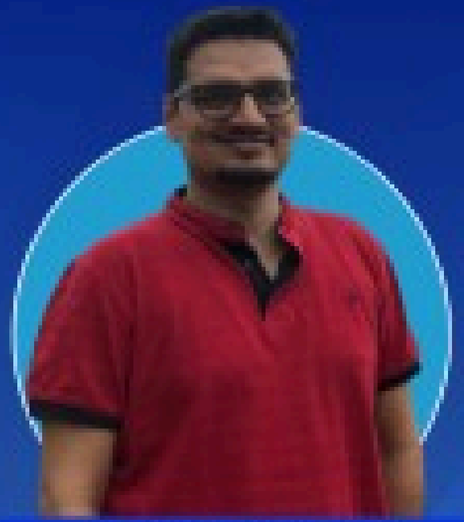
OSA CON 25

CLICKHOUSE® CHRONICLES: REAL-WORLD WAR ROOMS WITH HUMAN AND AI AGENTS



Shivji Kumar Jha

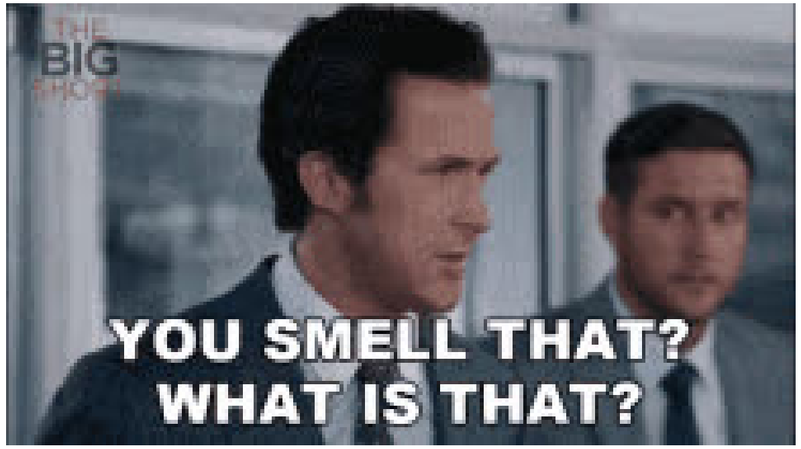
Staff Engineer @ Nutanix
Data & Cloud geek

