



KubeCon



CloudNativeCon

Europe 2026

#KubeCon #CloudNativeCon

{ } antithesis

Keeping the Cloud Afloat

with Deterministic Simulation Testing



What we will cover

1

History of 
etcd's correctness

2

 antithesis

3

 + 

4

Discovering a
four-year old
“stale reads” bug

History of etcd's correctness

etcd: The Source of Truth

Role:

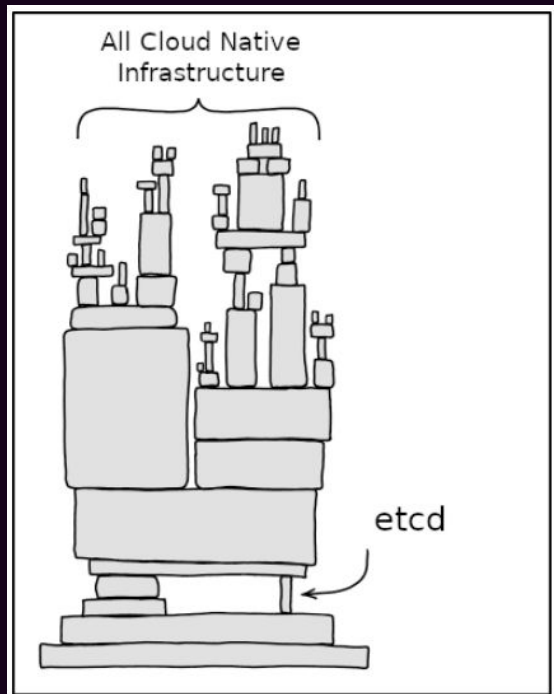
The source of truth for Kubernetes and critical cloud-native infrastructure

Weight of Responsibility:

Reliability is the number one property

Challenge:

Maintaining consistency guarantees (linearizability) in a complex, 12-year-old distributed system



Original image credited to xkcd.com/2347/, alterations by Josh Berkus.

The Crisis & The First Step



The v3.5 Crisis:

Maintainer rotation led to lost unwritten expertise. Code review missed bugs that caused inconsistent data between nodes.



Robustness Testing:

We responded with a custom testing framework inspired by Jepsen.



Model-Based Validation:

We validated etcd behavior using a simplified model of a map and codified watch guarantees.



Result:

Reproduced all v3.5 data inconsistencies and found bugs in earlier versions.

The Expertise Bottleneck



Verifying Correctness:

Validating distributed system guarantees is powerful but NP-hard problem.



Sharing Knowledge:

Bugs were so subtle that debugging was limited to a few experienced maintainers.



Ensuring Resilience:

Maintaining custom fault-injection tools (network, disk) was costly and required specialized expertise the community lacked.

