

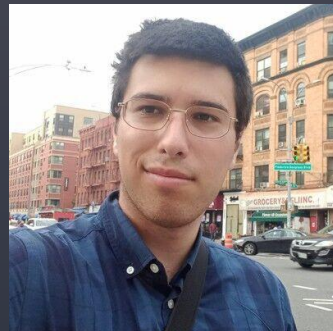
Making Sense of Cephadm and Rook

Two Ways To Manage Ceph

David Mohren, Software Engineer
October, 2026

Who am I?

- David Mohren
 - Works on Rook, Ceph-csi, Ceph
 - Part of the Ceph Security Working Group
 - Focus security related topics
-
- I have a clear bias towards Rook :D
 - Often try to evaluate Rook changes based on Cephadm



Same problem, two communities

cephadm

Custom Ceph orchestration tool

Rook

Ceph in Kubernetes

This talk: how each one thinks, so you can judge both.

Why this talk

Reviewing Rook's update for CVE-2025-30156,
I kept checking how cephadm does it.

What you take home

- 01. how each tool works, and the thinking behind it

What you take home

- 01. how each tool works, and the thinking behind it
- 02. how the differences play out in day-2 operations

What you take home

- 01. how each tool works, and the thinking behind it
- 02. how the differences play out in day-2 operations
- 03. a basis for choosing

01.

Basics for each tool

02.

Day-2 operations

object store · CVE Update procedure · orchestrator is down

Content

OSD replacement: see Artem Torubarov's talk

01.

Basics for each tool

Who makes them

cephadm

Ceph developers, in the Ceph repo
released with Ceph, once a year
since Octopus, 2019

Rook

Kubernetes people, a CNCF project
its own release, every 4 months
for two Ceph releases at a time

Each tool thinks like the people who build it.

— The reconcile loop

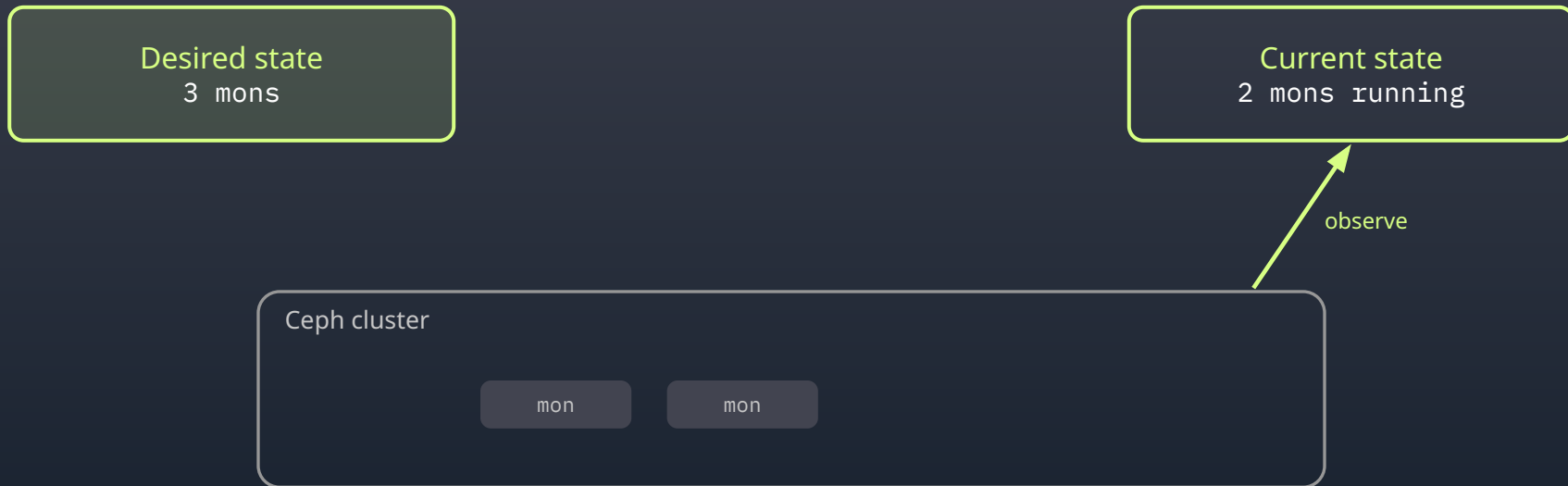
compare the desired state with the current state, act until they match

Desired state

3 mons

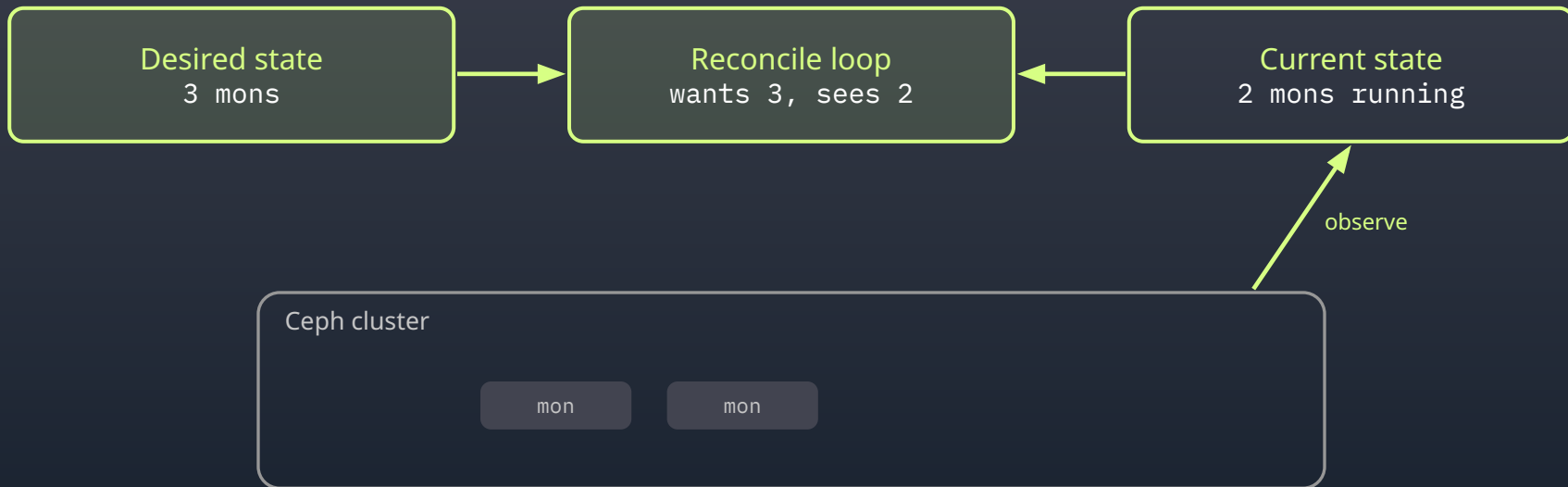
The reconcile loop

compare the desired state with the current state, act until they match



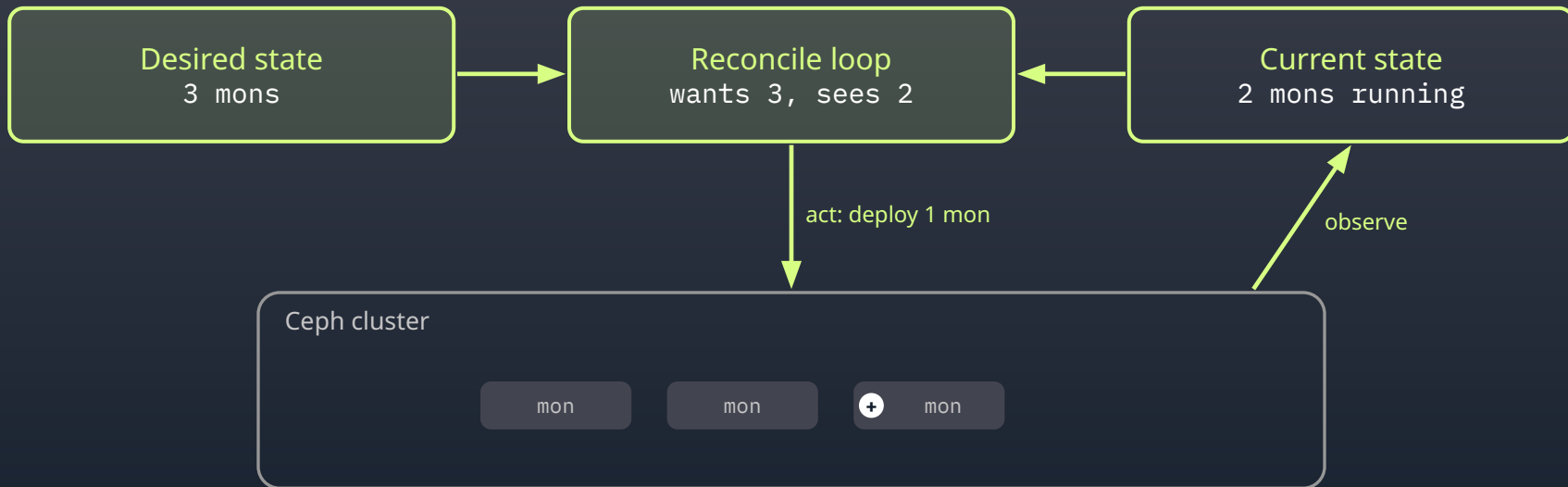
The reconcile loop

compare the desired state with the current state, act until they match



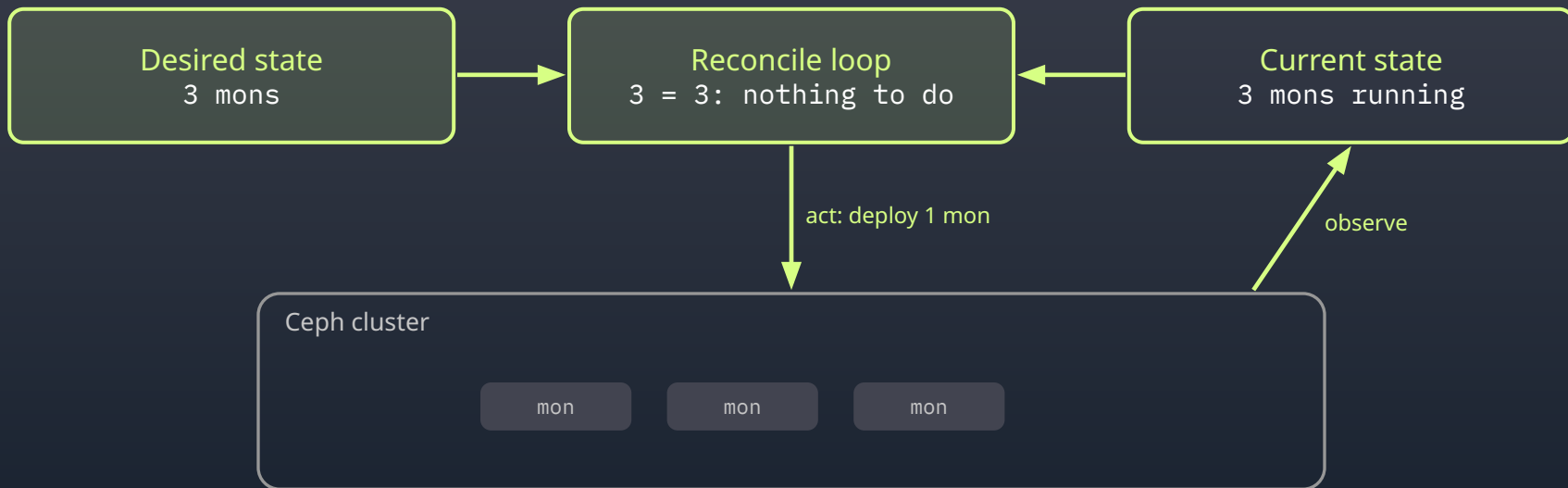
The reconcile loop

compare the desired state with the current state, act until they match



The reconcile loop

compare the desired state with the current state, act until they match



... and the loop keeps running

Two homes for the loop

cephadm

ceph-mgr

cephadm module: reconcile loop

a module in the ceph-mgr: Ceph manages itself

Two homes for the loop

cephadm

ceph-mgr

cephadm module: reconcile loop

a module in the ceph-mgr: Ceph manages itself

Rook

Kubernetes

Rook operator (a pod): reconcile loops

an operator on Kubernetes: Kubernetes runs the daemons

Two homes for the loop

cephadm

Rook

ceph-mgr

Kubernetes

cephadm module: reconcile loop

Rook operator (a pod): reconcile loops

a module in the ceph-mgr: Ceph manages itself

an operator on Kubernetes: Kubernetes runs the daemons

“cephadm is like Rook, but with SSH and systemd”

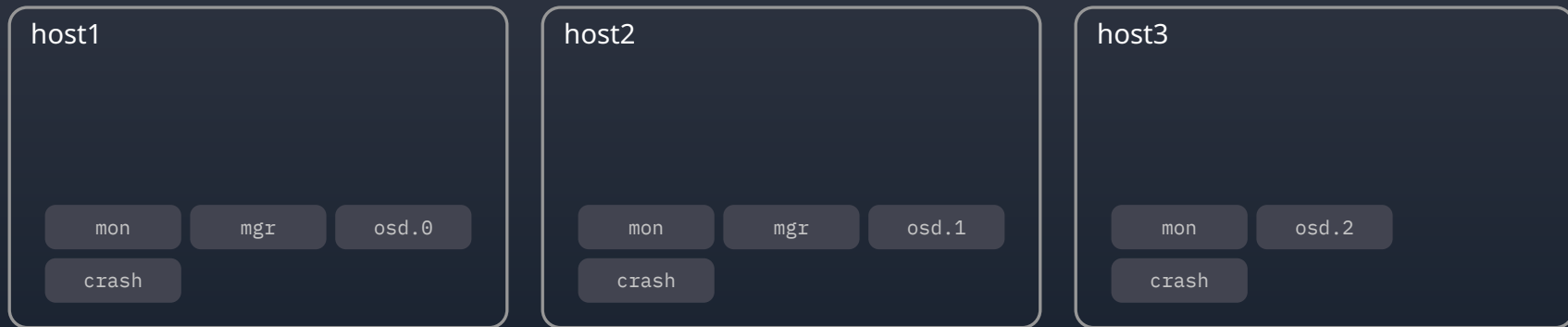
“cephadm is like Rook, but leveraging Kubernetes functionality”

01.

What is the setup behind
cephadm?

