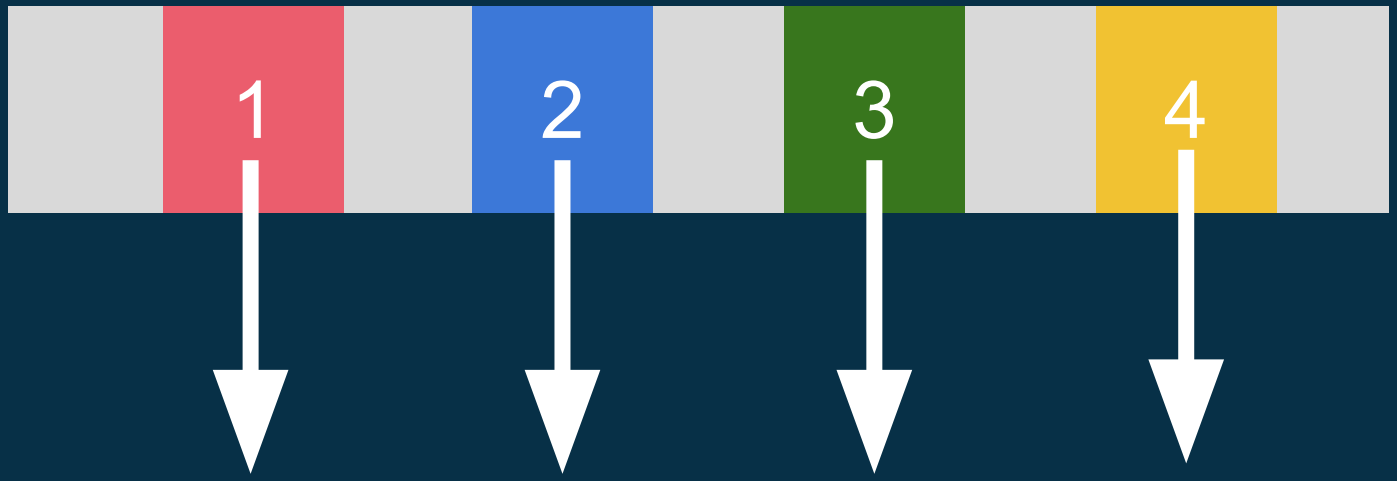
CREATE



Log



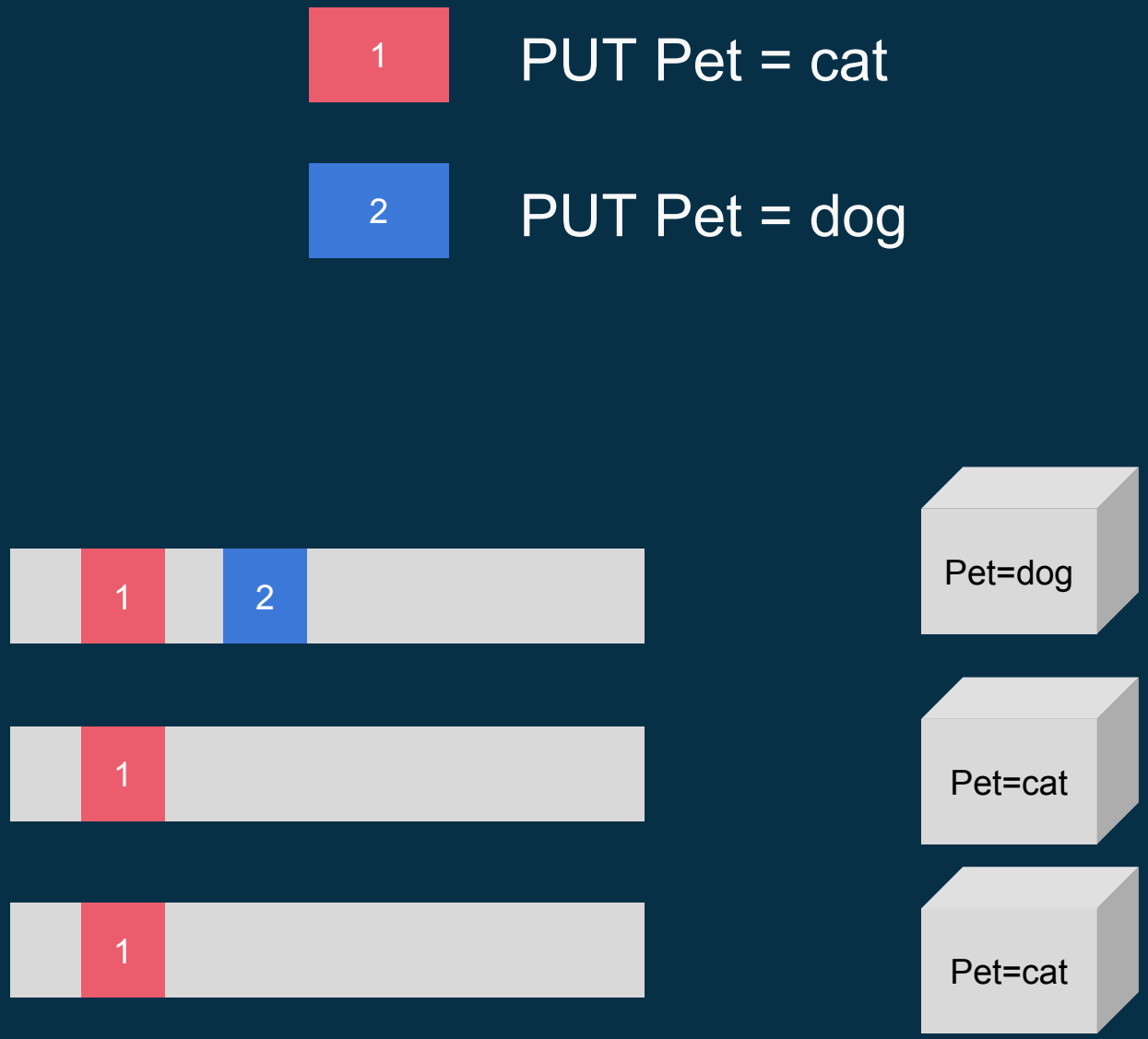
Entries

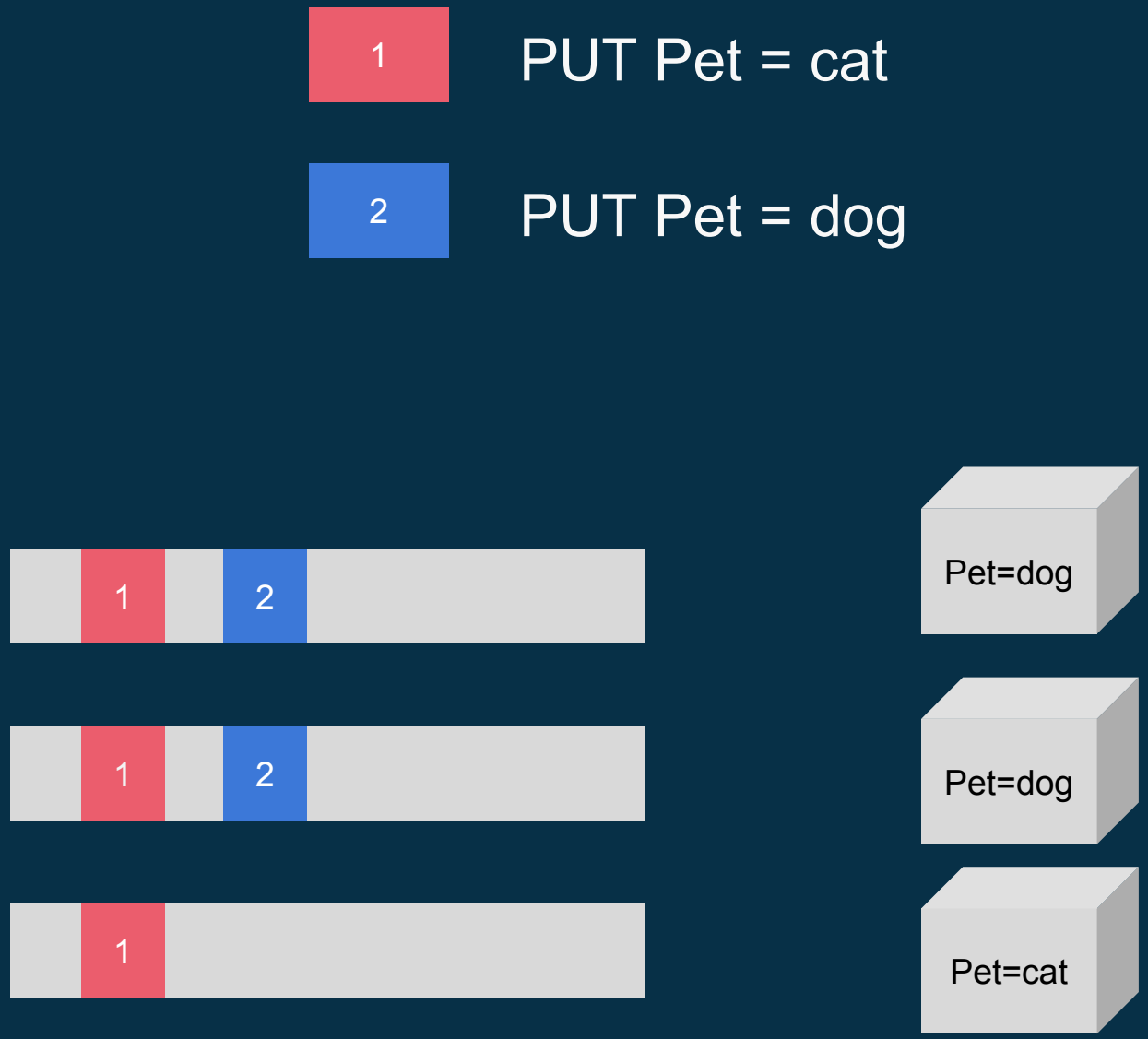


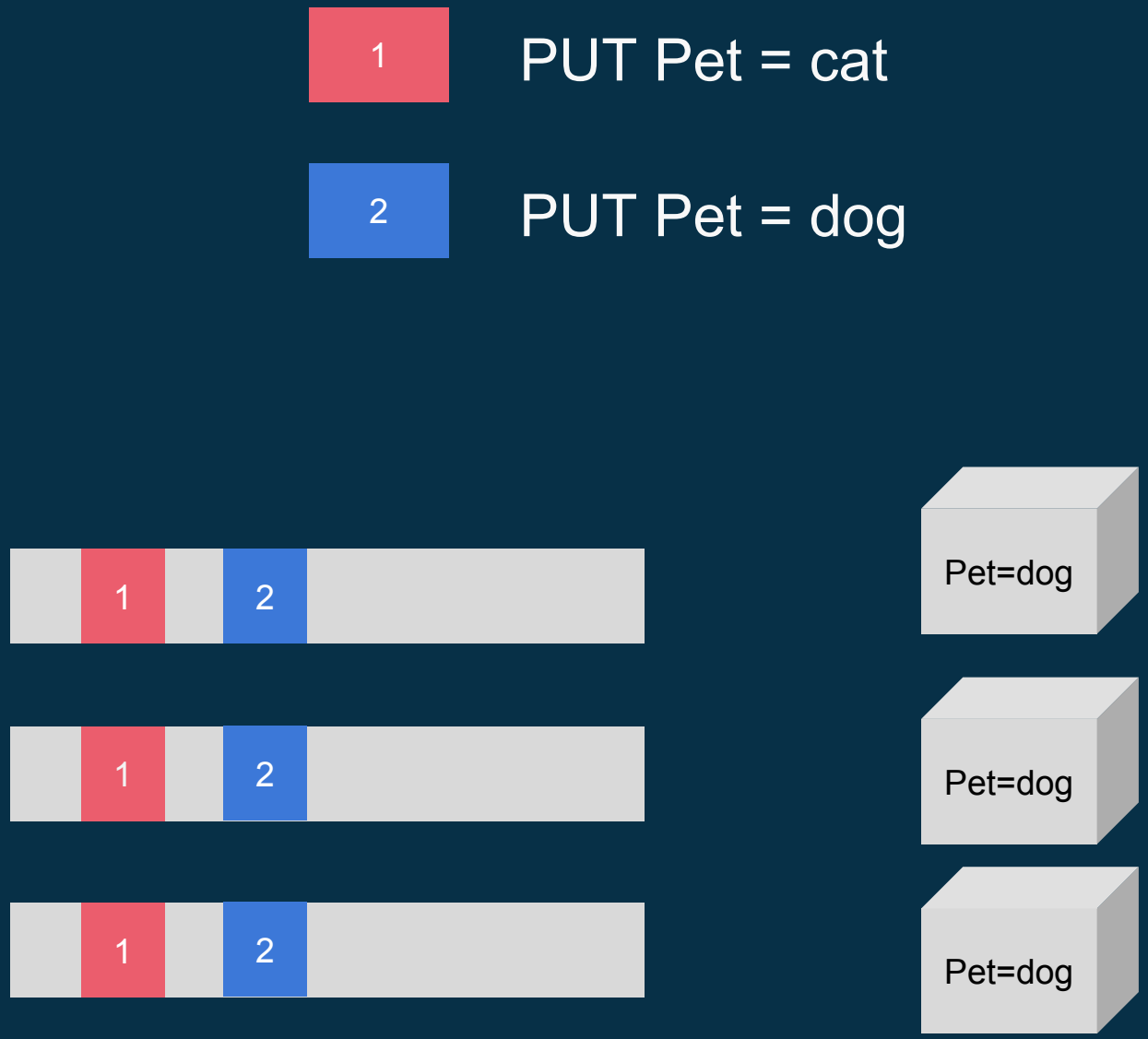
Indexes

Sequential Consistency

Operations* are atomically executed in the same sequential order on all machines.


1
PUT Pet = cat
2
PUT Pet = dog
1
2
1
1