{:} antithesis

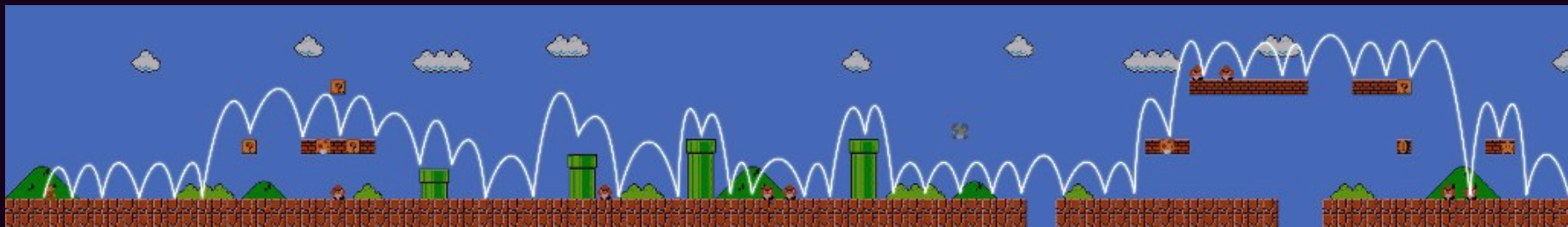
Enter Antithesis

☐ Autonomous search

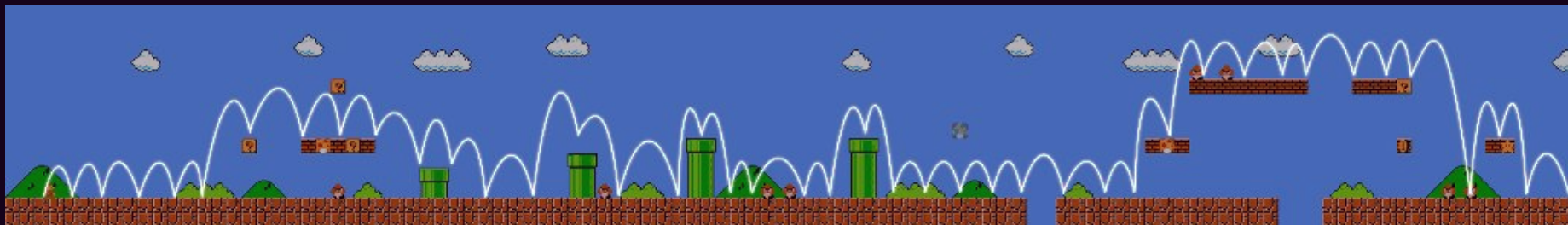
☐ Determinism



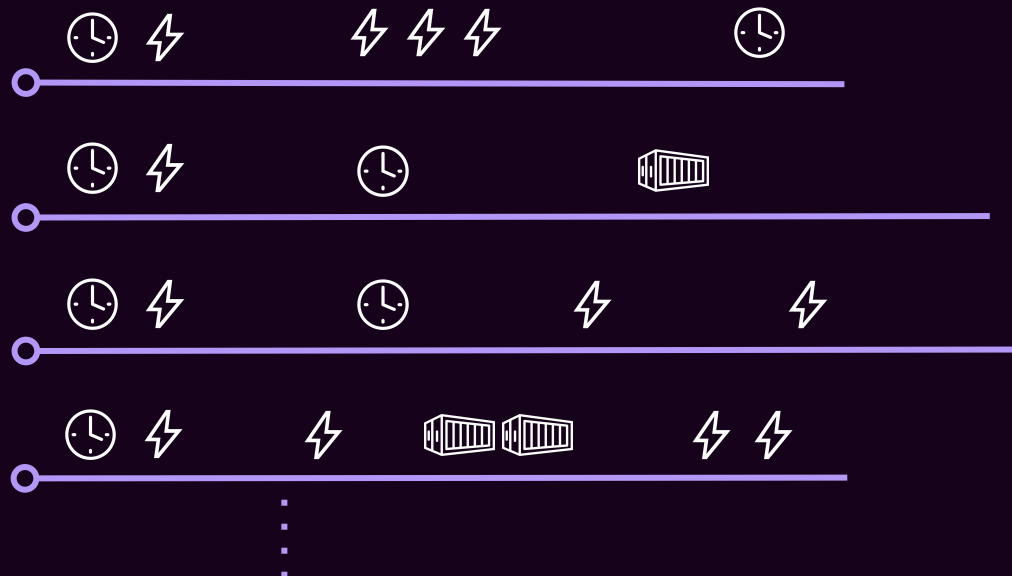
Autonomous Search



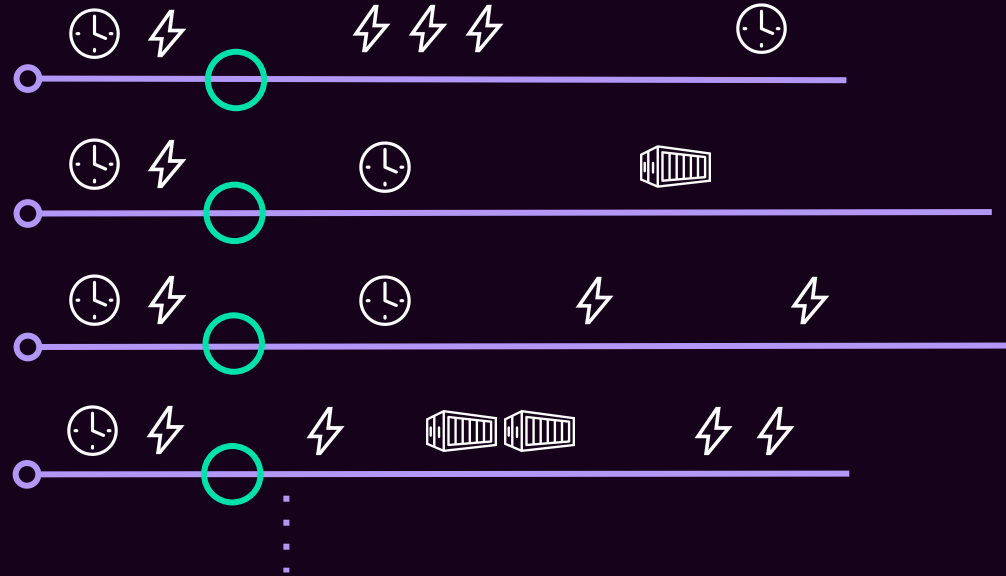
Autonomous Search



The Multiverse

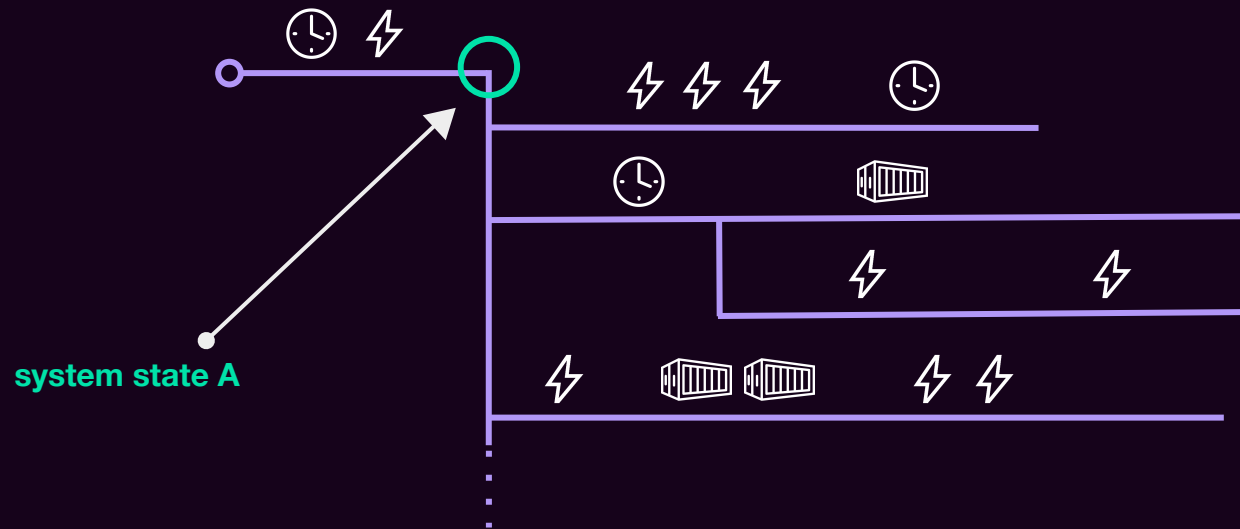


The Multiverse



system state A

The Multiverse



Enter Antithesis

☐ Autonomous search

☐ Determinism





Antithesis testing etcd



Setup:

Running a 3-node etcd cluster within a docker-compose.



Validation:

Almost “lift and shift” of the original robustness tests.



Guided exploration

Used robustness test markers of interesting etcd code locations to guide Antithesis exploration



Targeted Faults

Network partitions, thread pauses, process kills, and CPU throttling.

Discovering a four-year
old “stale reads” bug

The Stale Read Bug

The Finding:

A client received a stale read response (an old revision) that didn't include previous commits.

Hidden for 4 Years:

Stayed hidden across since v3.5.0

The Final Boss:

This is the last issue tracing back to the v3.5 release.



Finding the hardest bug

EXTREMELY RARE

Antithesis found a new set of inputs once every
~450h of real-world time.
(18.75 days)

Antithesis vs the real world?