cephadm: containers

every daemon is a container, run by systemd



cephadm



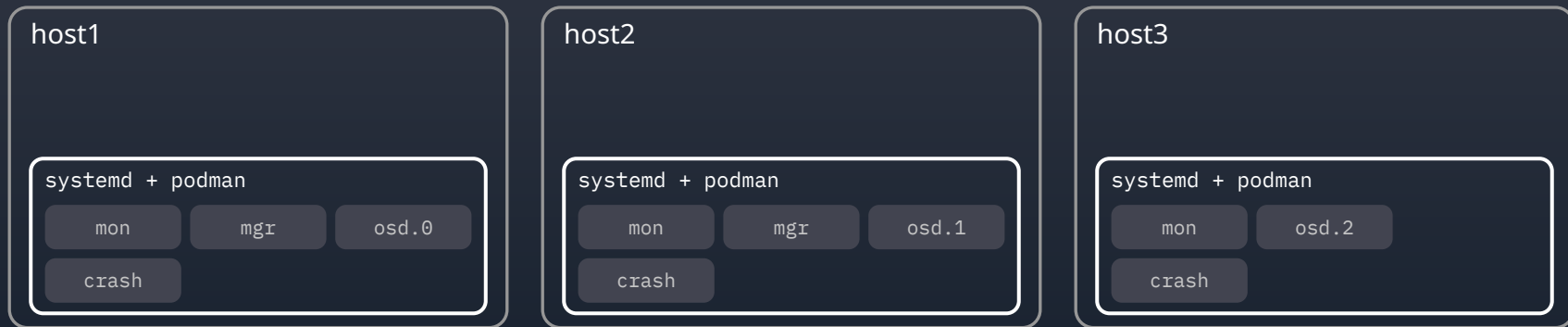
Linux



Ceph

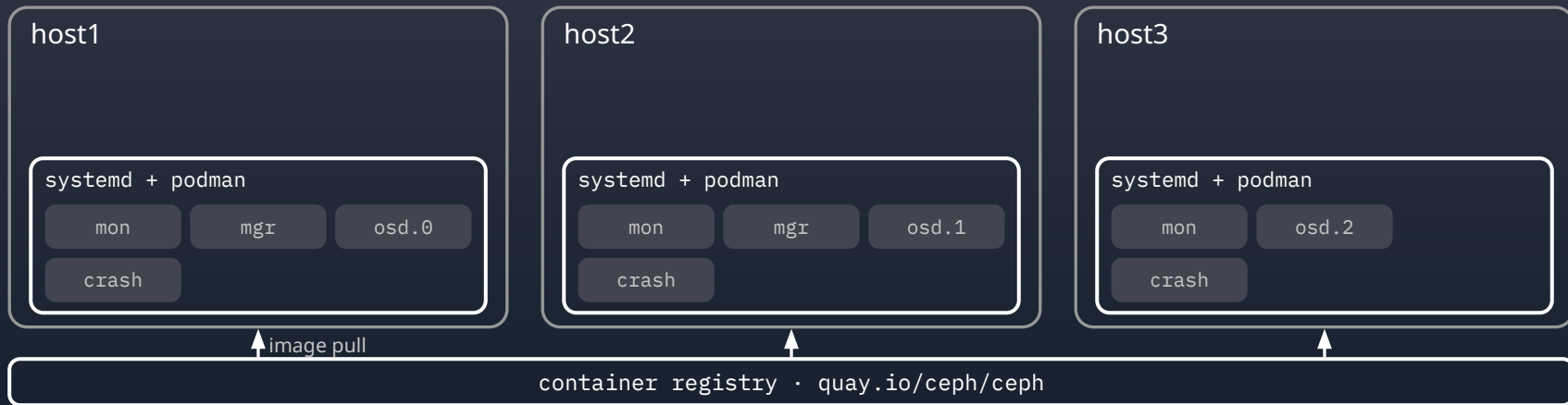
cephadm: containers

every daemon is a container, run by systemd



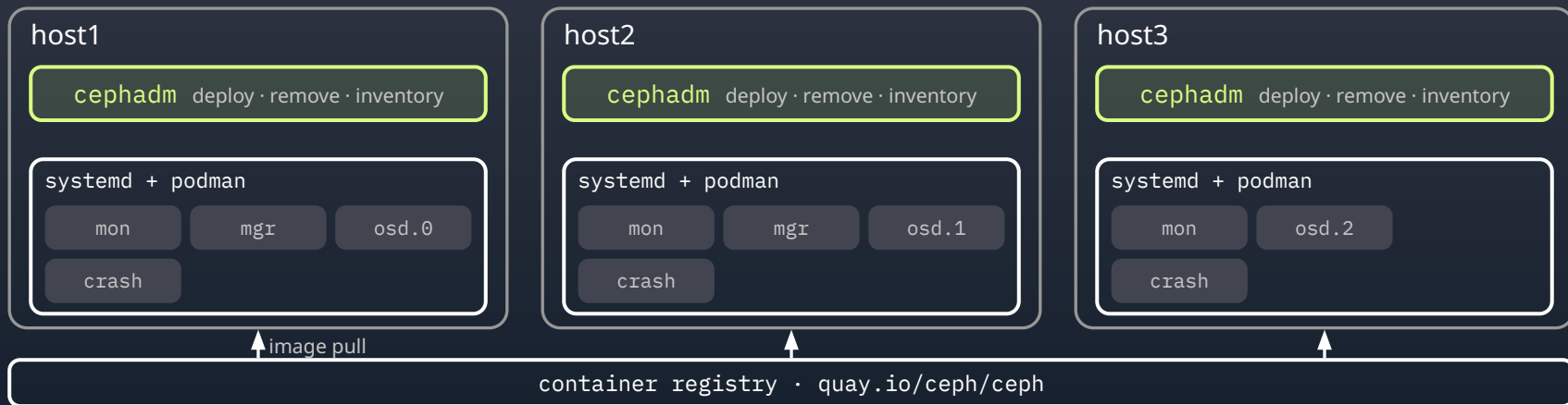
cephadm: containers

every daemon is a container, run by systemd



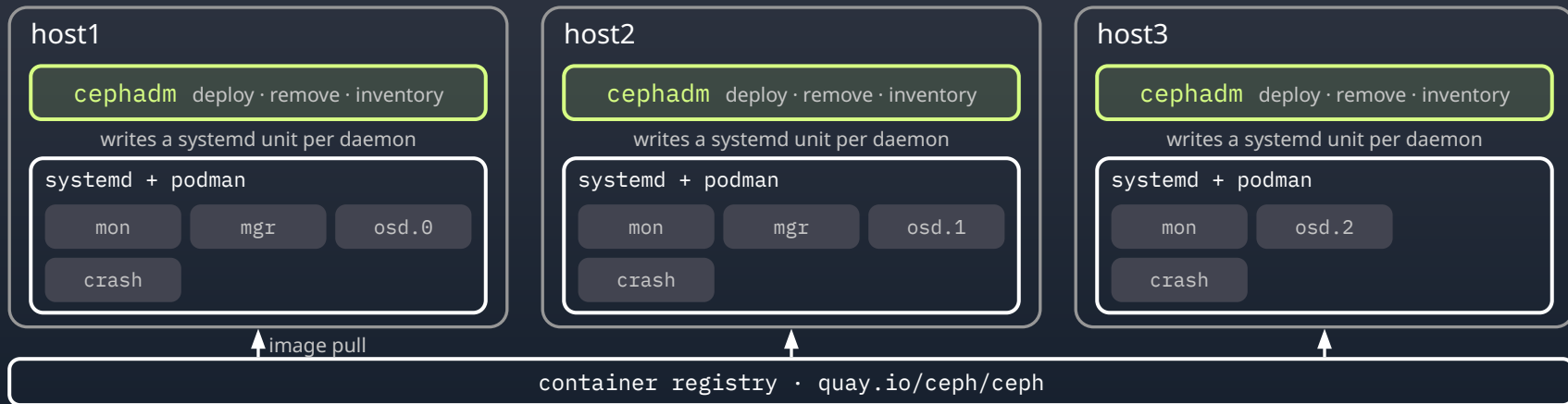
cephadm: the binary

a small binary on each host does the work



cephadm: the binary

a small binary on each host does the work

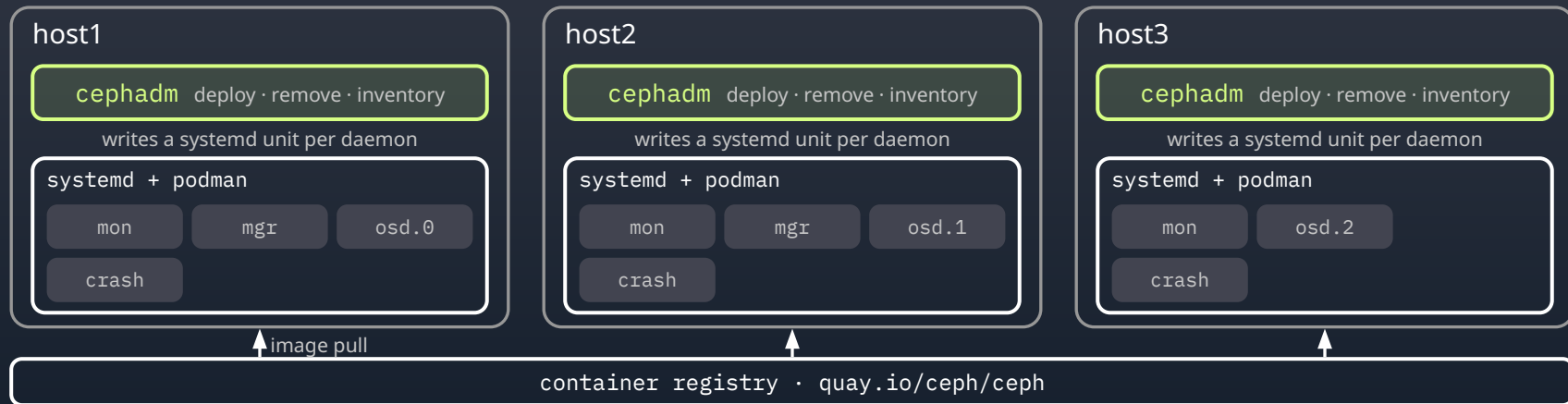


cephadm: the brain

the orchestrator is a module in the ceph-mgr

cephadm orchestrator module

in the active ceph-mgr



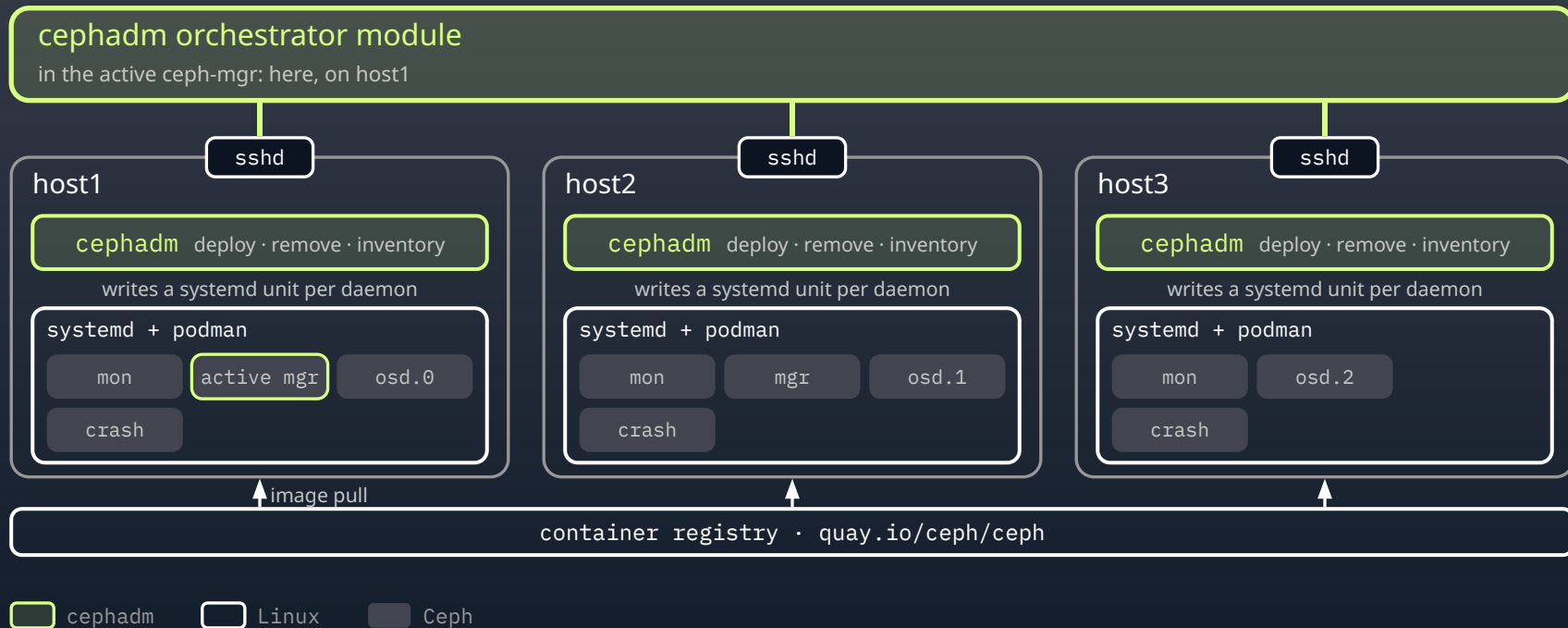
the orchestrator is a module in the ceph-mgr