Anurag Pandey

Building Ai Copilot
for Databases

NOVEMBER 4 & 5, 2025
ONLINE EVENT

AI + L1-2 Clickhouse production problems



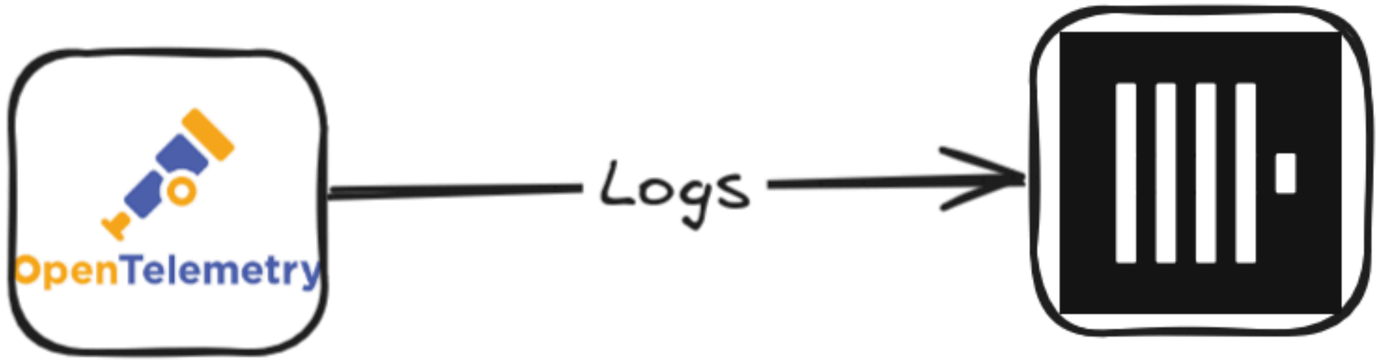
incerto

1st AI Problem

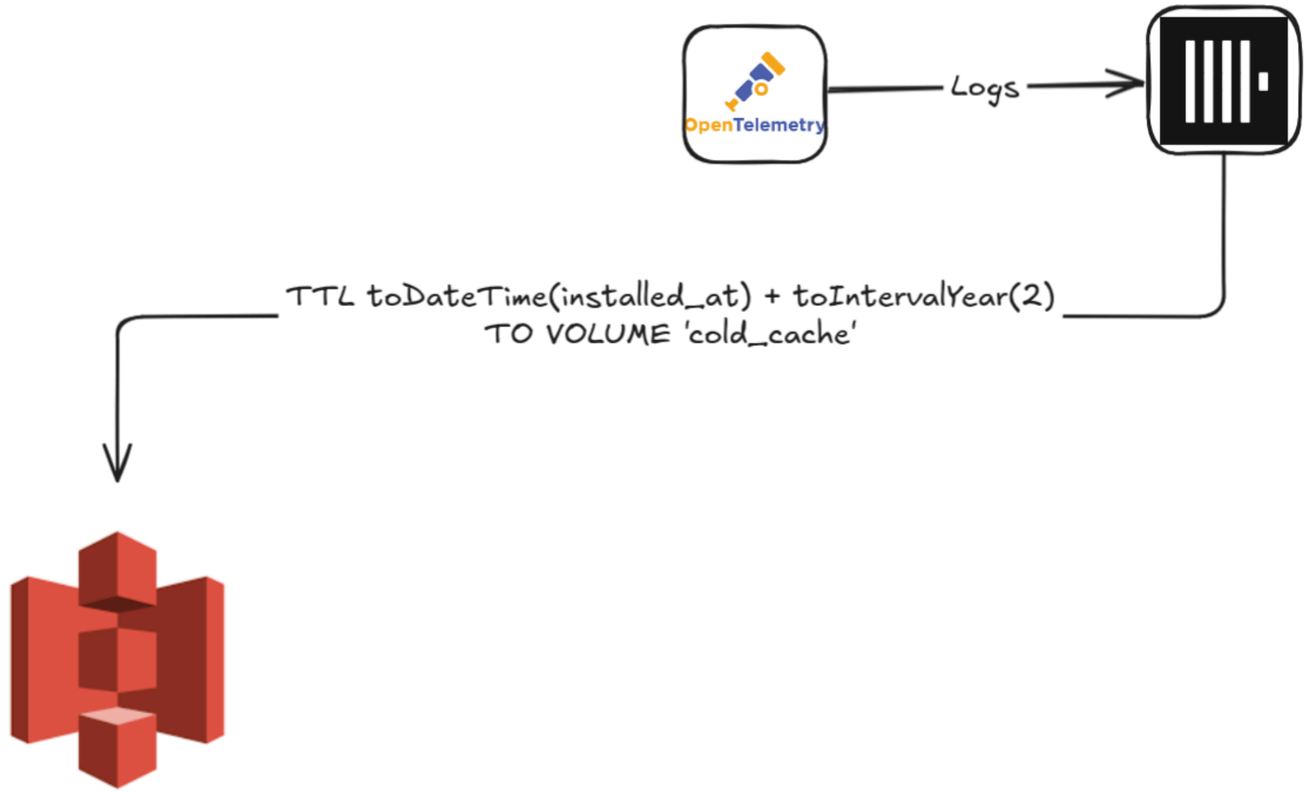
S3 spikes 100x in cost

incerto

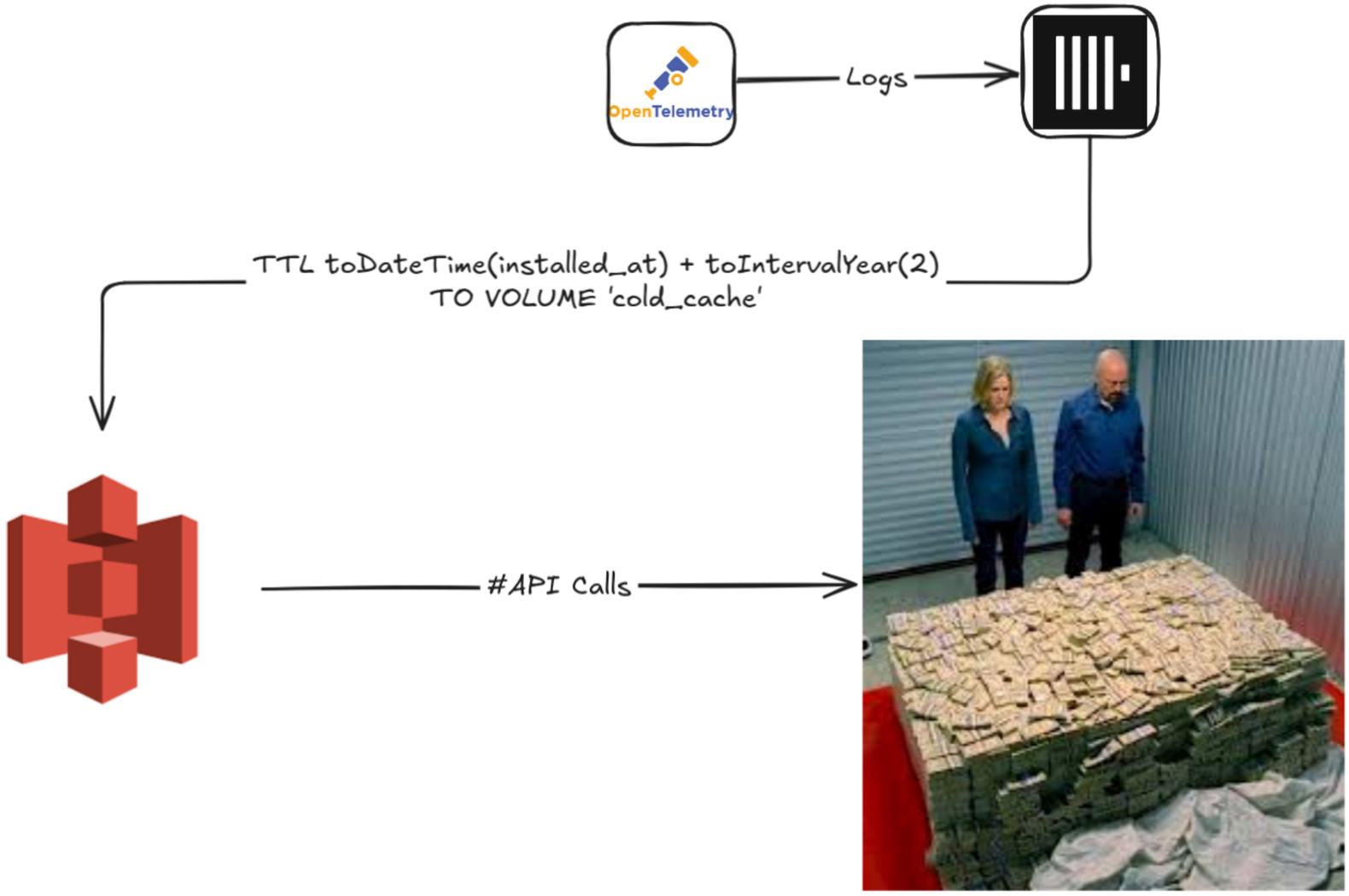
