MODERN INFRASTRUCTURE & IaC

etcd Is Your Kubernetes Database

What it does. What breaks. What to watch.

Most teams don't think about etcd until the control plane stops.

Where Does Kubernetes Store State?

✗ Your pods

✗ Your nodes

✗ Your scheduler

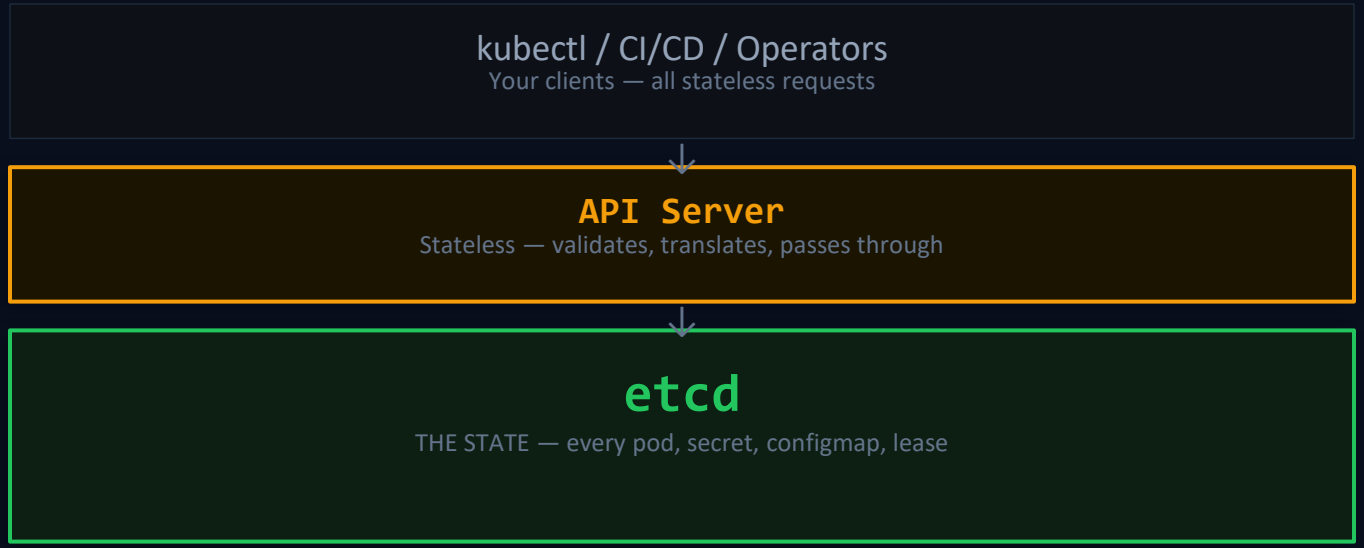
✗ Your API server

etcd

Source of Truth
All cluster state lives here

Kubernetes is a state machine. etcd is the state.

The Control Plane Stack



No fallback. No secondary store. Everything depends on etcd being readable and writable.

etcd Failures Don't Look Like Database Failures

They look like Kubernetes acting weird.

`kubectl get pods` hangs

API server latency spikes — writes to etcd not completing

Pods stuck in Pending / Terminating

Scheduler & kubelet waiting on state writes that aren't returning

Deployments not rolling

Controller watch loops lagging — etcd can't serve events fast enough

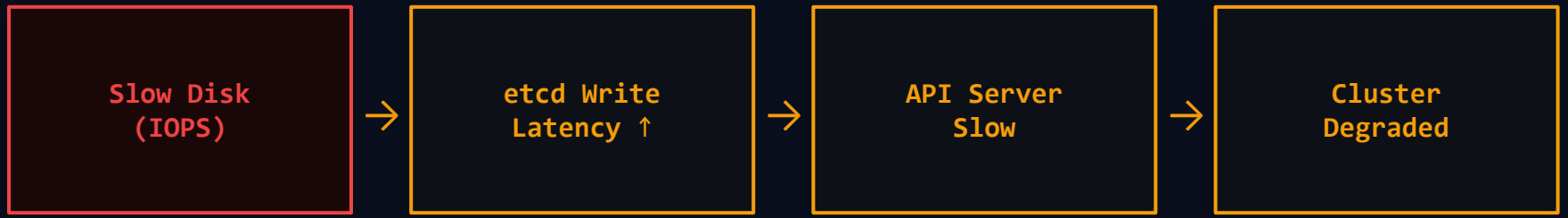
Leader election log storms

etcd lease renewals failing — component flapping at 2am

None of these point at etcd in your dashboard. That's the problem.