the orchestrator is a module in the ceph-mgr



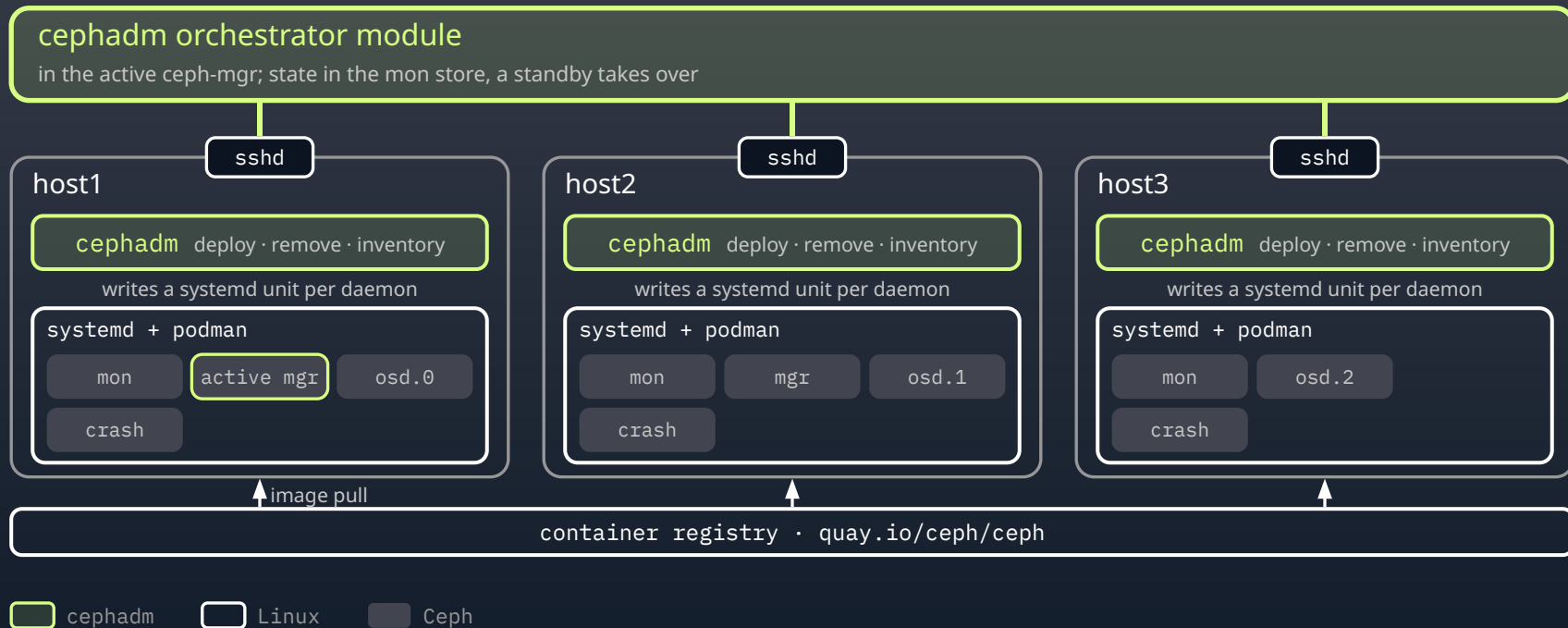
cephadm: the brain

the orchestrator is a module in the ceph-mgr



cephadm: the brain

the orchestrator is a module in the ceph-mgr



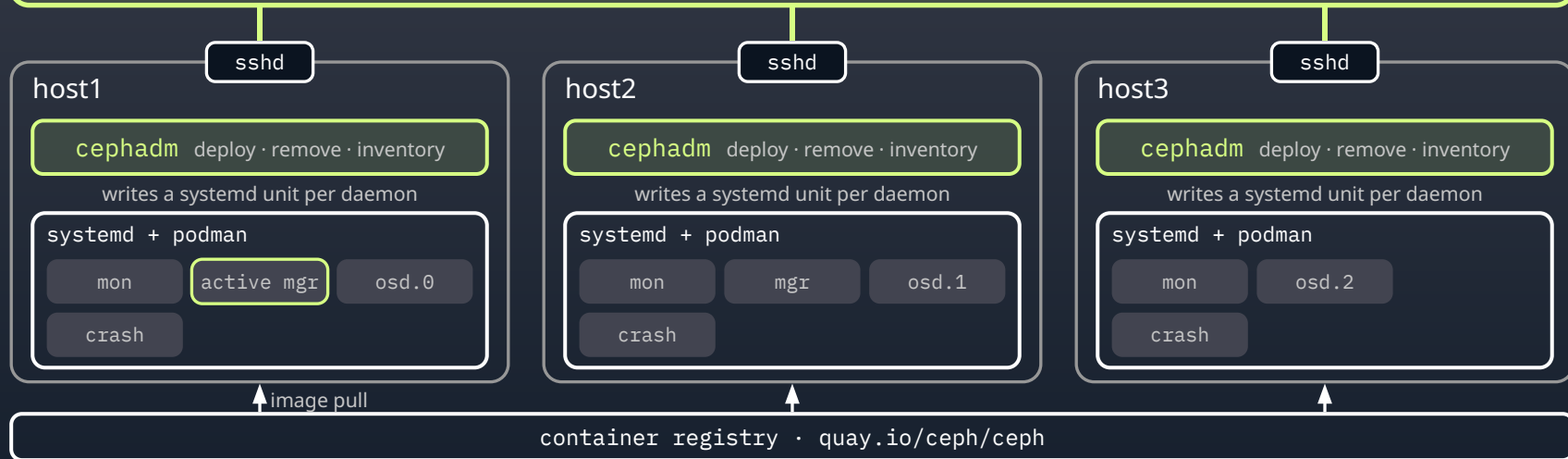
cephadm: the CLI

you talk to it with the ceph CLI

```
admin $ ceph orch apply -i spec.yaml
```

cephadm orchestrator module

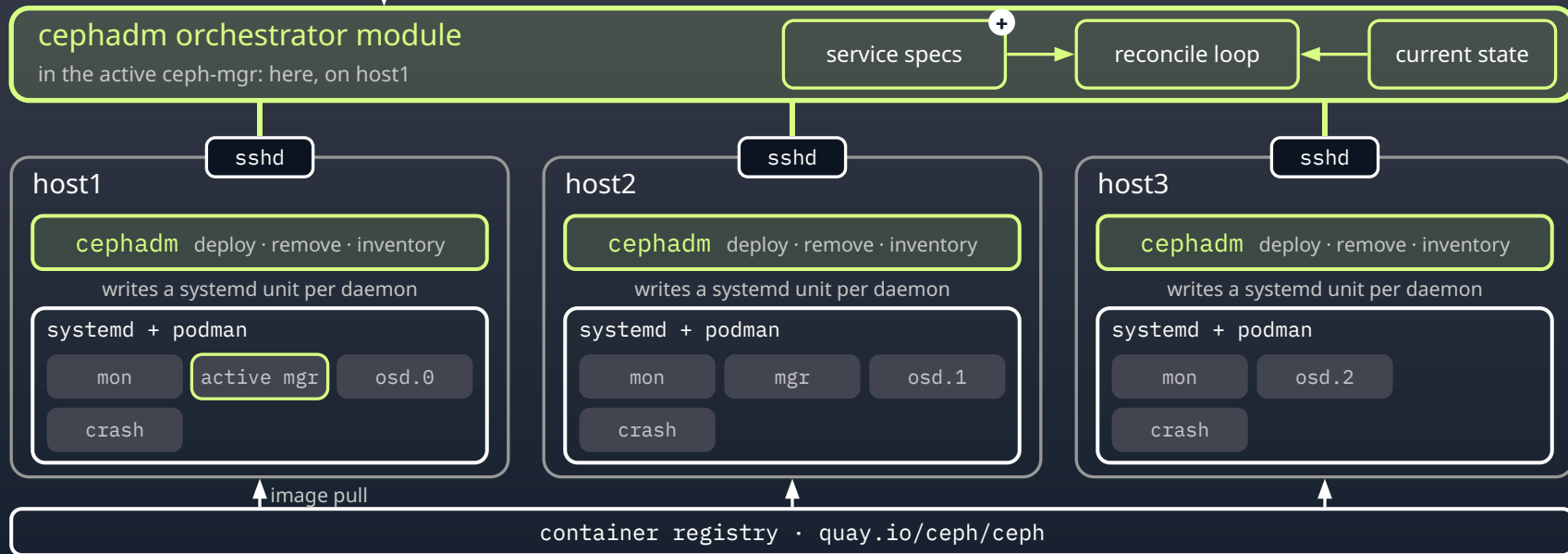
in the active ceph-mgr: here, on host1



cephadm: the CLI

you talk to it with the ceph CLI

```
admin $ ceph orch apply -i spec.yaml
```



01.

What is the setup behind
Rook?