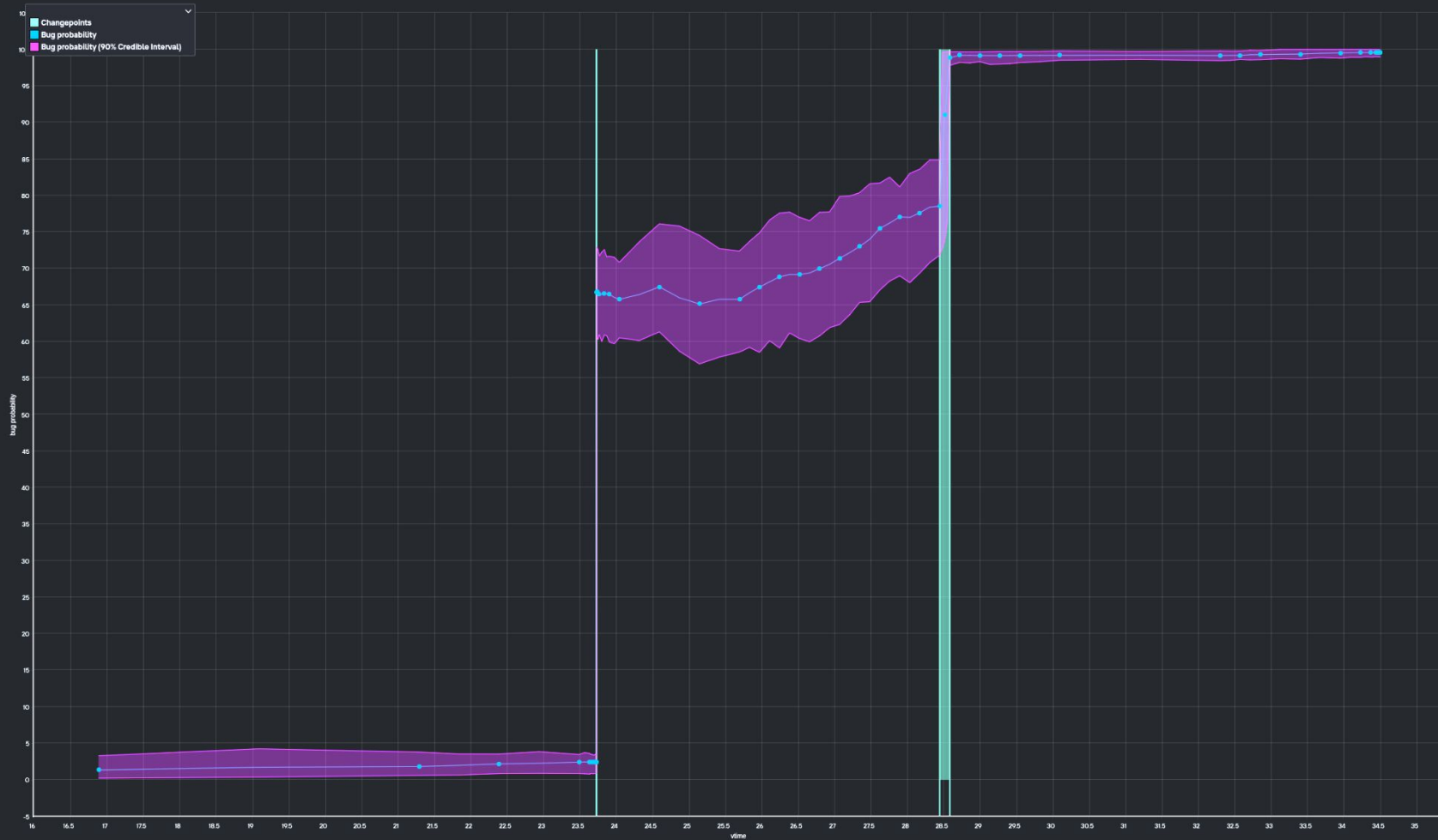


Finding the hardest bug

How do we find it
more often?