Pet=dog
Pet=cat
Pet=cat

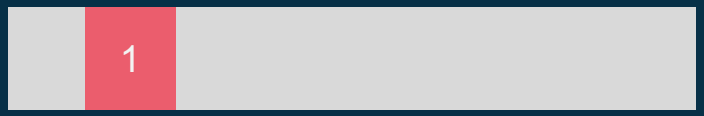

1
PUT Pet = cat
2
PUT Pet = dog
1
2
1
2
1
Pet=dog
Pet=dog
Pet=cat


1
PUT Pet = cat
2
PUT Pet = dog
1
2
1
2
1
2
Pet=dog
Pet=dog
Pet=dog

Sequential Consistency

Real-time

GET Pet @ 10:00.0 -> 2[dog]



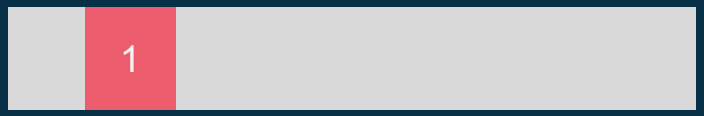
GET Pet @ 10:00.0 -> 1[cat]!?

GET Pet @ 10:00.1 -> 1[dog]

Sequential Consistency

Index Time

GET Pet @ 2 -> 2[dog]



GET Pet @ 2 -> ***blocking***

GET Pet @ 2 -> 2[dog]

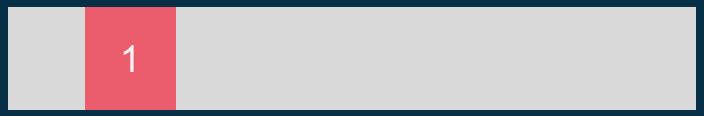
etcd guarantees that a get at index X will always return the same result.

Avoid thinking in terms of real time because with network latency the result is always out-of-date.