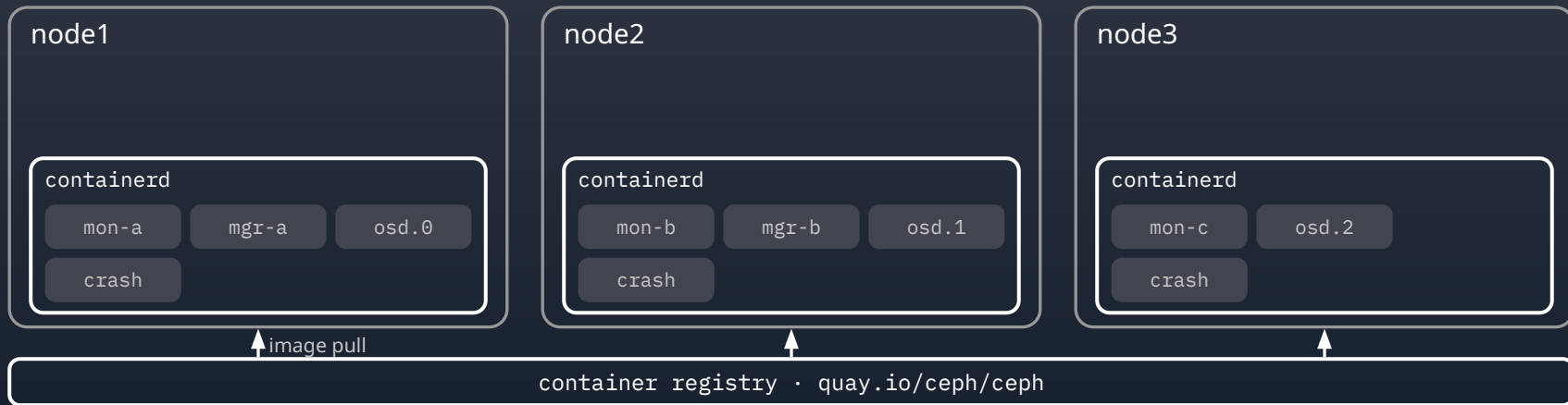
cephadm → Kubernetes

if you know cephadm, you already know Rook's building blocks



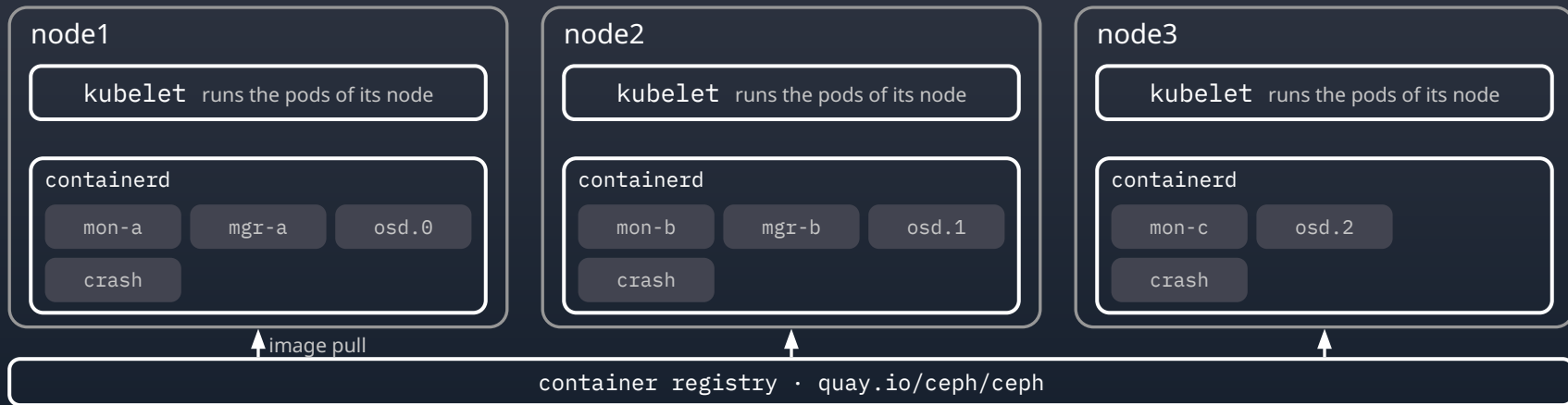
Rook: pods

every daemon is a pod, run by the kubelet



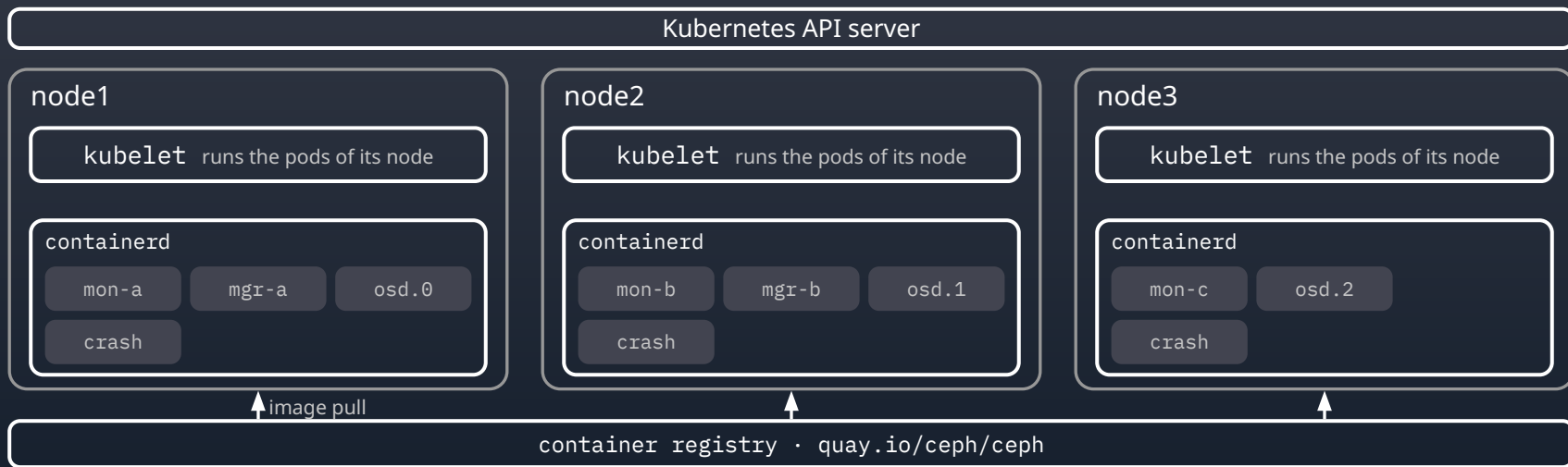
Rook: pods

every daemon is a pod, run by the kubelet



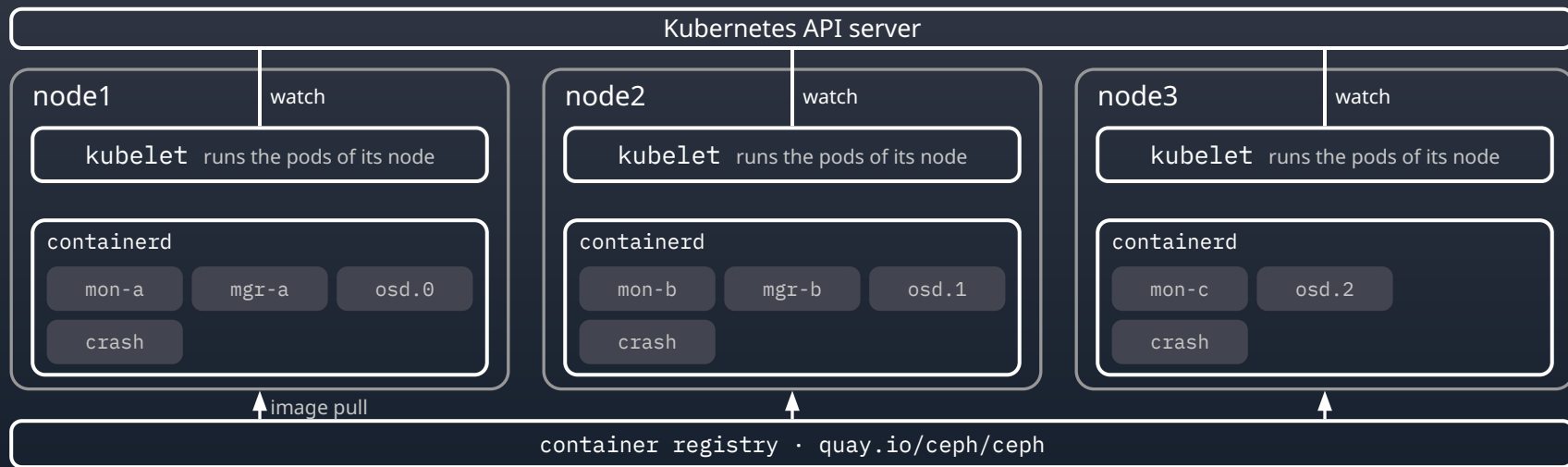
Rook: the API server

the API server is the hub



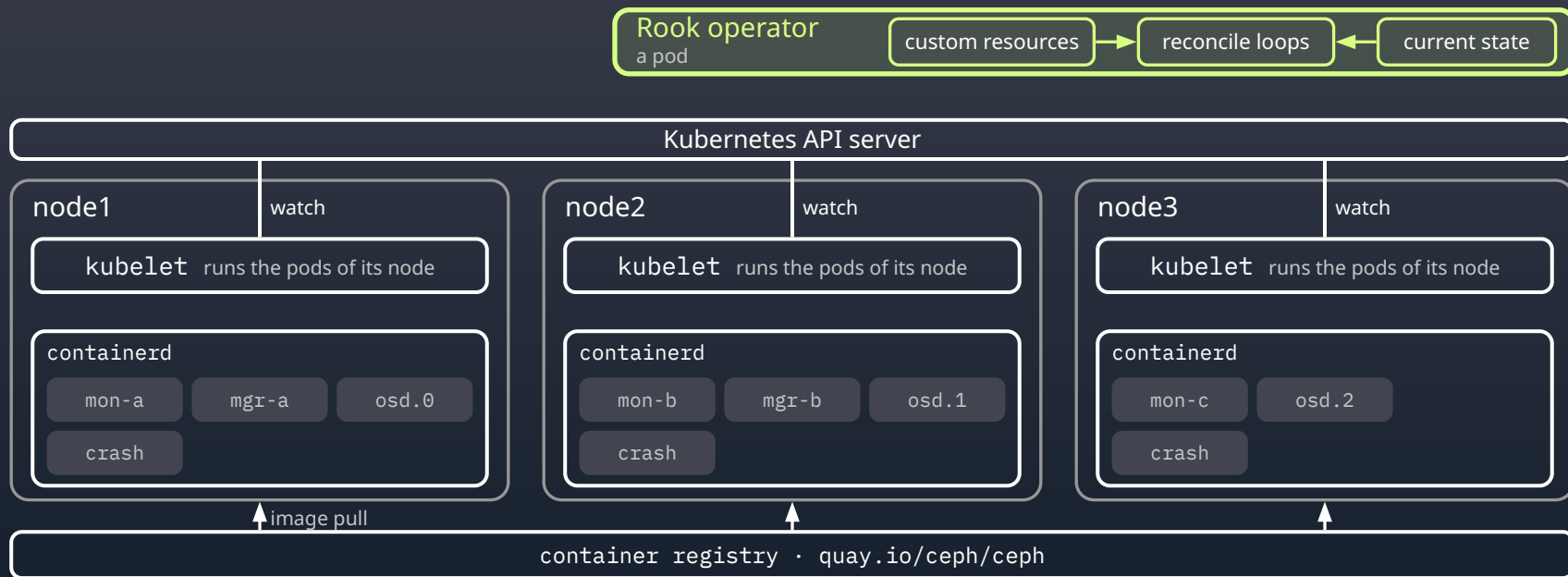
Rook: the API server

the API server is the hub



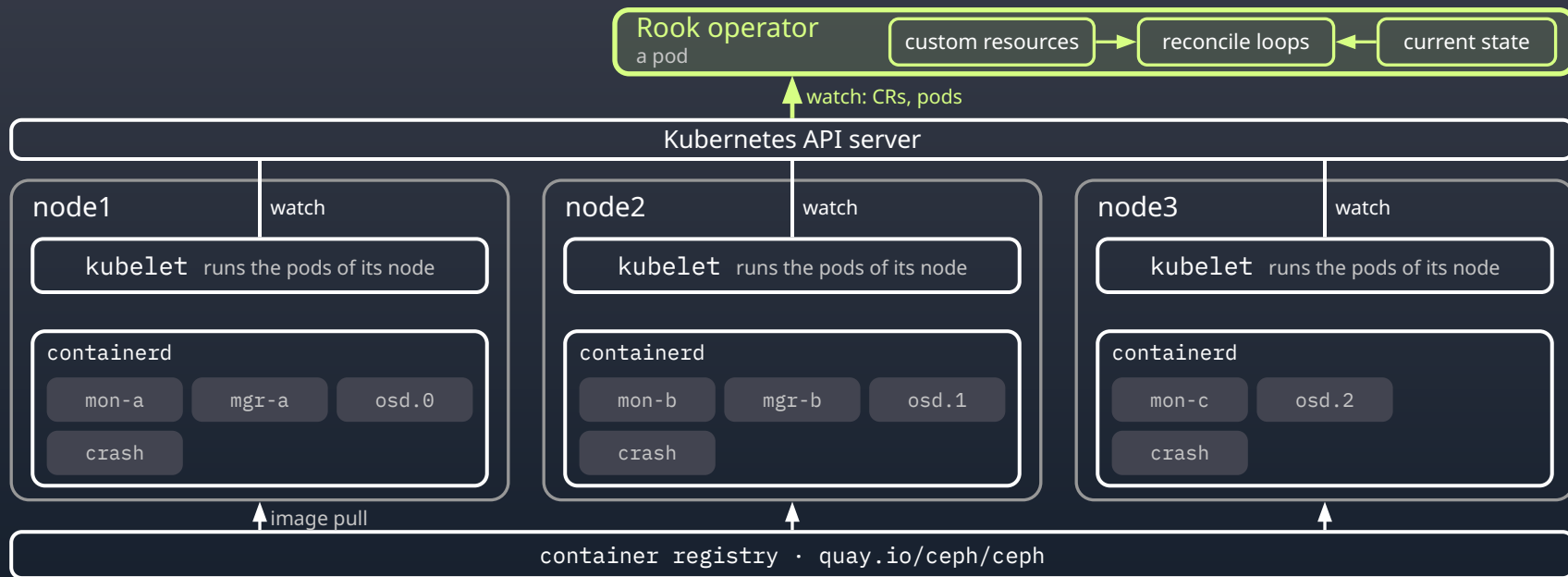
Rook: the operator

the operator only writes objects



Rook: the operator

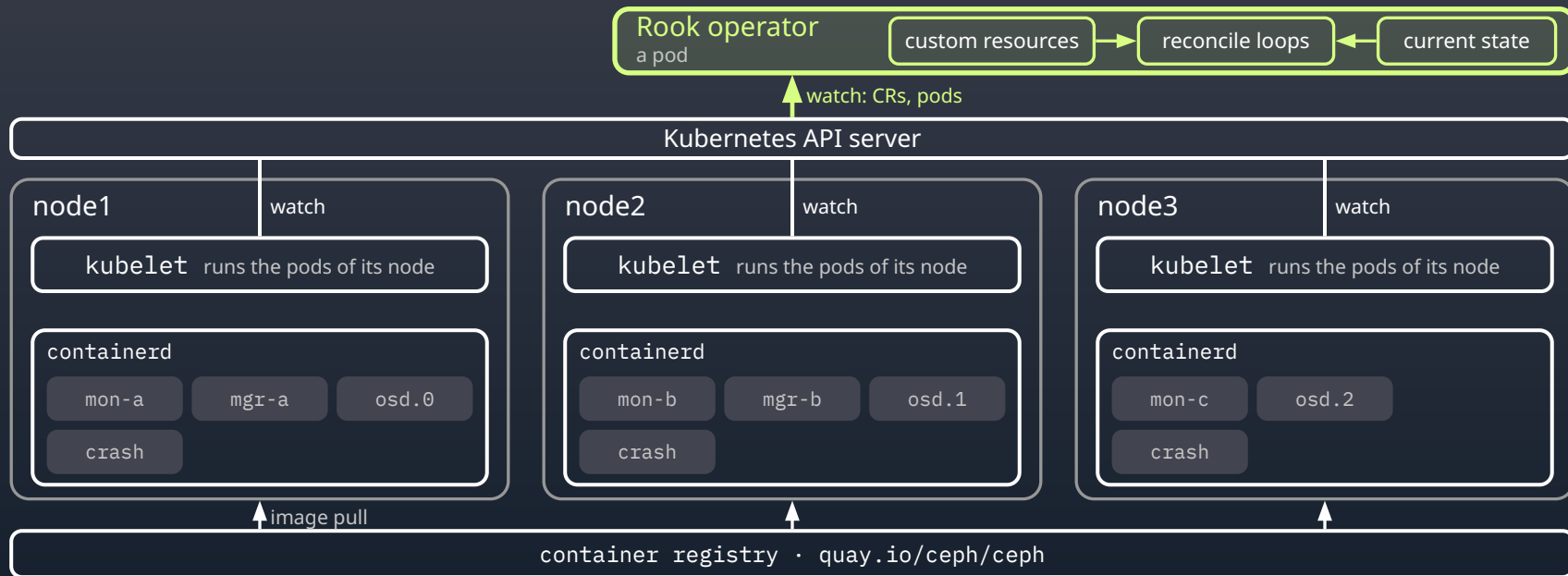
the operator only writes objects



Rook: the operator

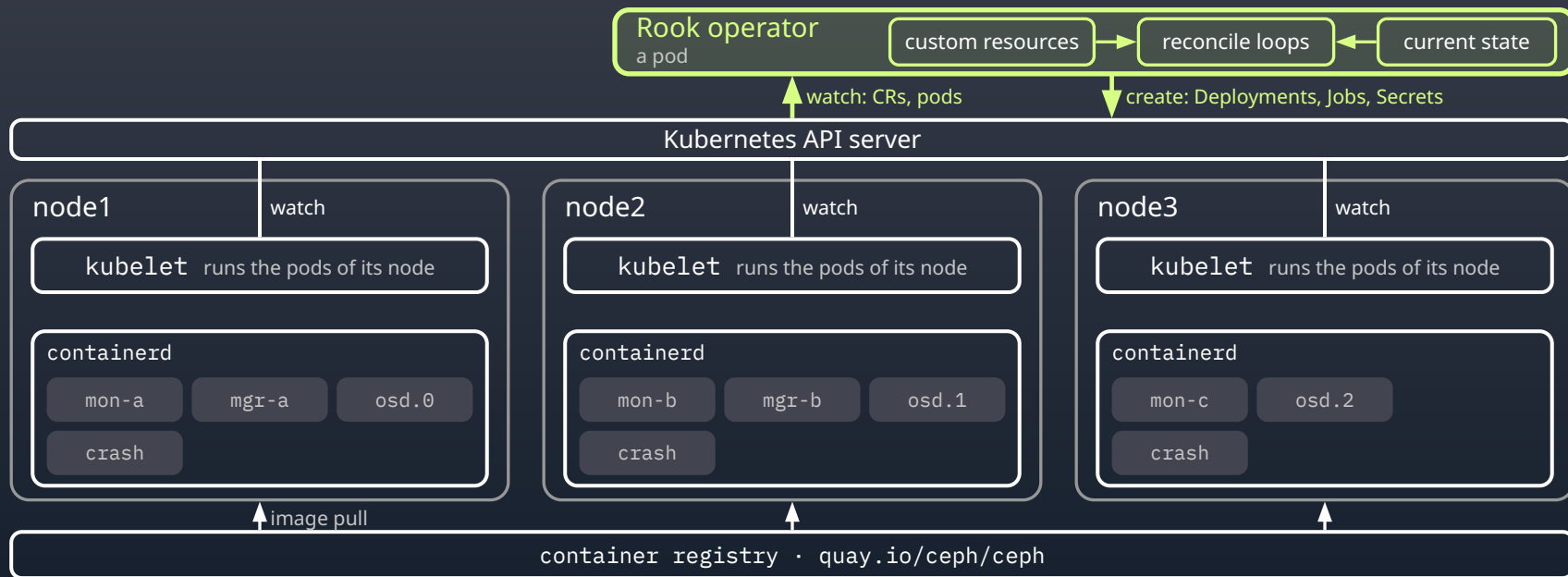
the operator only writes objects

the operator picks changes up through its watch



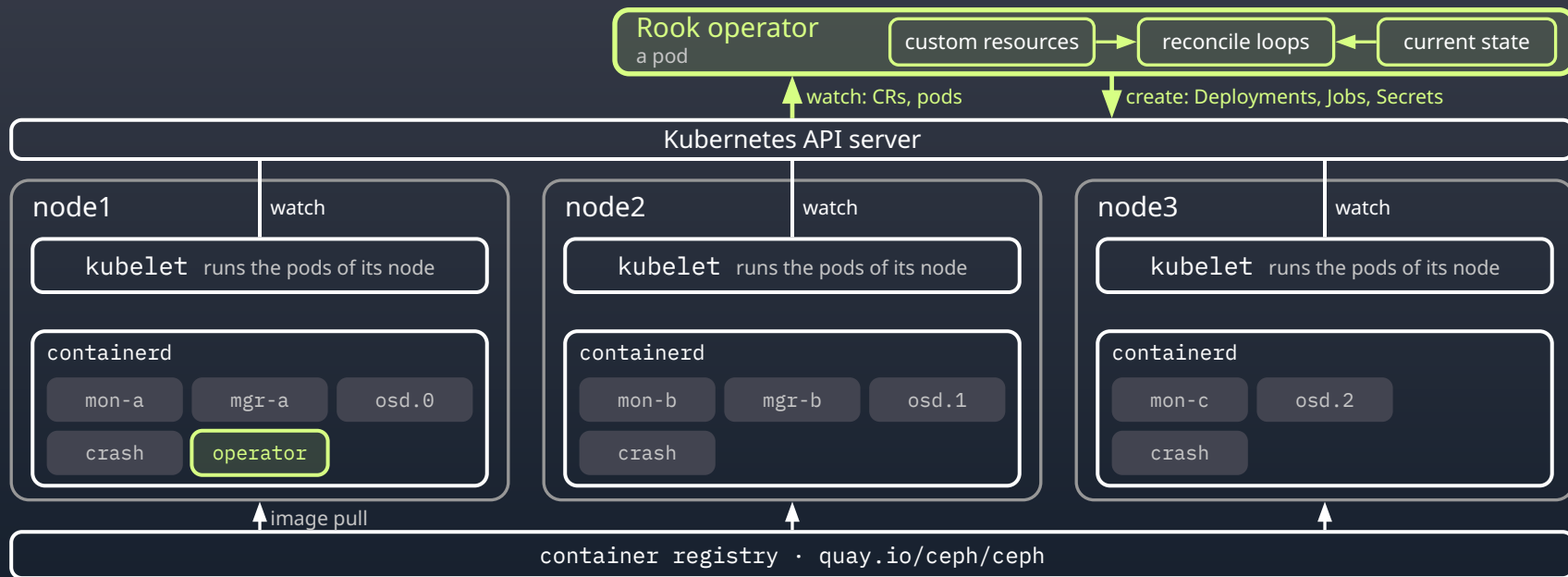
Rook: the operator

the operator only writes objects



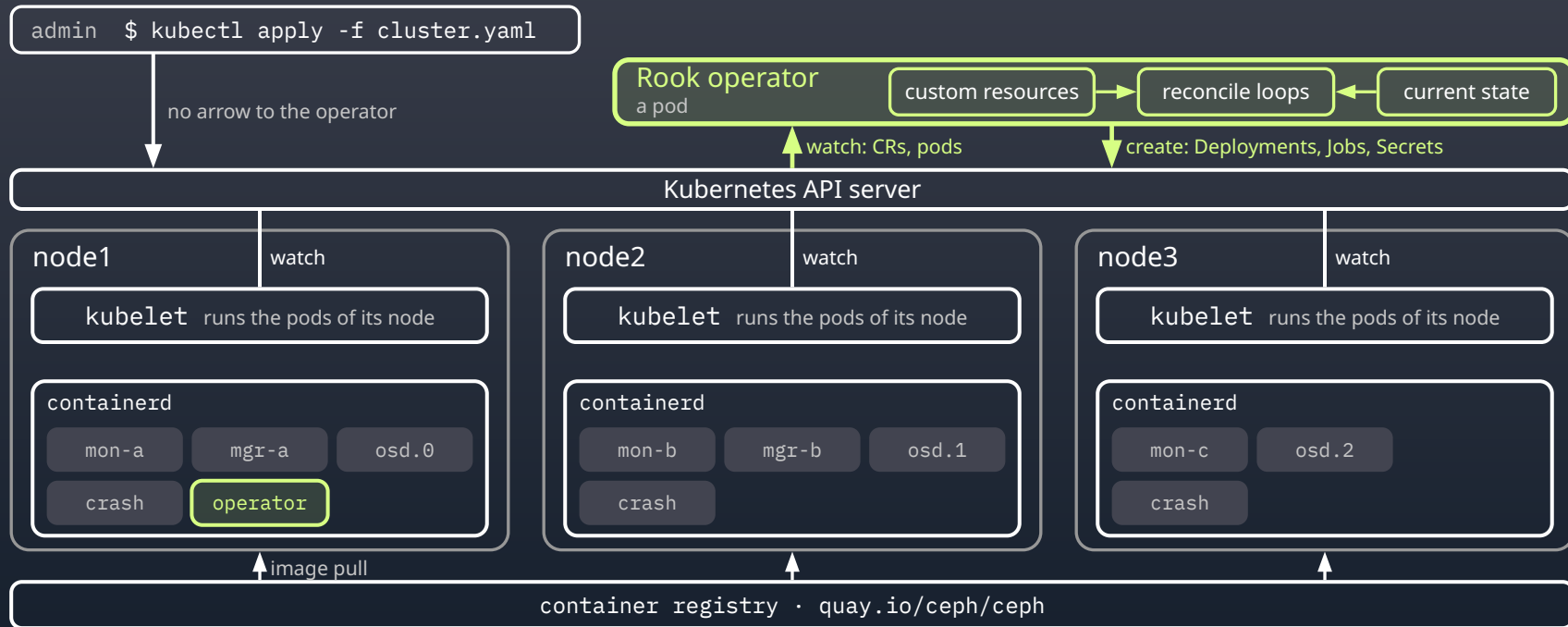
Rook: the operator

the operator only writes objects



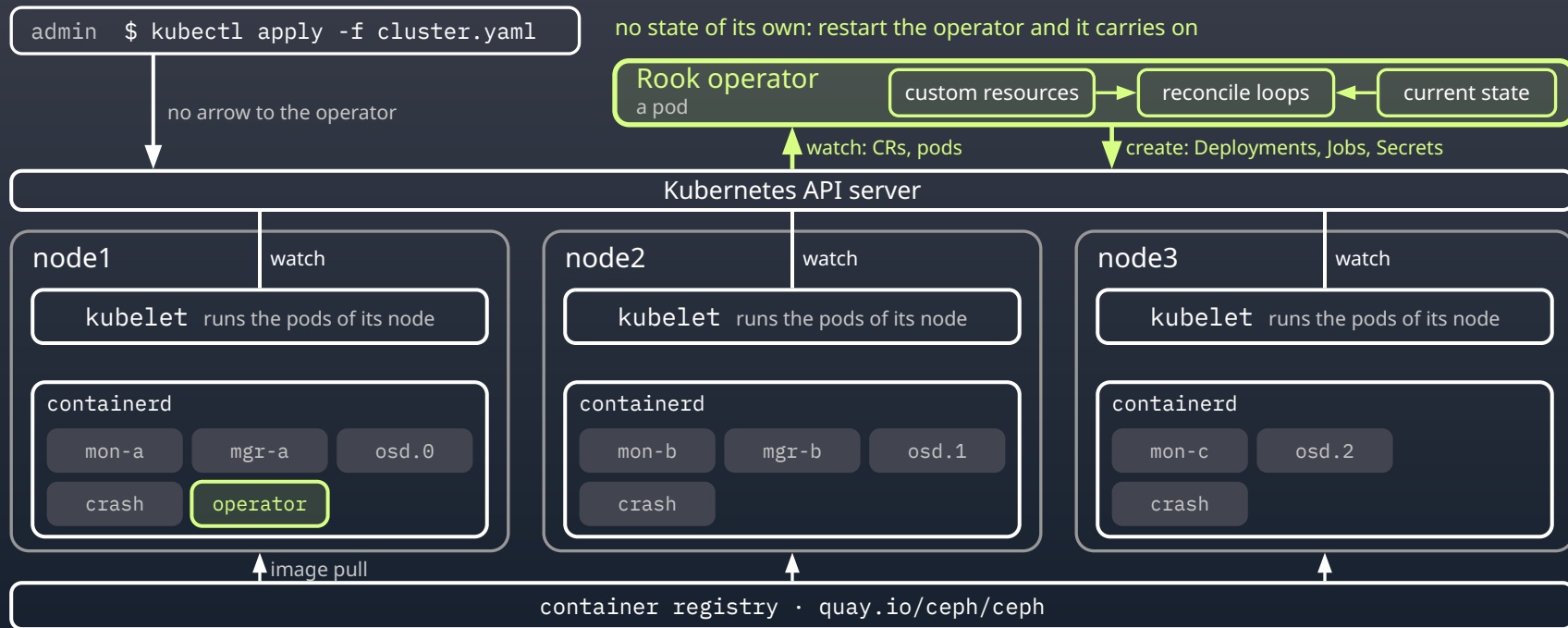
Rook: kubectl

you talk to Kubernetes, not to Rook



Rook: kubectl

you talk to Kubernetes, not to Rook



Side by side

	cephadm	Rook
where the brain runs	inside Ceph: a mgr module	outside Ceph: an operator pod
path to the hosts	SSH, pushed by the mgr	API server; the kubelet pulls
agent on the host	cephadm binary + systemd	kubelet, no Ceph logic
desired state in	the mon store	the API server (etcd)
you talk to	ceph orch	kubectl

 two cells: they differ  one bar: the same in both

What starts a reconcile?