▼ Bug Probability

Shows the probability of the bug occurring in the original bug timeline



Finding the hardest bug

Debug Experiment + Event Correlation

Linearization bug is highly correlated with etcd server container hangs

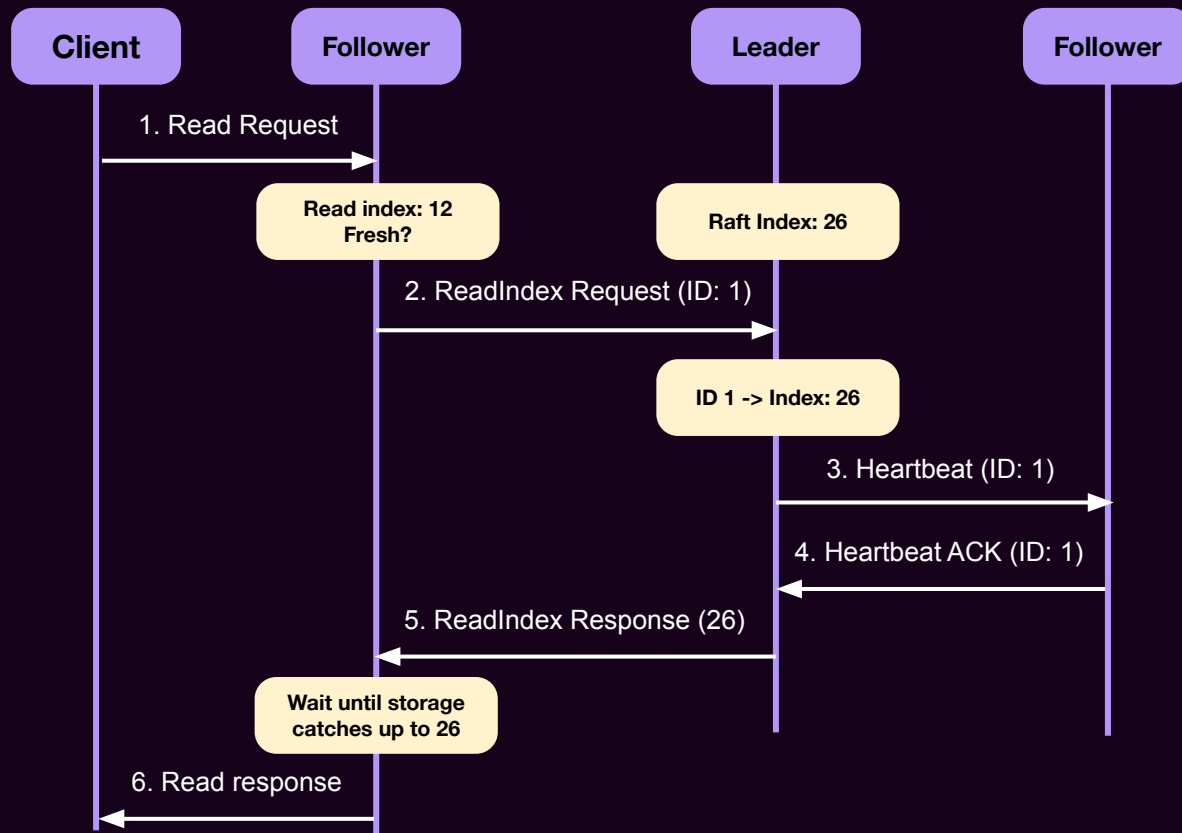
Tuning Antithesis

Update the settings to include container hangs at a much higher rate

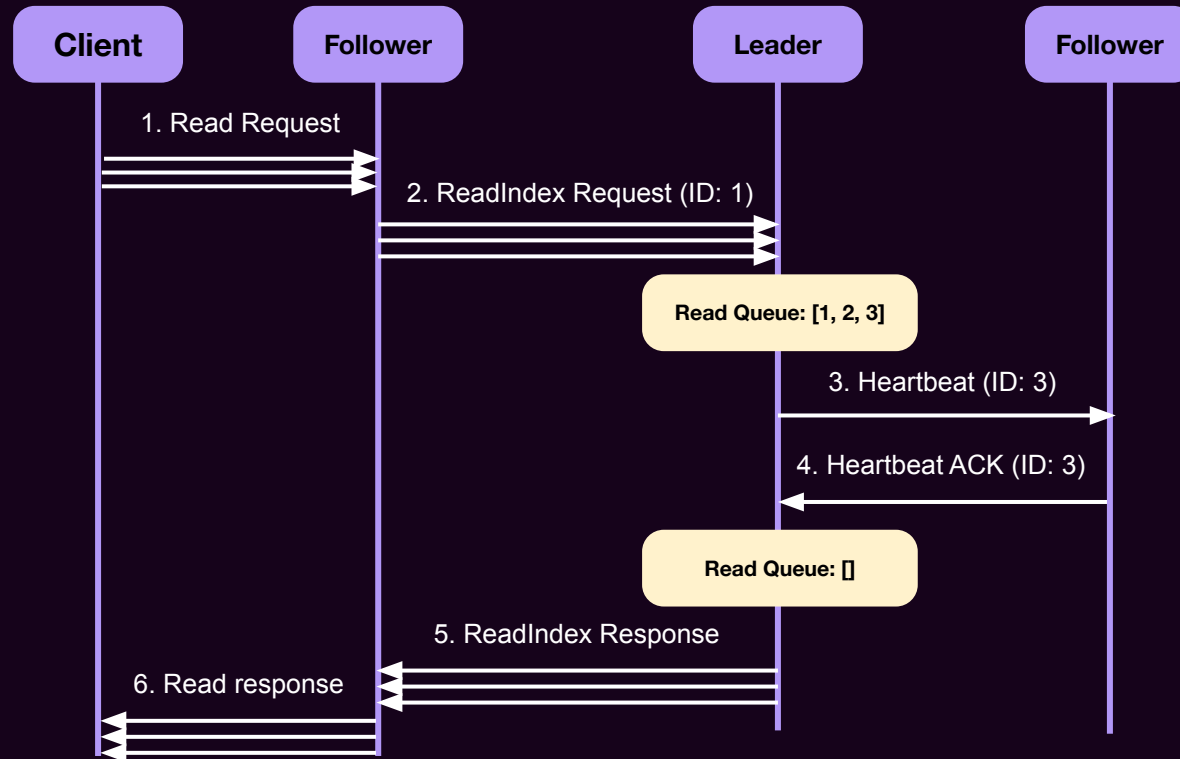
Result?

Found a new set of inputs to trip the bug once every 20 hours (20x better)