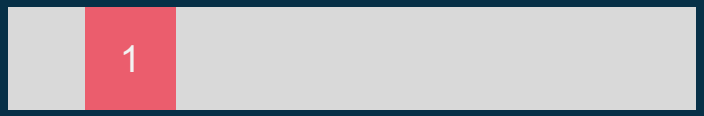
Quorum GETs

GET via Raft

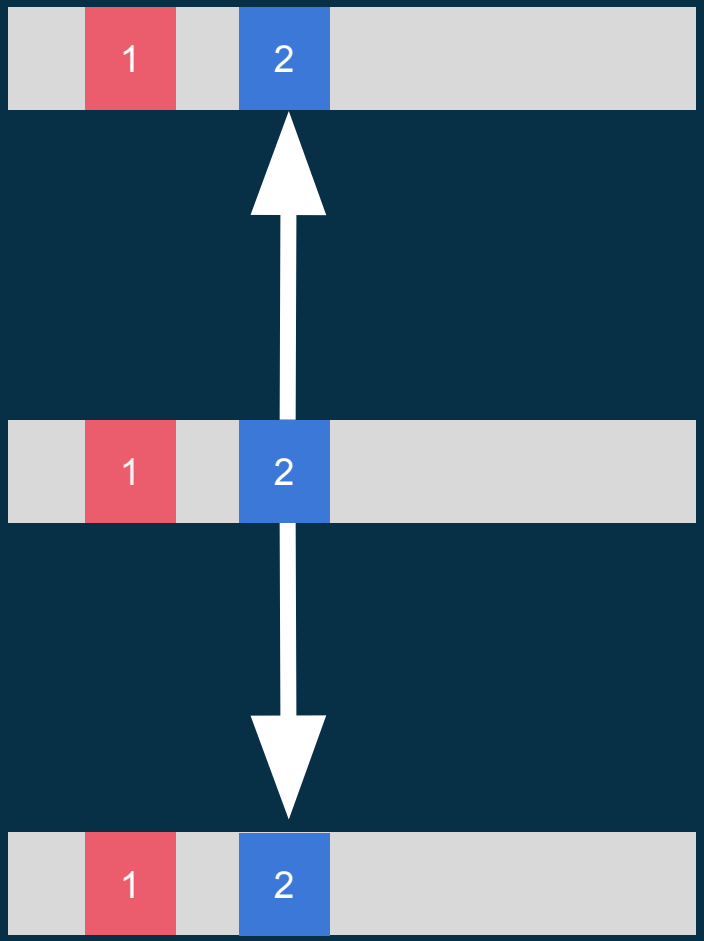

1
2


1

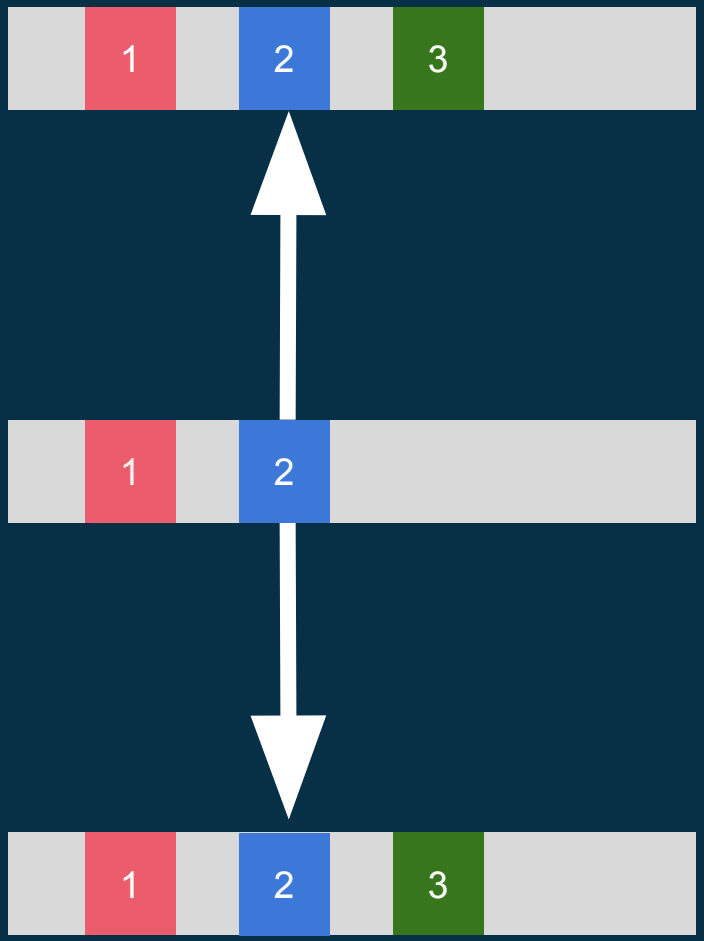

1
2



QGET A



QGET A -> 2[dog]



QGET A -> 2[dog]

Watchable Changes

HTTP Long-poll


```
> GET asdf?waitIndex=4&wait=true HTTP/1.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< X-Etcd-Index: 3
< X-Raft-Index: 97
< X-Raft-Term: 0
<
BLOCK
```


```
> GET asdf?waitIndex=4&wait=true HTTP/1.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< X-Etcd-Index: 3
< X-Raft-Index: 97
< X-Raft-Term: 0
<
{"action":"set","node":{"key":"/asdf","value":"foobar","
modifiedIndex":4,"createdIndex":4}}
```