Otel Logs being sent to Clickhouse



2yrs old logs moved to s3



API calls = Money

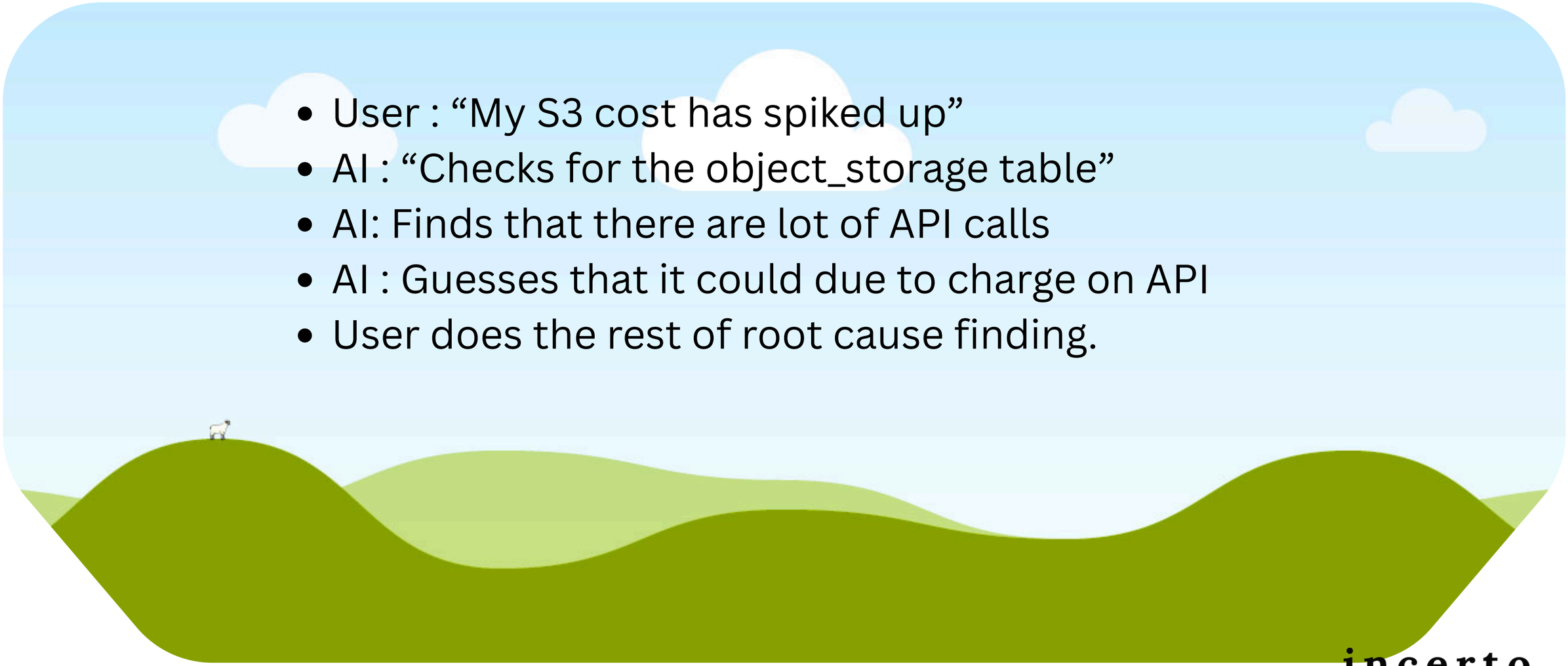


Root Cause ?

1970-01-01

How did AI help ?

- User : “My S3 cost has spiked up”
- AI : “Checks for the object_storage table”