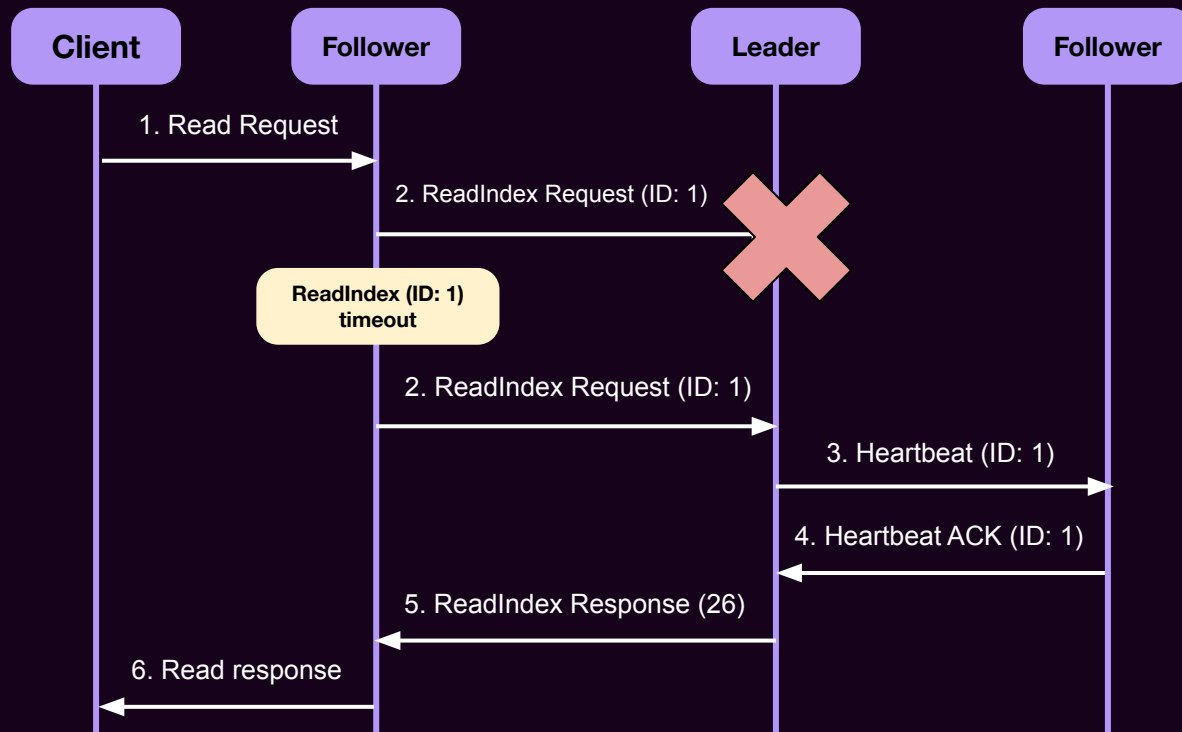
How Normal Reads Work



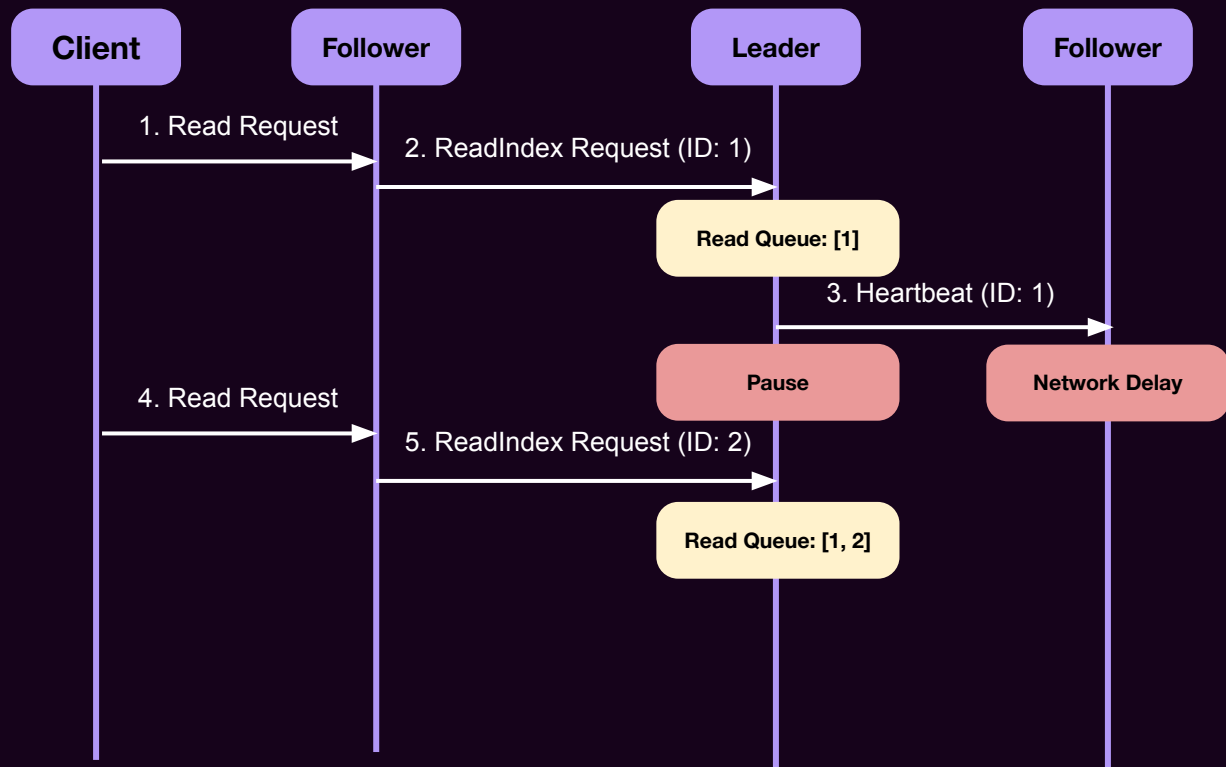
Processing Multiple Reads (The Queue)