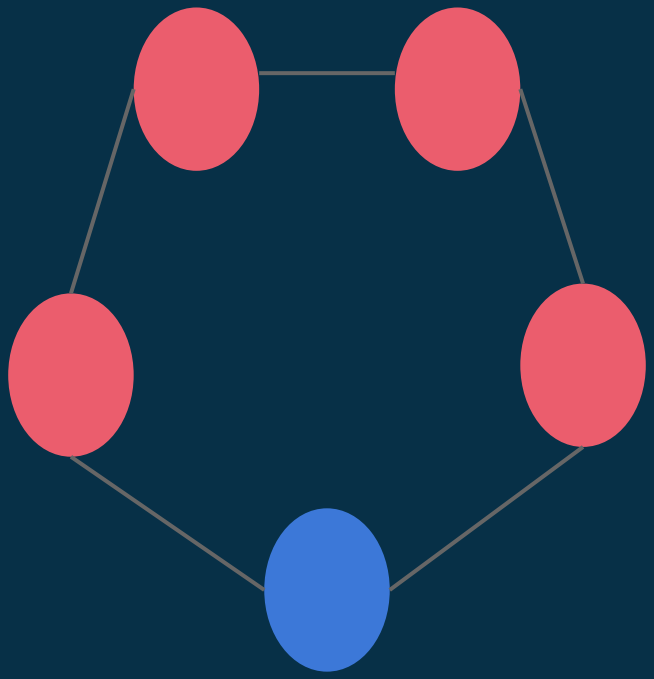

```
> GET asdf?waitIndex=4&wait=true HTTP/1.1
> Accept: */*
>
< HTTP/1.1 200 OK
< Content-Type: application/json
< X-Etcd-Index: 4
< X-Raft-Index: 516
< X-Raft-Term: 0
<
{"action":"set","node":{"key":"/asdf","value":"foobar","
modifiedIndex":4,"createdIndex":4}}
```