- AI: Finds that there are lot of API calls
- AI : Guesses that it could due to charge on API
- User does the rest of root cause finding.



User queries failing

2nd AI Problem

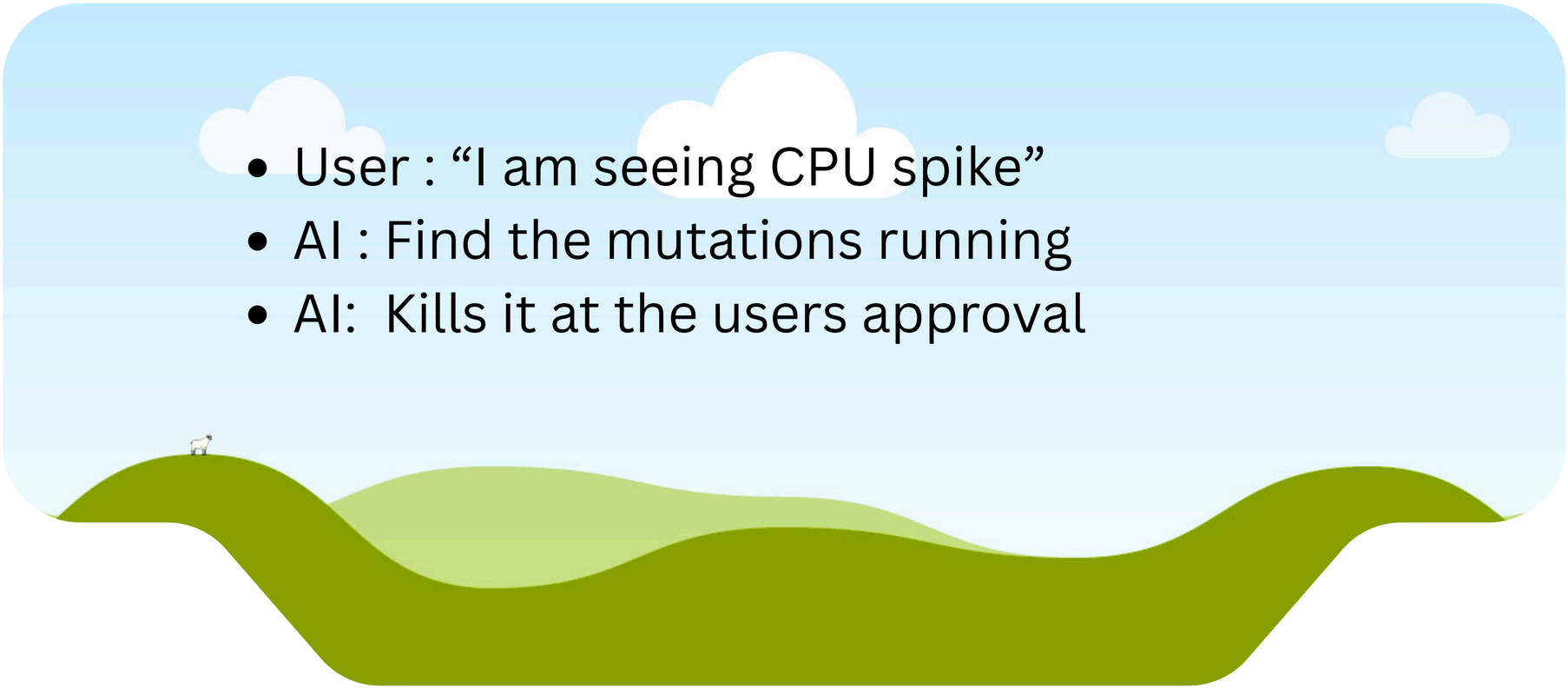
1. **QPS was the same**
2. **All Worker UP**
3. **No unusual queries**

Root Cause ?