cephadm: starting a reconcile

one reconcile at a time: refresh the hosts, then apply every spec

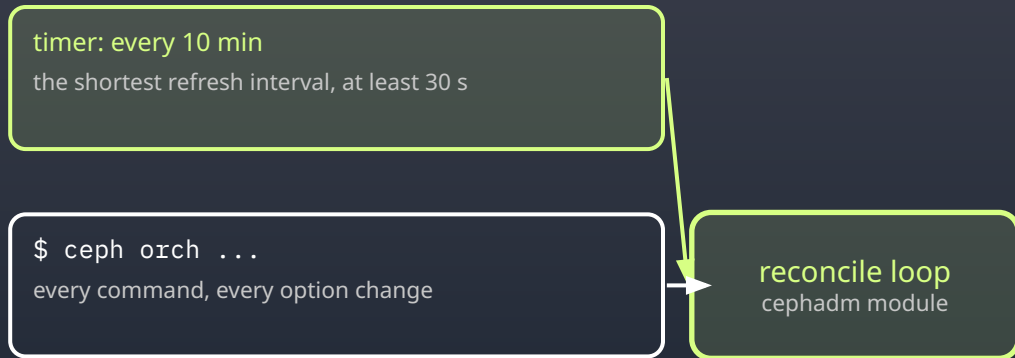
timer: every 10 min

the shortest refresh interval, at least 30 s

reconcile loop
cephadm module

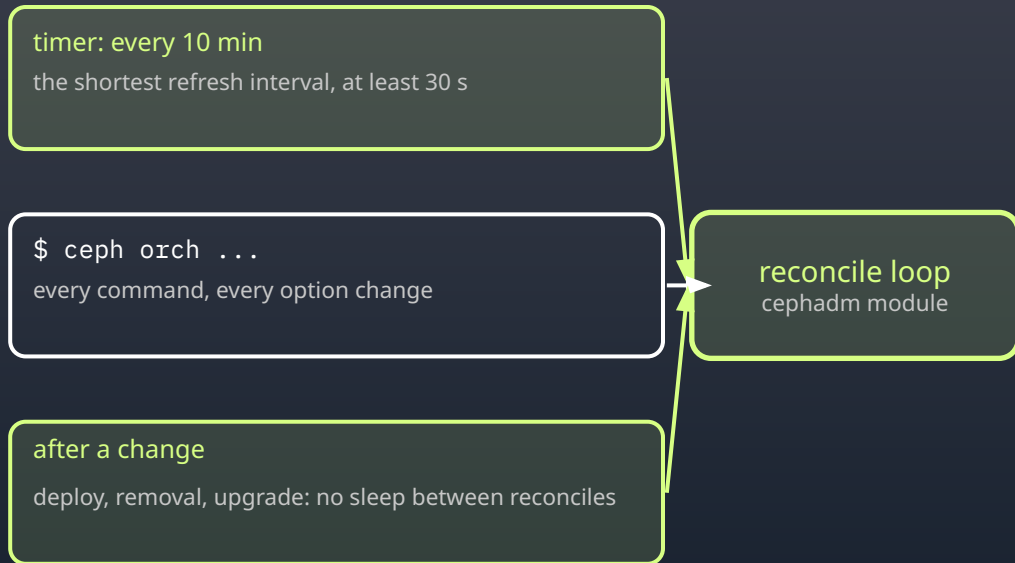
cephadm: starting a reconcile

one reconcile at a time: refresh the hosts, then apply every spec



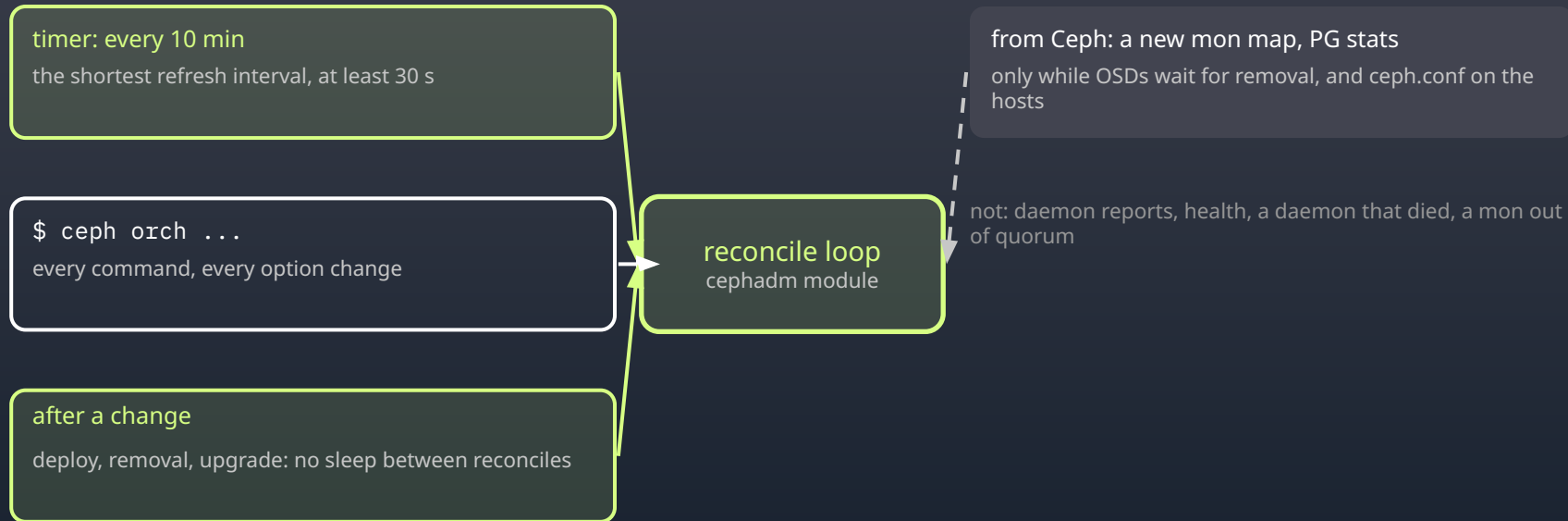
cephadm: starting a reconcile

one reconcile at a time: refresh the hosts, then apply every spec



cephadm: starting a reconcile

one reconcile at a time: refresh the hosts, then apply every spec



Rook: starting a reconcile

controllers wake up on events and requeues

watch: a resource created, changed or deleted
CRs, and the Deployments, Secrets, ConfigMaps it owns;
not status

reconcile loop
Rook operator



Rook



Kubernetes

Rook: starting a reconcile

controllers wake up on events and requeues

watch: a resource created, changed or deleted
CRs, and the Deployments, Secrets, ConfigMaps it owns;
not status

requeue: 10 s, 1 min, back-off
cluster not ready, upgrading, or after an error

reconcile loop
Rook operator



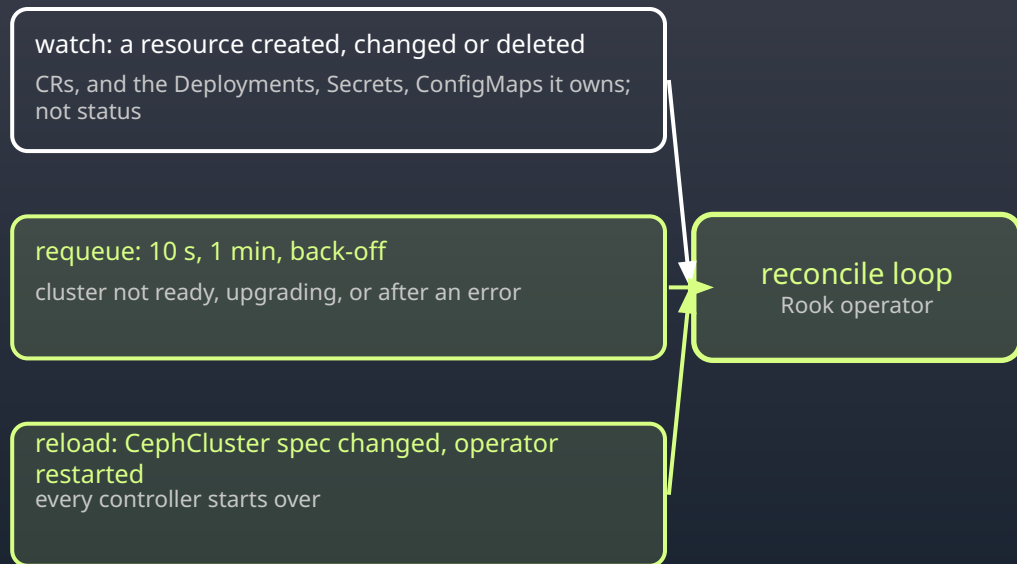
Rook



Kubernetes

Rook: starting a reconcile

controllers wake up on events and requeues



Rook: starting a reconcile

controllers wake up on events and requeues

watch: a resource created, changed or deleted
CRs, and the Deployments, Secrets, ConfigMaps it owns;
not status

requeue: 10 s, 1 min, back-off
cluster not ready, upgrading, or after an error

reload: CephCluster spec changed, operator
restarted
every controller starts over

reconcile loop
Rook operator

not: Ceph health, Kubernetes Events, a pod that died

Rook's health checks (mon 45 s, status and OSDs 60 s) run
beside the loop and act on their own: failover, cleanup

Timers versus "events"

cephadm

a timer and your commands;
Ceph itself almost never

Timers versus "events"

cephadm

a timer and your commands;
Ceph itself almost never

Rook

Kubernetes events and requeues;
Ceph itself never

02.

Day-2 operations

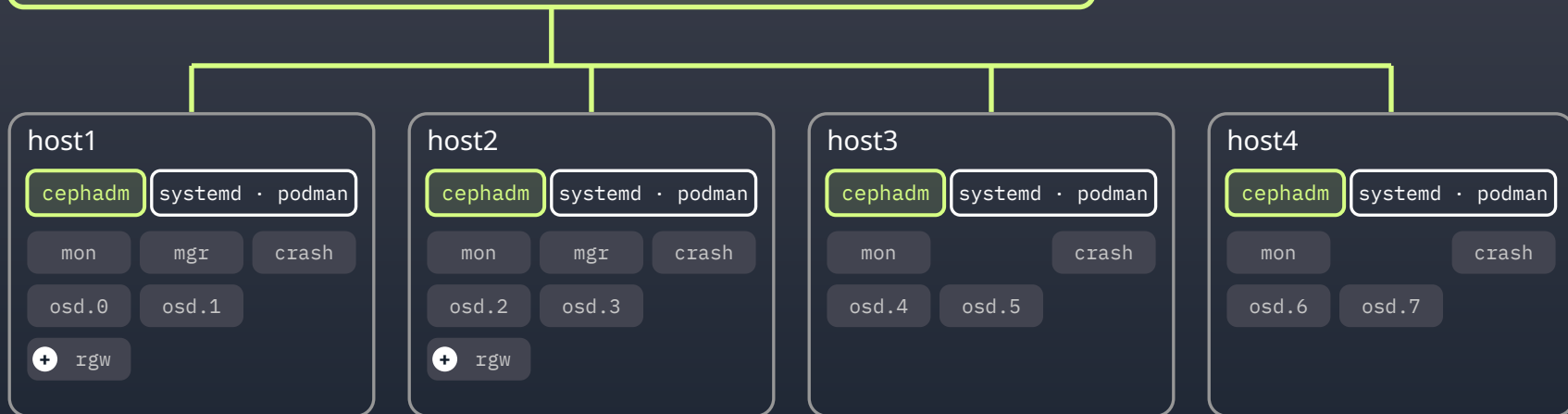
Object store: the scenario

Your users want S3.

What do you do?

cephadm: object store

cephadm orchestrator module in the active ceph-mgr

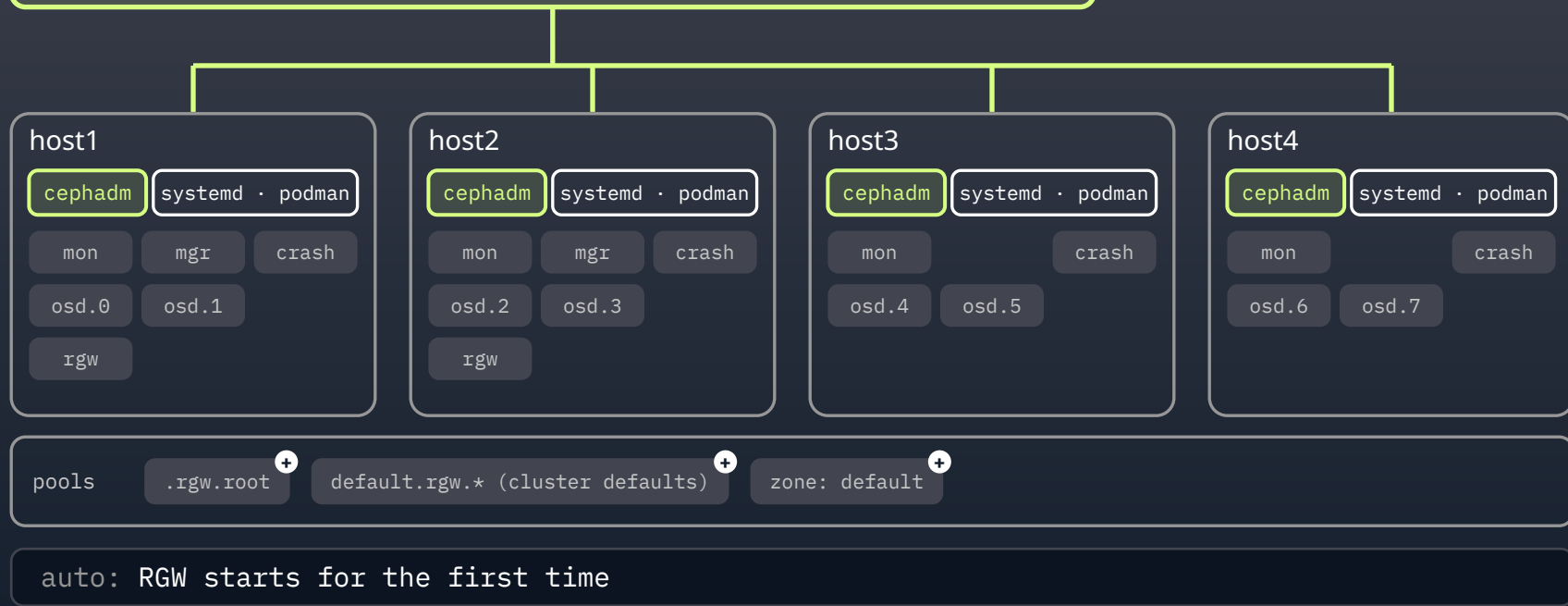


pools

```
$ ceph orch apply rgw s3 --placement="2 host1 host2"
```

cephadm: object store

cephadm orchestrator module in the active ceph-mgr



Rook: object store

Rook operator a pod on Kubernetes

Kubernetes API server

CephObjectStore s3

node1

kubelet · containerd

mon-a

mgr-a

crash

osd.0

osd.1

node2

kubelet · containerd

mon-b

mgr-b

crash

osd.2

osd.3

node3

kubelet · containerd

mon-c

crash

osd.4

osd.5

node4

kubelet · containerd

crash

osd.6

osd.7

pools

```
$ kubectl apply -f object-store.yaml
```



Rook



Kubernetes



Ceph



new

Rook: object store

Rook operator a pod on Kubernetes

Kubernetes API server

CephObjectStore s3

node1

kubelet · containerd

mon-a

mgr-a

crash

osd.0

osd.1

node2

kubelet · containerd

mon-b

mgr-b

crash

osd.2

osd.3

node3

kubelet · containerd

mon-c

crash

osd.4

osd.5

node4

kubelet · containerd

crash

osd.6

osd.7

pools

.rgw.root

s3.rgw.* (replicated 3)

s3.rgw.buckets.data (EC 2+1)