Event History

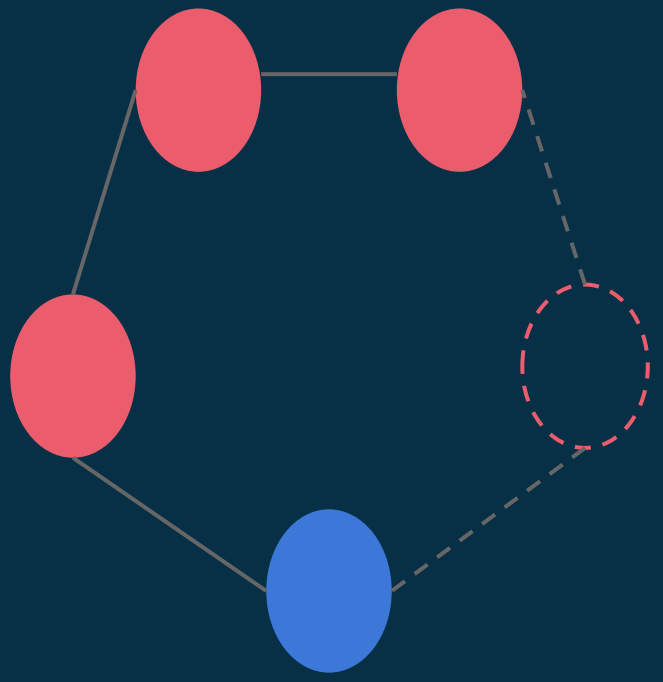
Availability

In a 2F+1 cluster tolerate F machine failures

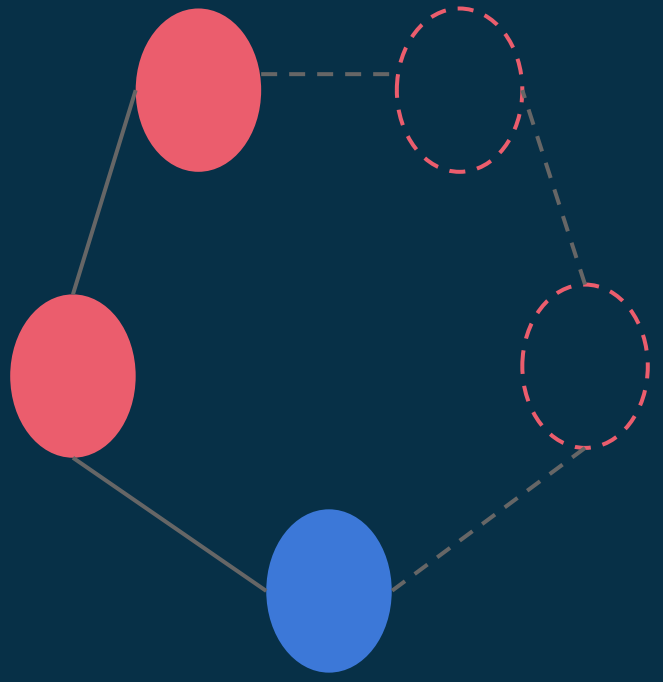
Available



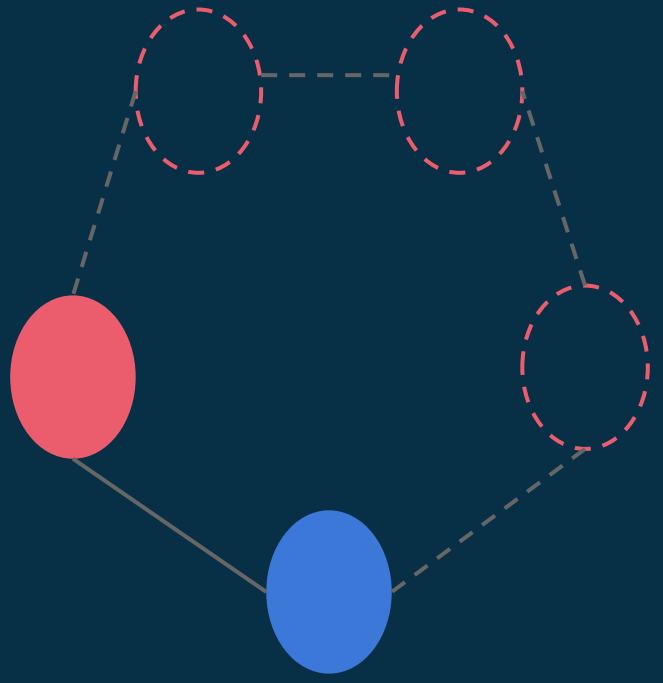
Available



Available



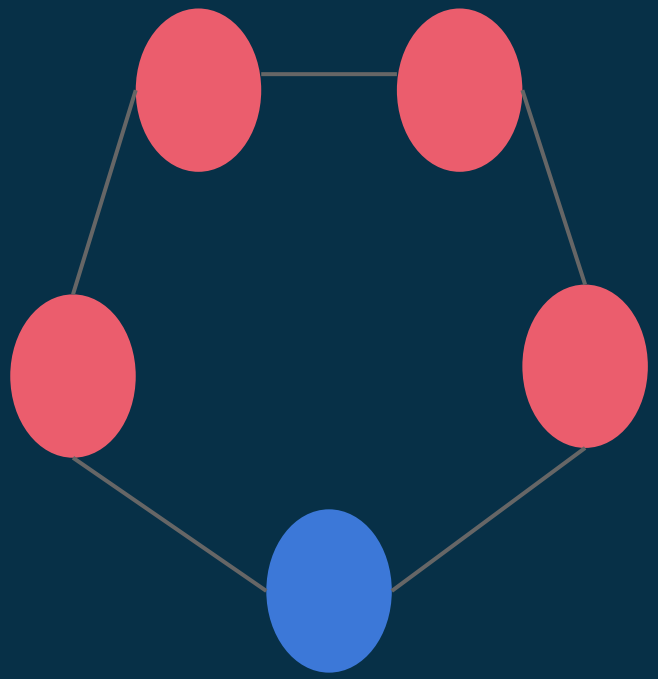
Unavailable



Master Election

Fast recovery (5-10*typical RTT) from temporarily unavailable

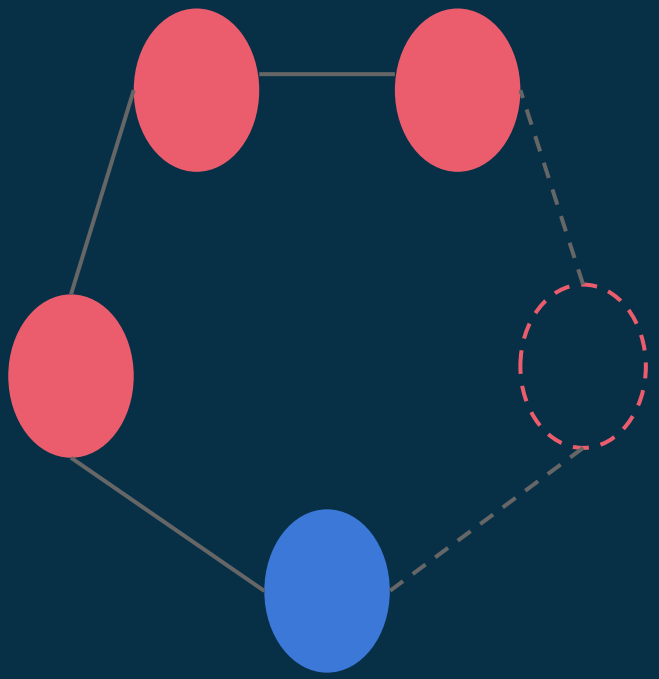
Available



Leader

Follower

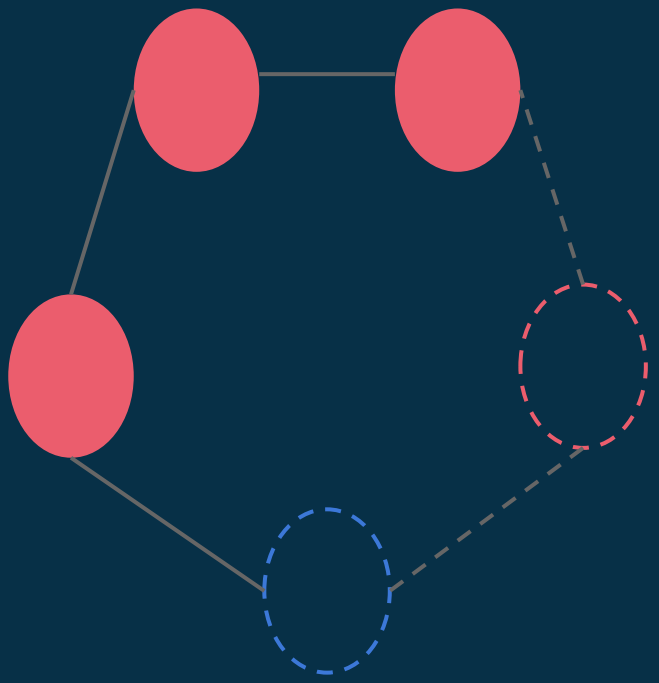
Available



Leader

Follower