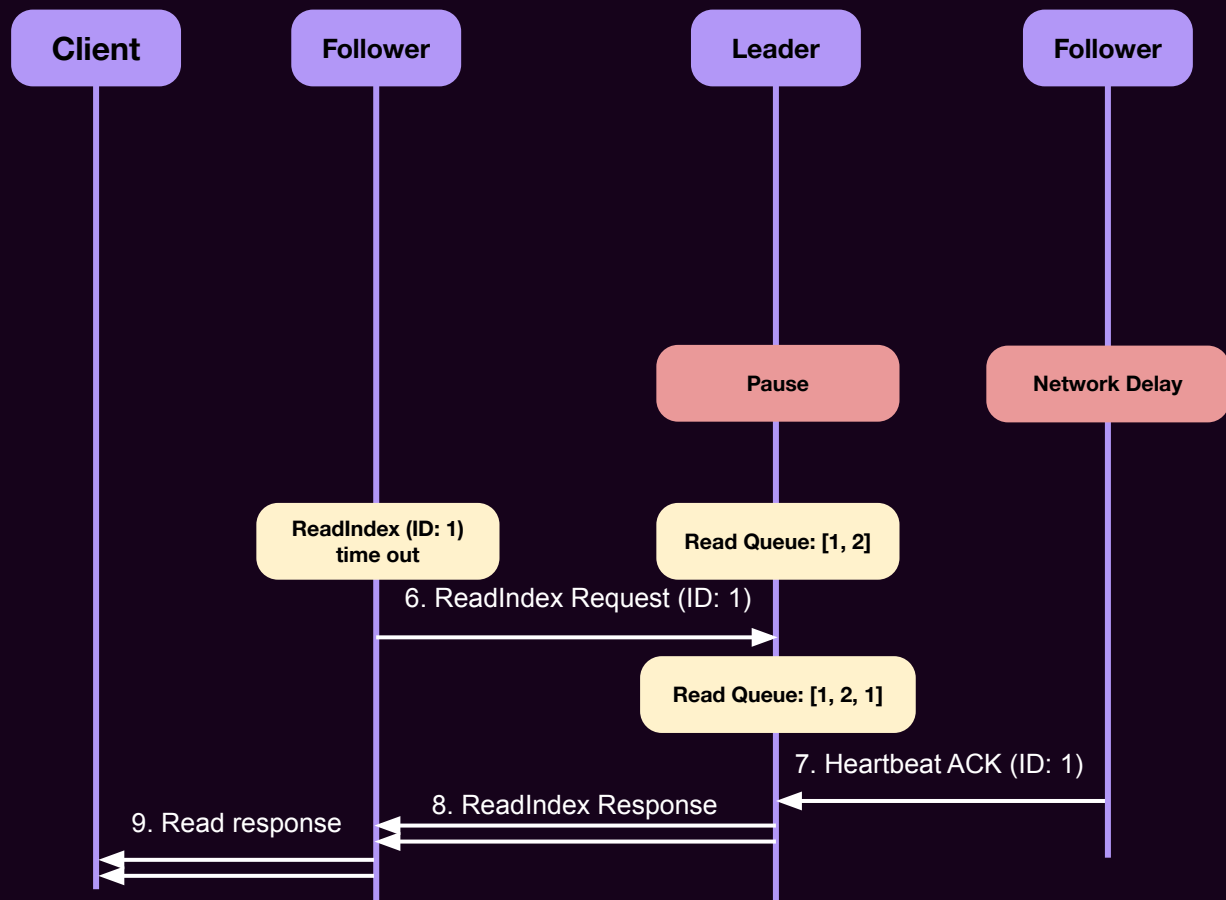
ReadIndex retries - Etcd 3.5 feature



Anatomy of the Stale Read: The Setup



Anatomy of the Stale Read: The Reordering



Anatomy of the Stale Read Bug (#20418)

The Trigger:

A specific combination of a node pause and a network slowdown.

The Readering:

A paused leader queued a new read request, but falsely confirmed it using delayed heartbeat responses from an older request.

The Fix:

Prevented the reuse of the same ReadIndex ID during retries.

The Result:

The node served stale data, breaking Linearizability, despite a new leader already committing newer transactions.

Conclusion: Freeing the Maintainers



Velocity:

Trust in testing allows us to tackle risky features (e.g., 2-node clusters via witness).



Democratization:

Perfectly reproducible reports allow any contributor to fix complex distributed system bugs.



The Future:

Autonomous testing is available to all incubating and graduated CNCF projects.



KubeCon



CloudNativeCon

Europe 2026



KubeCon



CloudNativeCon

Europe 2026

{:} antithesis

Thank you!

Booth #888 / 889