realm, zone: s3

auto: CephObjectStore s3 created



Rook



Kubernetes



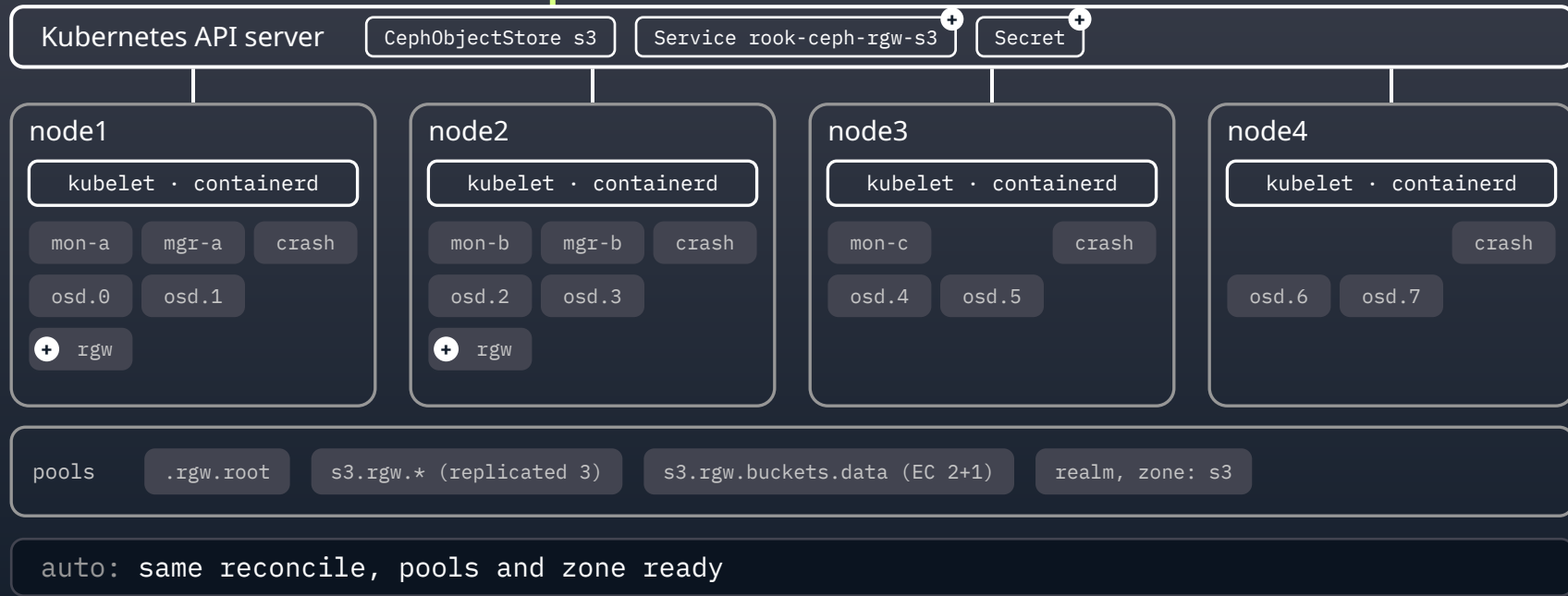
Ceph



new

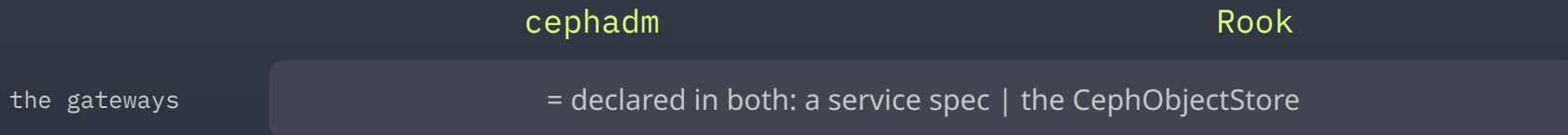
Rook: object store

Rook operator a pod on Kubernetes



Object store: what's different

once RGW is up: cephadm hands it to you, Rook keeps managing it as resources



Object store: what's different

once RGW is up: cephadm hands it to you, Rook keeps managing it as resources

	cephadm	Rook
the gateways	= declared in both: a service spec the CephObjectStore	
users, keys	radosgw-admin or the Admin Ops API	CephObjectStoreUser; keys in a Secret

 two cells: they differ  one bar: the same in both

Object store: what's different

once RGW is up: cephadm hands it to you, Rook keeps managing it as resources

	cephadm	Rook
the gateways	= declared in both: a service spec the CephObjectStore	
users, keys	radosgw-admin or the Admin Ops API	CephObjectStoreUser; keys in a Secret
buckets	over S3, or radosgw-admin	ObjectBucketClaim, or COSI

Object store: a side note

cephadm

RGW creates its pools:
one less thing to do

Rook

"We would strongly like to be able to disable
RGW's and radosgw-admin's default behavior
to create pools that don't exist."

Rook to Ceph, tracker #75457, March 2026

Same Ceph behaviour, opposite feelings.

— Patch a CVE: the scenario

A CephX CVE is out: CVE-2025-30156.

You need a fixed Ceph release, and keys of the new type.

key rotation: a new secret · key type transition: aes to aes256k, done by a rotation

In this example: Ceph 20.2.3 to 20.2.4, Rook v1.20.5 to v1.20.8.

cephadm: the command

```
1 ceph orch upgrade start --ceph-version 20.2.4
```

1 the update: the target version

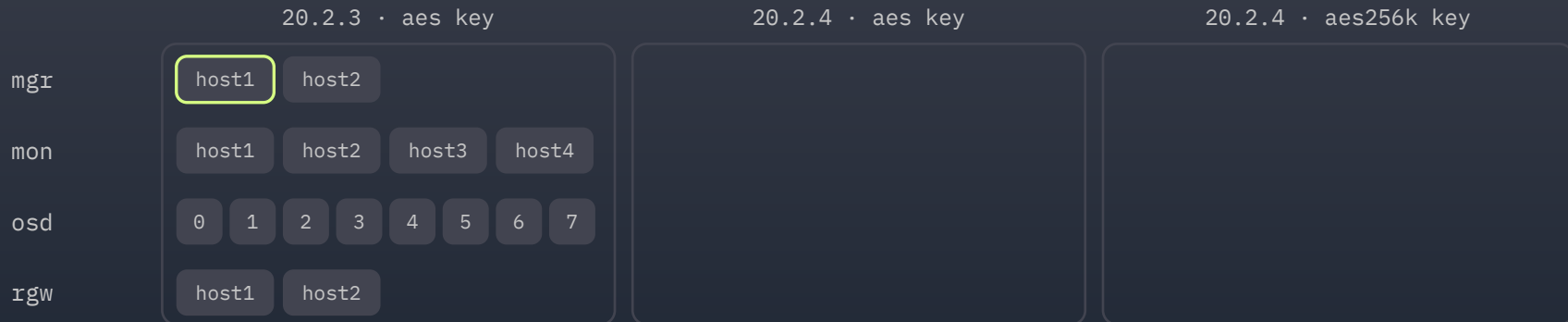
--- 2 the type transition: no flag, part of the upgrade

one thing in one command: the key type transition happens inside the upgrade

cephadm: upgrade and keys

cephadm orchestrator module runs the upgrade from the active mgr

first pull of 20.2.4 on one host pins the digest

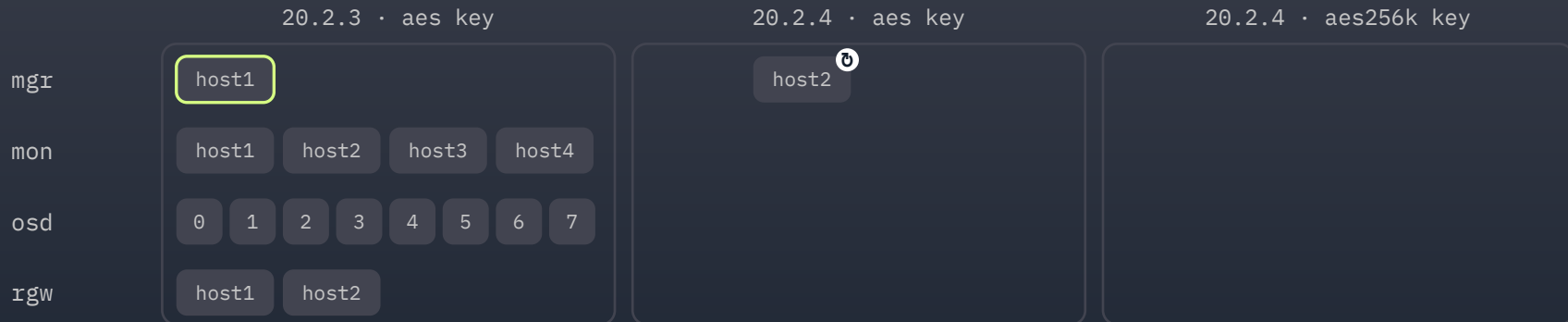


```
$ ceph orch upgrade start --ceph-version 20.2.4
```

 active mgr: runs the upgrade Ceph

cephadm: upgrade and keys

cephadm orchestrator module runs the upgrade from the active mgr



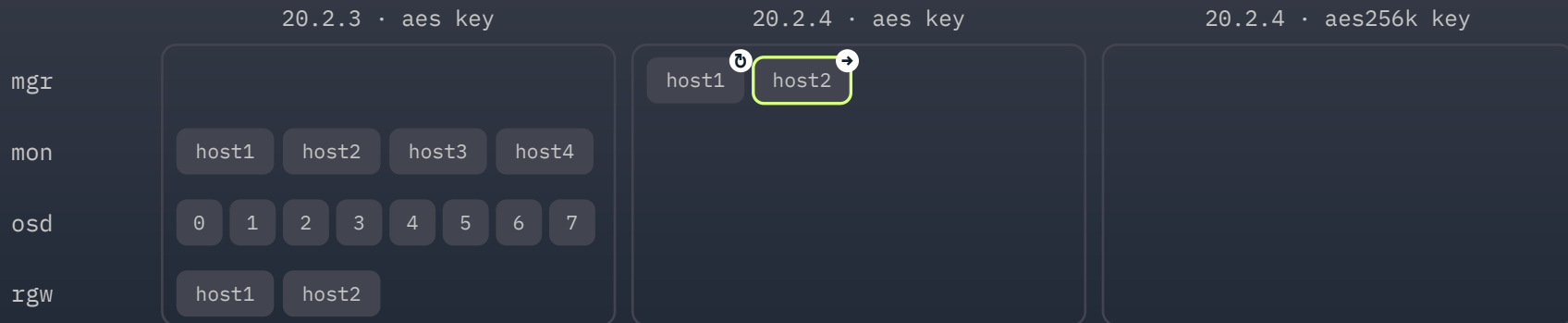
auto: next upgrade pass

active mgr: runs the upgrade Ceph restarted

cephadm: upgrade and keys

cephadm orchestrator module runs the upgrade from the active mgr

20.2.4 cephadm code drives the rest



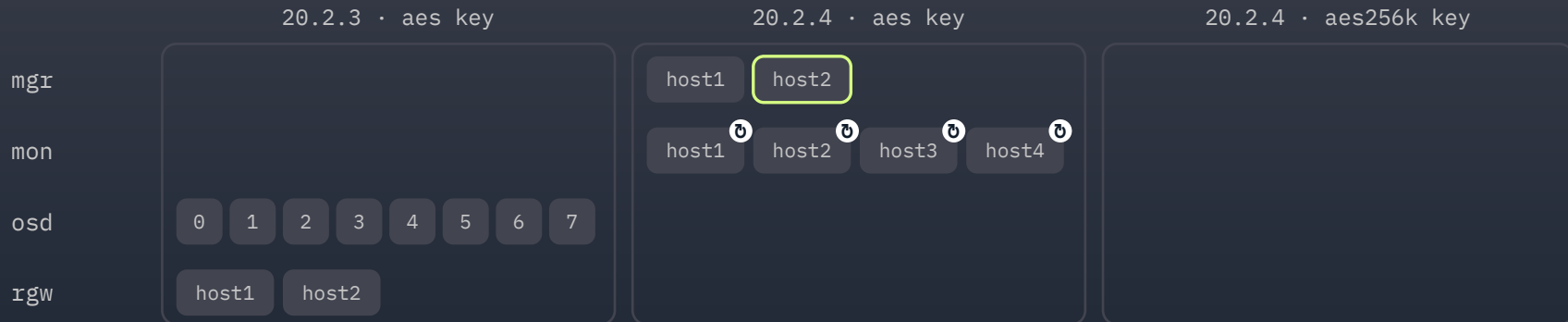
auto: mgr fail

active mgr: runs the upgrade Ceph restarted moved

cephadm: upgrade and keys

cephadm orchestrator module runs the upgrade from the active mgr

one mon per pass

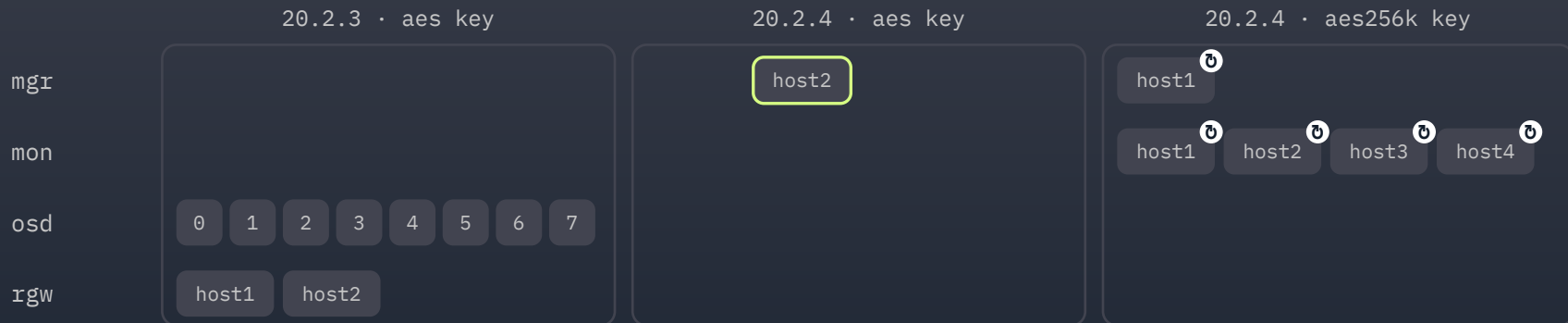


auto: mon ok-to-stop

 active mgr: runs the upgrade  Ceph  restarted

cephadm: upgrade and keys

cephadm orchestrator module runs the upgrade from the active mgr standby mgr first; then the mon. key, all mons back to back



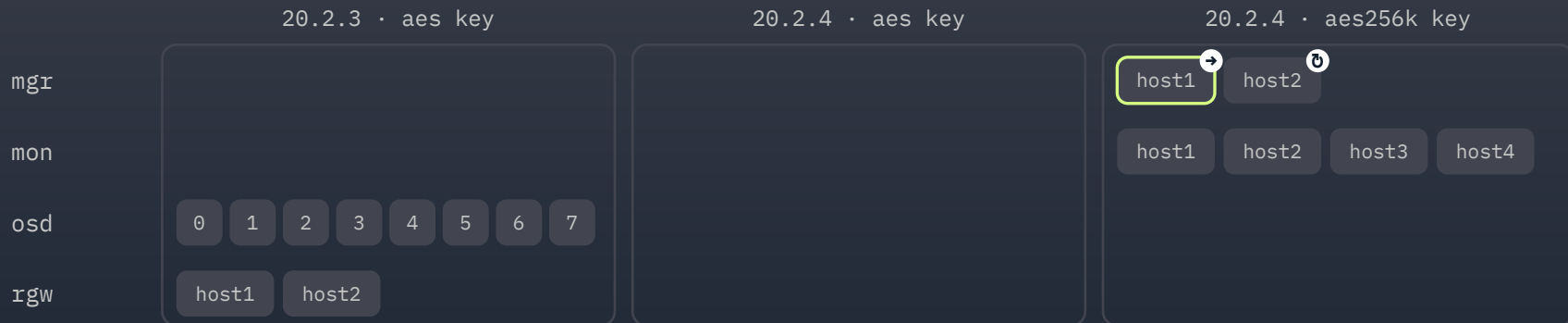
auto: all mons on 20.2.4

 active mgr: runs the upgrade  Ceph  restarted

cephadm: upgrade and keys

cephadm orchestrator module runs the upgrade from the active mgr

the active mgr cannot rotate itself: one more failover



auto: mgr fail, then the next pass

active mgr: runs the upgrade Ceph restarted moved

cephadm: upgrade and keys

cephadm orchestrator module runs the upgrade from the active mgr

OSDs in ok-to-stop batches, then RGW



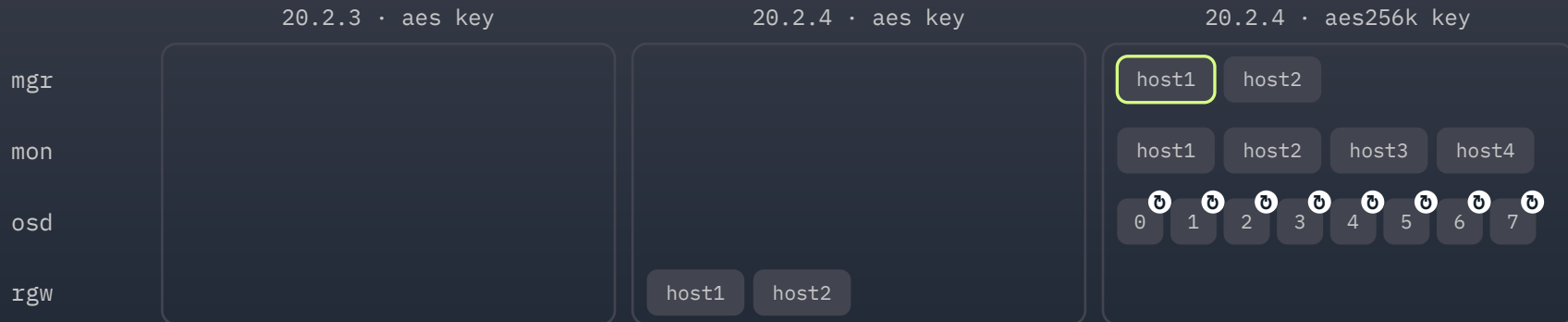
auto: osd ok-to-stop, per batch

 active mgr: runs the upgrade  Ceph  restarted

cephadm: upgrade and keys

cephadm orchestrator module runs the upgrade from the active mgr

one at a time: ok-to-stop, rotate, redeploy



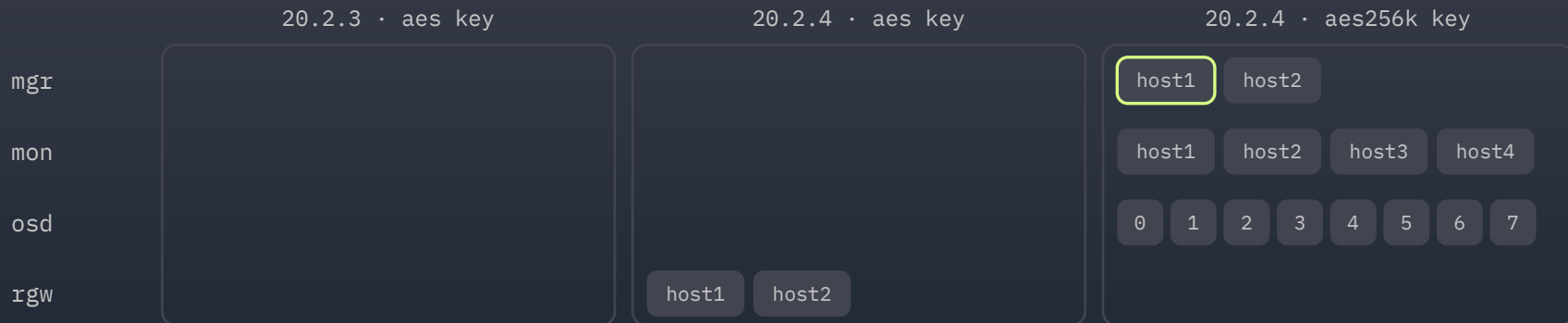
auto: every OSD on 20.2.4

 active mgr: runs the upgrade Ceph restarted

cephadm: upgrade and keys

cephadm orchestrator module runs the upgrade from the active mgr

not rotated: rgw, crash, nfs, client.admin, clients



auto: all daemons on 20.2.4

active mgr: runs the upgrade Ceph

Rook: the patch

```
spec:
  cephVersion:
    ❶ image: quay.io/ceph/ceph:v20.2.4
    security:
      cephx:
        daemon:
          ❷ keyRotationPolicy: KeyGeneration
            keyGeneration: 2
```