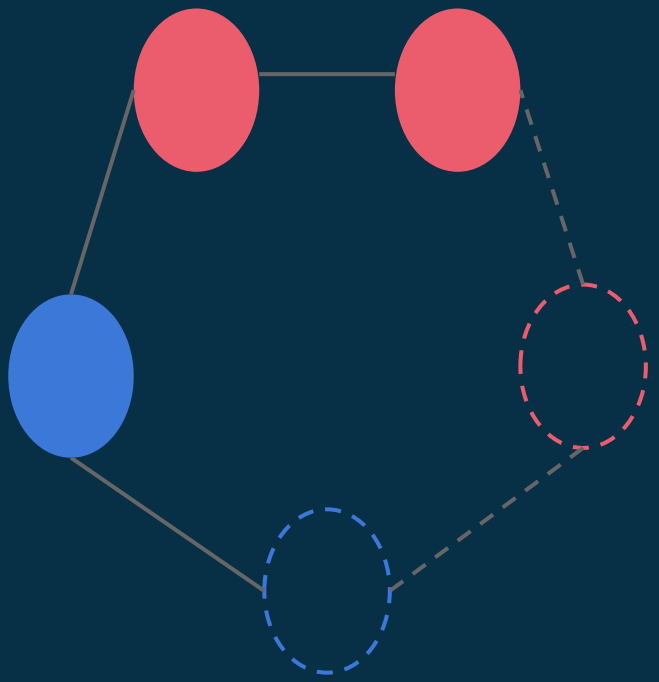
Temporarily Unavailable



Leader

Follower

Available



Leader

Follower

Durable

log files, snapshots and backups

Mistakes so far...

Log files

Filesystems truncate and corrupt data.

Solutions:

- Must use checksumming in the file to ensure sanity
- Throwing out broken log files must be handled by the server

etcd machine naming

Trusted users to manage unique names across the cluster. This went poorly.

- Misconfiguration from bugs
- Misconfiguration by users
- Machine cloning on the cloud

Solution: etcd data-dir owns a unique uuid.

sync() in the cloud

Slow, slow, slow:

- User #1 OpenStack on spinning disk: 6s
- User #2 AWS EBS backed: 1.5s

Solution:

- Tune etcd to expect this long latency.
- Write batching and handling of behind machines.

Wednesday 10:40am LCA
CoreOS: An Introduction

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

Thursday 6:00 PM Go AKL Meetup
Something about Go

Friday 10:40am LCA
CoreOS Tutorial

Thanks

we like pull requests
github.com/coreos/etcd

CoreOS