Failure Mode 1 — Disk Latency

etcd is disk-bound — not CPU-bound, not memory-bound.



Every write requires fsync before acknowledgment.
The entire call chain collapses to the speed of your disk.

Rule: SSD or NVMe only — NFS and gp2 EBS will quietly degrade your control plane.

Failure Modes 2 + 3

Quorum Instability

3-node cluster needs 2 to agree.
Lose quorum → cluster goes **read-only**.

Common mistakes:
- 2-node clusters (zero tolerance)
- 4-node clusters (waste)
- Members across high-latency zones

Raft needs <10ms between members.

Large Object Writes

Default limits:
- 1.5MB per value
- 2GB total DB (8GB max)

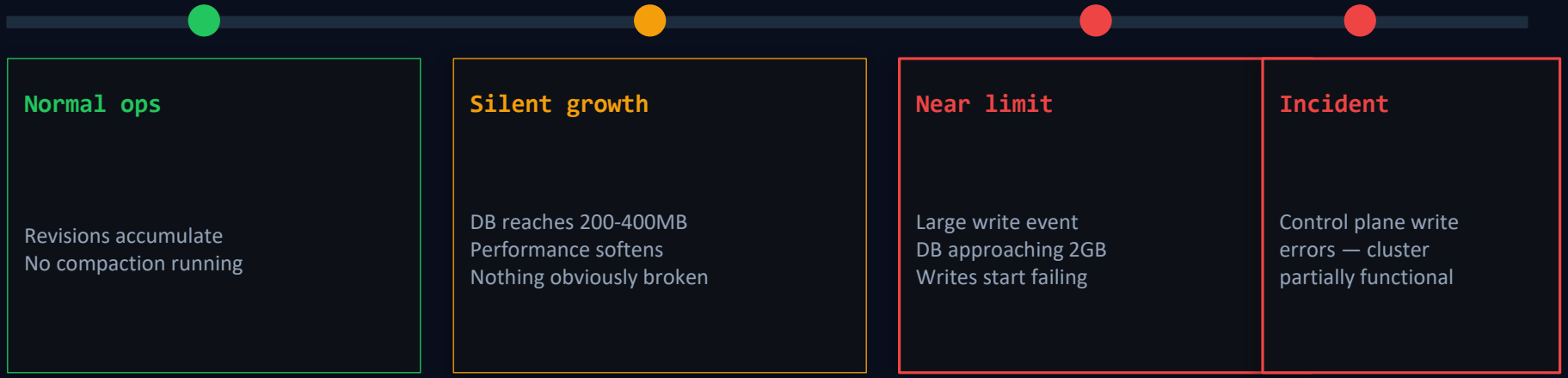
Both are reachable.

Usual offenders:
- CRDs storing runtime state
- Secrets used as blob storage
- ConfigMaps holding large files

etcd is not an object store.

Failure Mode 4 — Compaction + Fragmentation

The DB grows. Performance degrades. Nobody notices until it hurts.



Fix: automate compaction + defragmentation. Don't wait for symptoms.

One Metric. Non-Negotiable.

`etcd_disk_wal_fsync_duration_seconds`

P99 > 10ms

WARNING
Investigate now

P99 > 25ms

PROBLEM
Control plane at risk

Also watch:

- `etcd_server_leader_changes_seen_total`
- `etcd_mvcc_db_total_size_in_bytes`
- `etcd_mvcc_db_total_size_in_use_in_bytes`
- `etcd_server_slow_apply_total`

If you're only watching CPU and memory, you are not monitoring etcd.

The Rules

✓ DO

- ✓ Dedicated SSD/NVMe for etcd data directories
- ✓ 3 or 5 members — always odd, never 2 or 4
- ✓ Monitor fsync latency as primary health signal
- ✓ Automate compaction + defragmentation
- ✓ Snapshot etcd like a production DB backup
- ✓ Keep members in same region, low latency

✗ DON'T

- ✗ Co-locate etcd with noisy I/O workloads
- ✗ Store large payloads in ConfigMaps or Secrets
- ✗ Ignore fragmentation growth
- ✗ Stretch members across high-latency zones
- ✗ Assume managed etcd needs no visibility
- ✗ Treat etcd as a transparent detail

Architect's Verdict

If etcd is slow → **Kubernetes lies to you.**

If etcd is unavailable → **Kubernetes stops.**

If etcd is corrupted → **Recovery is a rebuild, not a restart.**

Treat etcd like the database it is.
It's the most important one in your cluster.

Full post + signal table: rack2cloud.com/etcd-kubernetes-database