❶ the update: a new image

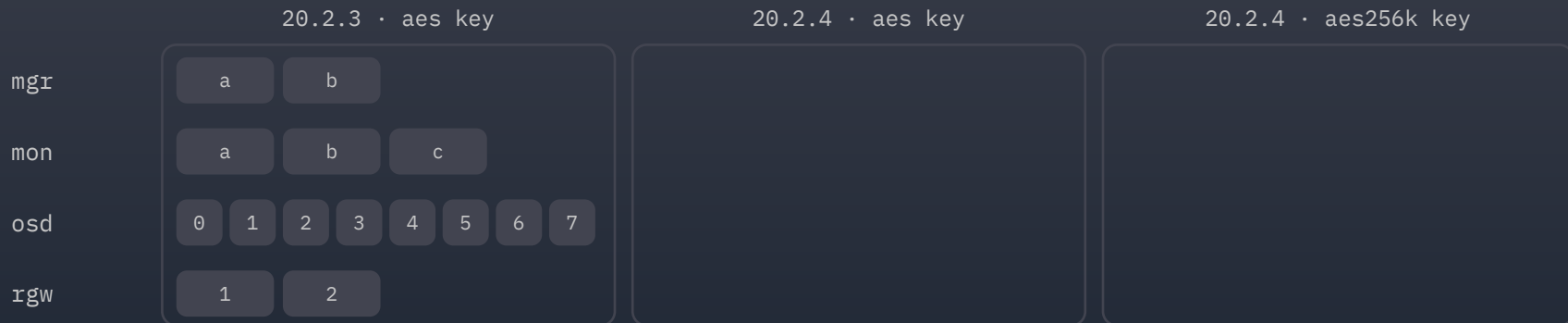
❷ the key rotation, here to aes256k: policy and generation

two things in one patch: an update and a key rotation; the CVE guide combines them

Rook: upgrade and keys

Rook operator v1.20.8, measured

update Rook

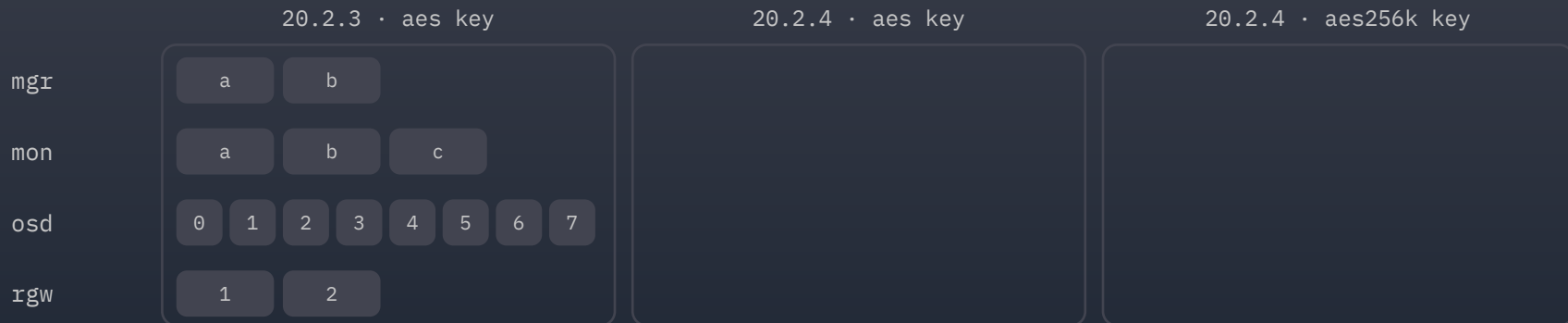


```
$ helm upgrade rook-ceph ... (or the manifests)
```

Rook: upgrade and keys

Rook operator v1.20.8, measured

the operator reloads all its controllers

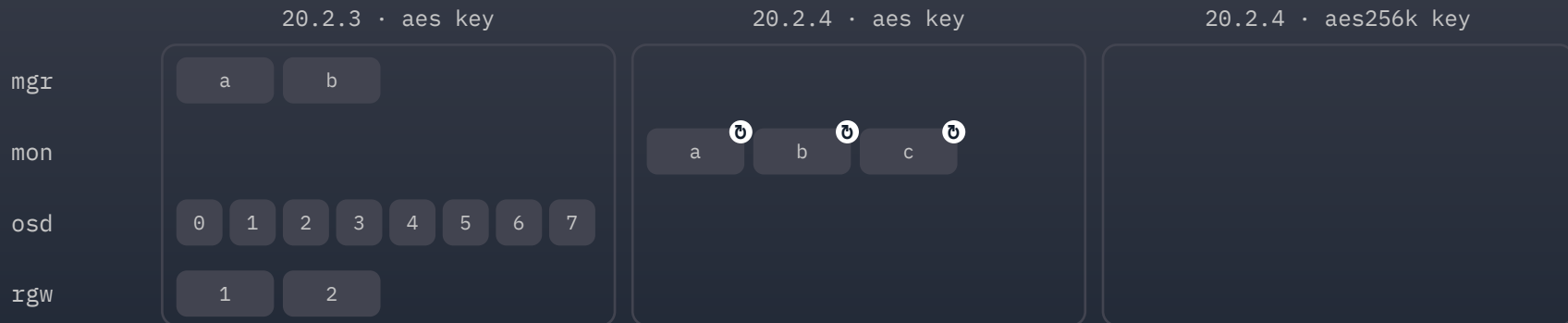


```
$ kubectl patch cephcluster ... (image v20.2.4, keyGeneration: 2)
```

Rook: upgrade and keys

Rook operator v1.20.8, measured

one mon at a time, with ok-to-stop

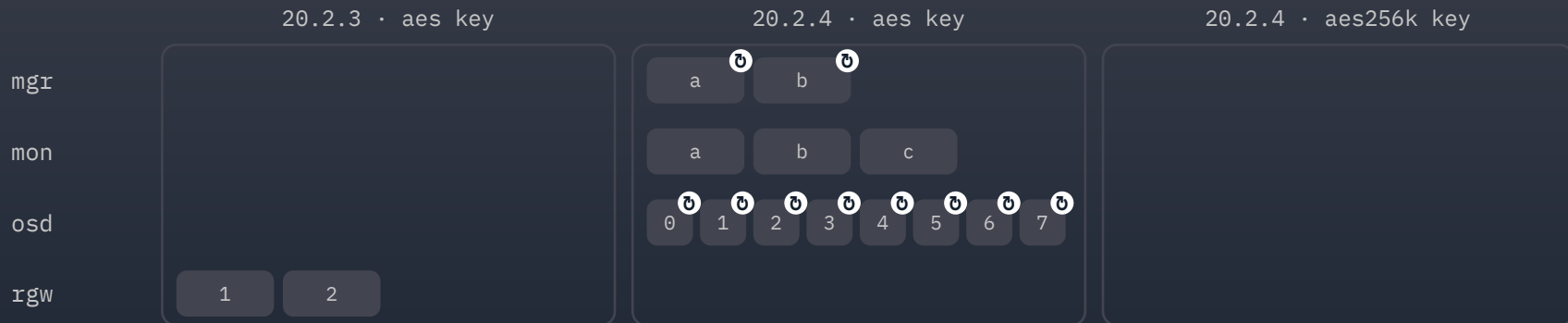


auto: mon ok-to-stop

➤ Rook: upgrade and keys

Rook operator v1.20.8, measured

key rotation held back: OSDs still on 20.2.3

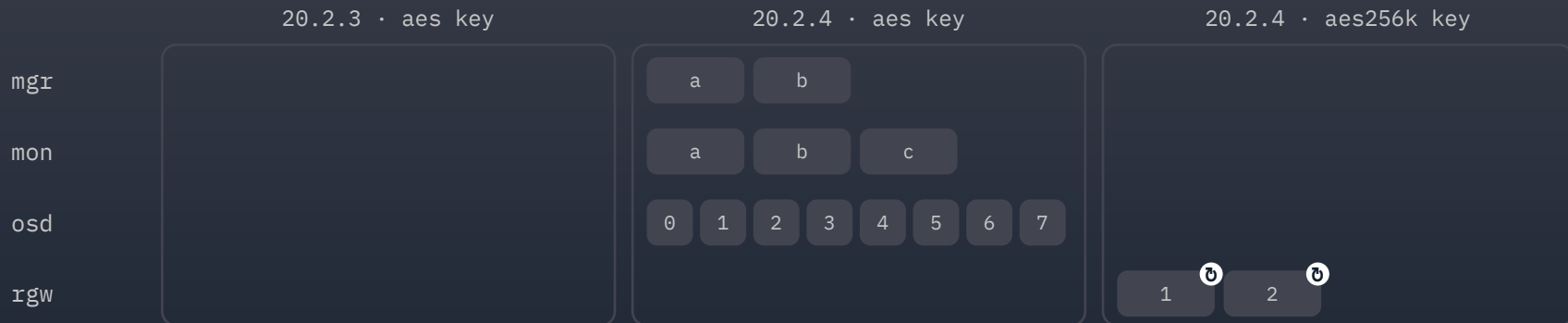


auto: same reconcile, mgr then OSDs

➤ Rook: upgrade and keys

Rook operator v1.20.8, measured

rotation allowed again; cluster waits 41 s at HEALTH_ERR

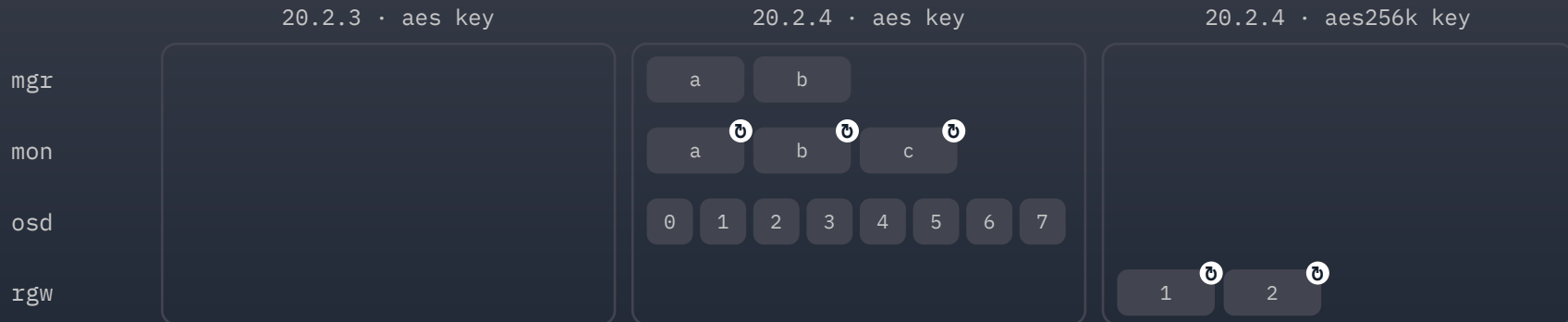


auto: all OSDs on 20.2.4

➤ Rook: upgrade and keys

Rook operator v1.20.8, measured

admin key: every mon restarts, and RGW once more

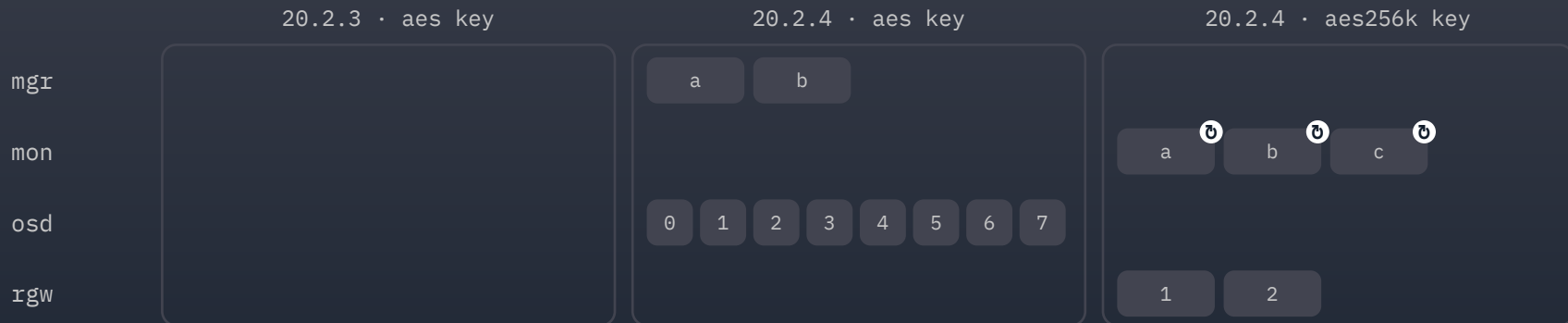


auto: no daemon left on 20.2.3

Rook: upgrade and keys

Rook operator v1.20.8, measured

mon. key: the mons restart a third time



auto: next reconcile after the reload

➤ Rook: upgrade and keys

Rook operator v1.20.8, measured

crash and exporter keys too, without restarts

20.2.3 · aes key

20.2.4 · aes key

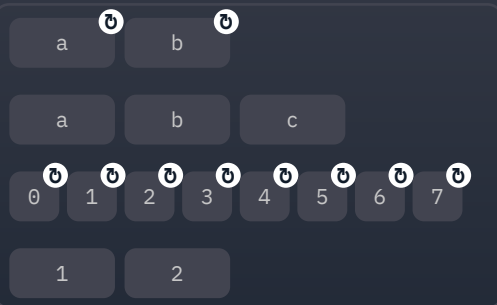
20.2.4 · aes256k key

mgr

mon

osd

rgw



auto: mons restarted

 Rook  Ceph  restarted

Rook: upgrade and keys

Rook operator v1.20.8, measured

cluster done after 9 min 55 s

20.2.3 · aes key

20.2.4 · aes key

20.2.4 · aes256k key

mgi

mon

osd

rgw

a

b

a

b

c

0

1

2

3

4

5

6

7

1

2

auto: mgrs and OSDs done

— CVE: what each tool assumes

	cephadm	Rook
keys	only the type transition, inside the upgrade	a rotation feature; the transition uses it

— CVE: what each tool assumes

	cephadm	Rook
keys	only the type transition, inside the upgrade	a rotation feature; the transition uses it
cluster health	no cluster check; ok-to-stop per daemon	must not be HEALTH_ERR, or it waits

— CVE: what each tool assumes

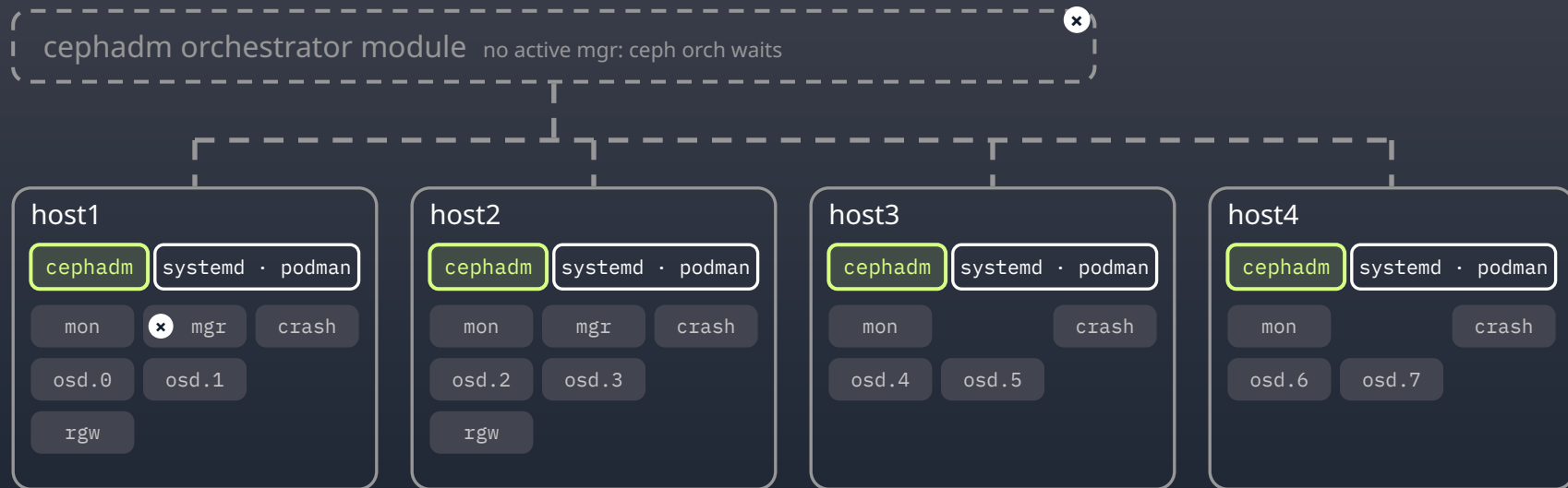
	cephadm	Rook
keys	only the type transition, inside the upgrade	a rotation feature; the transition uses it
cluster health	no cluster check; ok-to-stop per daemon	must not be HEALTH_ERR, or it waits
clients	= clients are yours to update in both	

— The brain dies: the scenario

The orchestrator itself goes down.

cephadm: the active mgr. Rook: the operator pod. What keeps running, what stops?

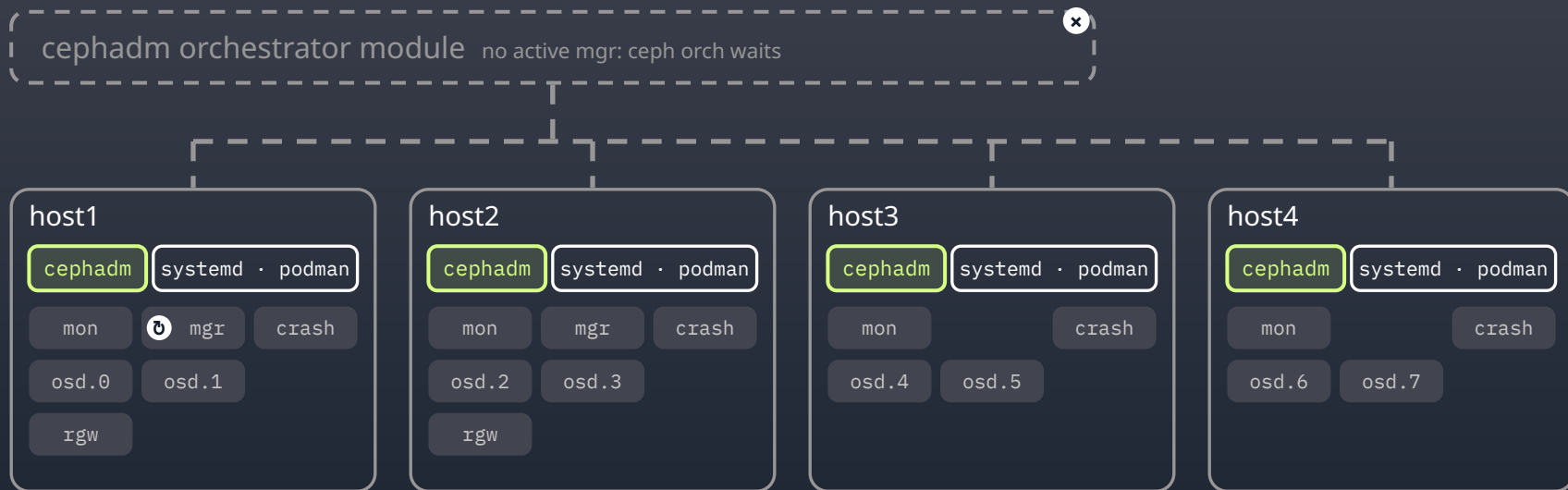
cephadm: the active mgr dies



Ceph keeps serving I/O; cephadm is gone with the mgr: no ceph orch

```
auto: ceph-mgr exits
```

cephadm: the active mgr dies

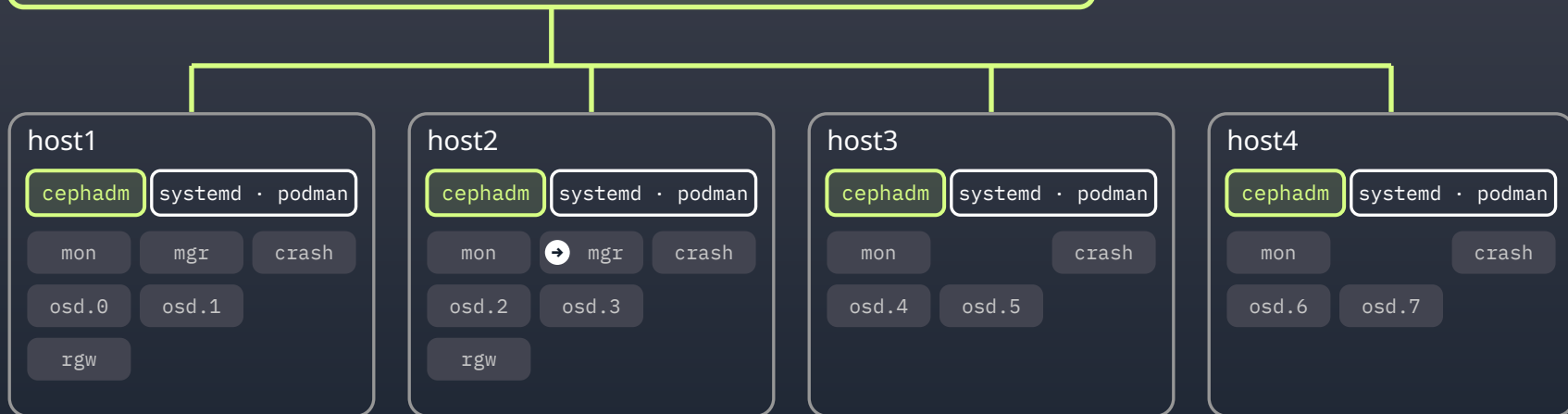


systemd restarts it after 10 s; it comes back as a standby, the mons still wait

```
auto: systemd Restart=on-failure
```

cephadm: the active mgr dies

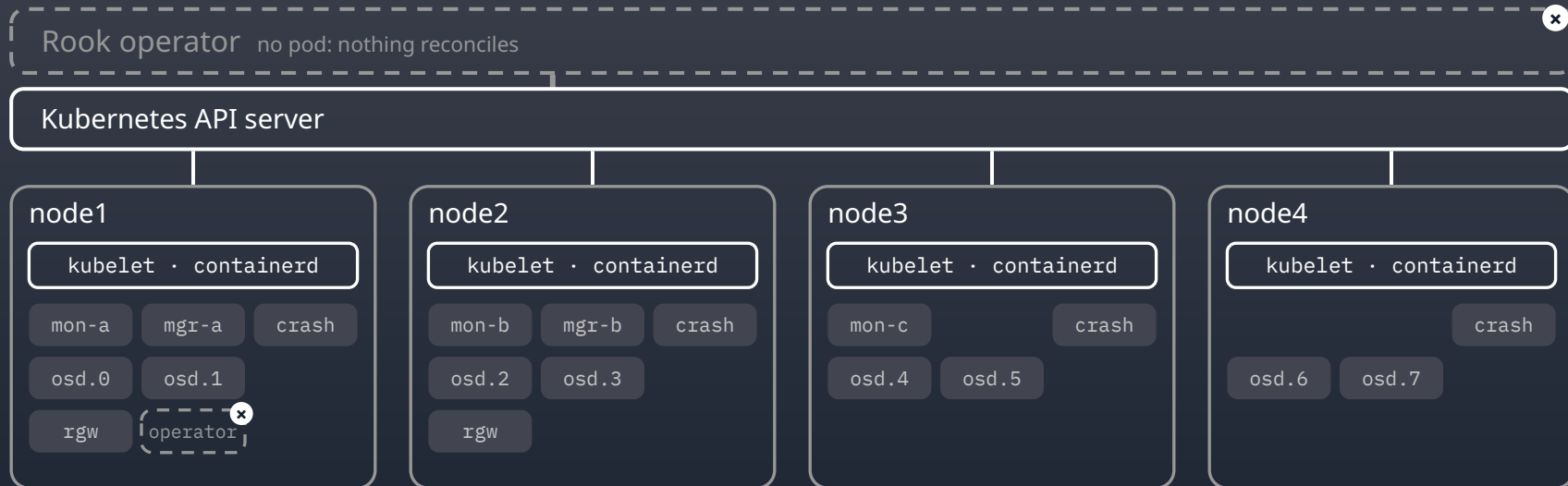
cephadm orchestrator module in the active ceph-mgr: now host2



30 s without a beacon: a standby is promoted; cephadm reloads its state from the mon store

```
auto: mon_mgr_beacon_grace: 30 s without a beacon
```

Rook: the operator dies



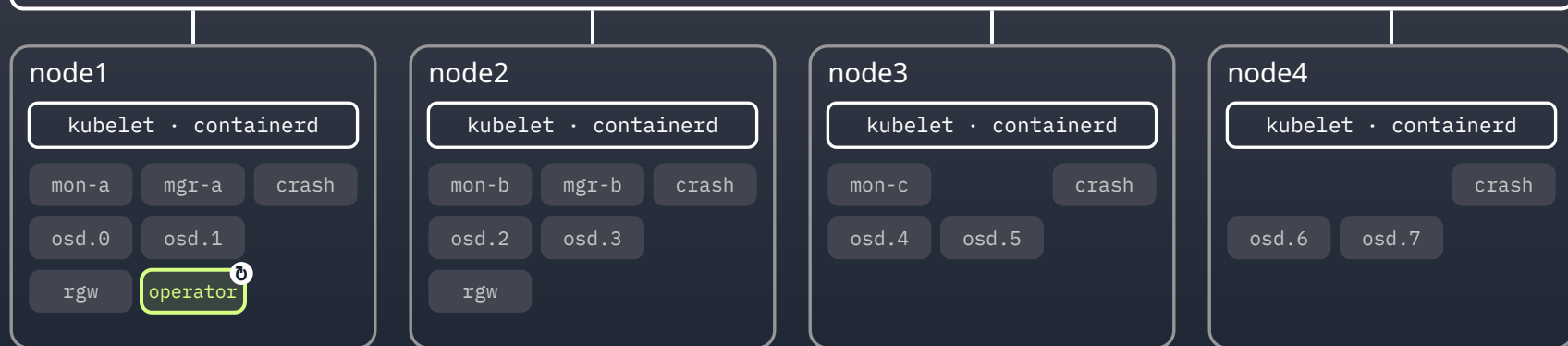
Ceph keeps serving I/O; no reconciles, no mon failover, no upgrades

auto: operator pod exits

Rook: the operator dies

Rook operator a pod on Kubernetes, like any other

Kubernetes API server



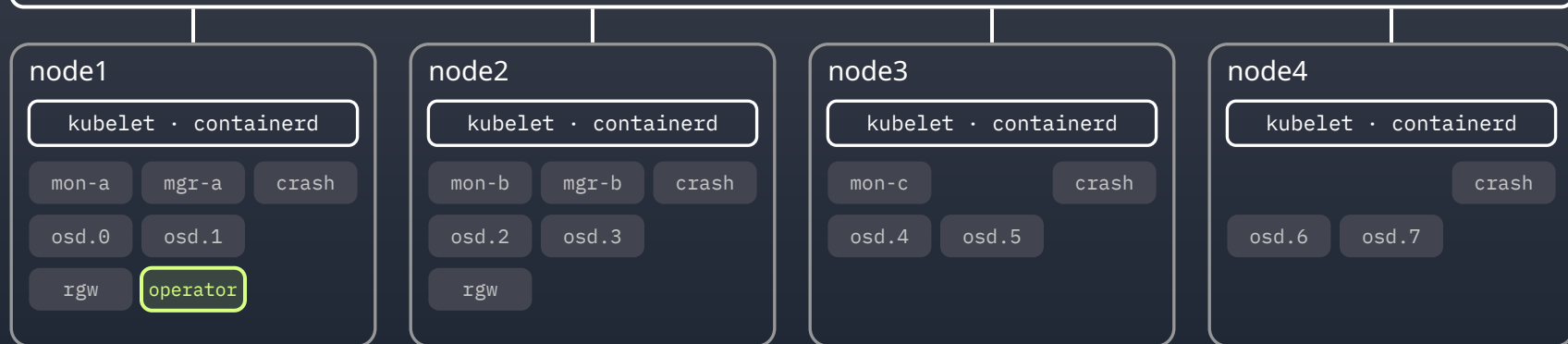
a normal pod: the kubelet restarts it; if the node is gone, a new pod after 5 min

```
auto: kubelet restartPolicy: Always
```

Rook: the operator dies

Rook operator a pod on Kubernetes, like any other

Kubernetes API server



every controller starts over from the CRs in etcd; an upgrade in flight resumes

auto: all controllers start, a reconcile of every CR

Brain dies: what's different

cephadm

Rook

the I/O path

= Ceph keeps serving I/O in both

 two cells: they differ one bar: the same in both

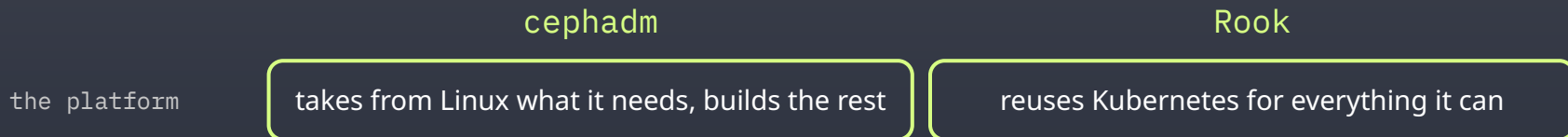
Brain dies: what's different

	cephadm	Rook
the I/O path	= Ceph keeps serving I/O in both	
a second brain	2 mgrs by default: a standby takes over	1 replica: Kubernetes recreates it

Brain dies: what's different

	cephadm	Rook
the I/O path	= Ceph keeps serving I/O in both	
a second brain	2 mgrs by default: a standby takes over	1 replica: Kubernetes recreates it
its memory	the mon store, safe with the quorum	etcd; back up the mon Secret and endpoints

Philosophy behind each tool



two cells: they differ one bar: the same in both

Philosophy behind each tool

	cephadm	Rook
the platform	takes from Linux what it needs, builds the rest	reuses Kubernetes for everything it can
the model	Ceph's own: specs, hosts, Ceph's defaults	Kubernetes': desired state, declared up front

 two cells: they differ  one bar: the same in both

Philosophy behind each tool

	cephadm	Rook
the platform	takes from Linux what it needs, builds the rest	reuses Kubernetes for everything it can
the model	Ceph's own: specs, hosts, Ceph's defaults	Kubernetes': desired state, declared up front
built for	Ceph people: hosts and the ceph CLI	Kubernetes people: kubectl and YAML

— A “rough” basis for choosing

Choose the home for the loop, not the better tool.

Choose on your system environment and context

Neither if tool is perfect. So choose the best trade off....

— One last thing

Knock knock.

Who's there?

One last thing

Knock knock.

Who's there?

cephadm

cephadm.

One last thing

Knock knock.

Who's there?

cephadm

cephadm.
cephadm who?

— One last thing

Knock knock.

Who's there?

cephadm

cephadm.
cephadm who?
cephadm I let myself in over SSH.

— One last thing

Knock knock.

Who's there?

cephadm

cephadm.
cephadm who?
cephadm I let myself in over SSH.

Rook

Rook.

— One last thing

Knock knock.

Who's there?

cephadm

cephadm.
cephadm who?
cephadm I let myself in over SSH.

Rook

Rook.
Rook who?

— One last thing

Knock knock.

Who's there?

cephadm

cephadm.
cephadm who?
cephadm I let myself in over SSH.

Rook

Rook.
Rook who?
Rook never knocks.

— One last thing

Knock knock.

Who's there?

cephadm

cephadm.
cephadm who?
cephadm I let myself in over SSH.

Rook

Rook.
Rook who?
Rook never knocks.
It asks Kubernetes (kubelet) to
open the door.

Thank you

Questions?