Huge Delete Mutations

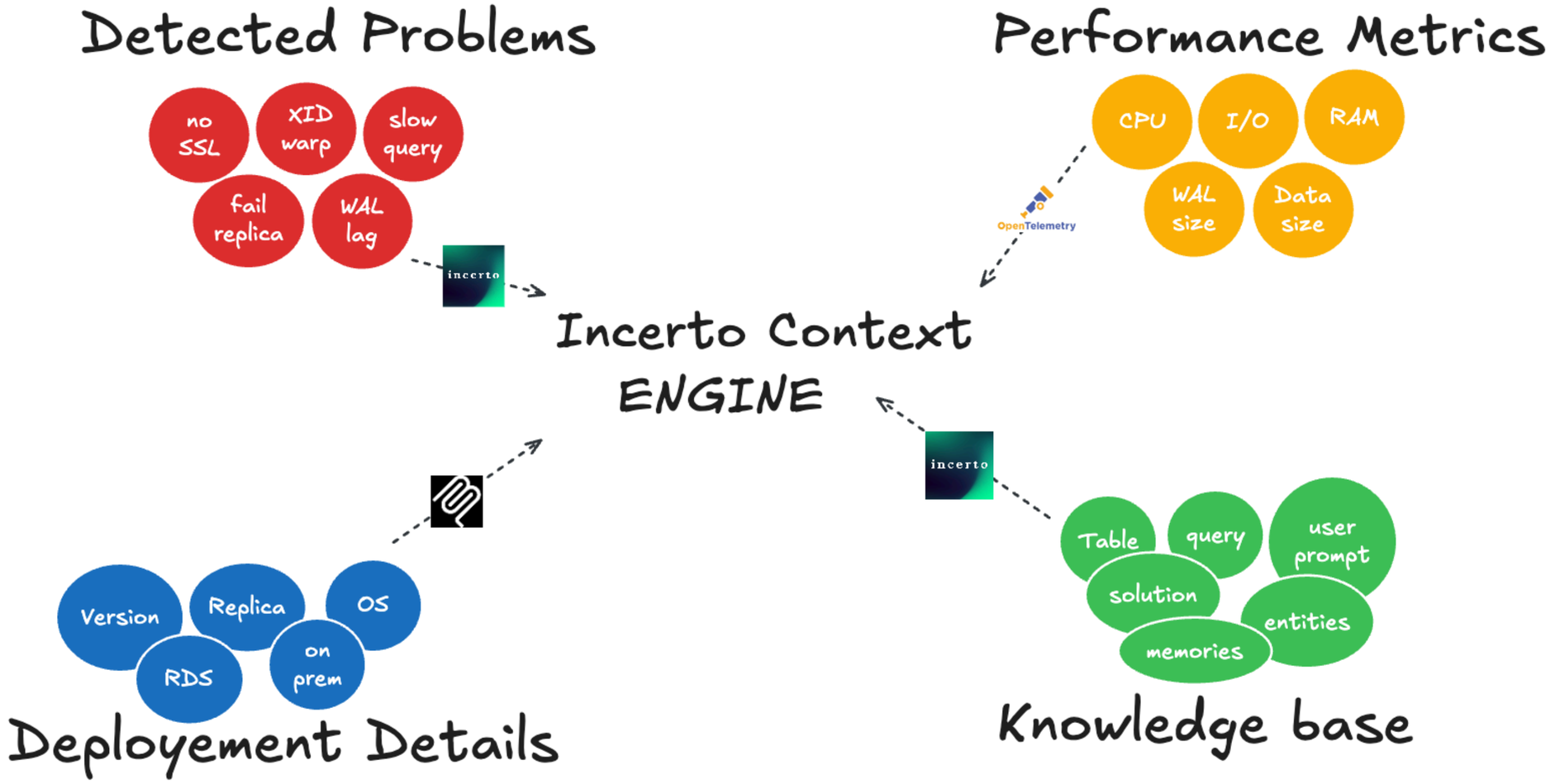
How did AI help ?

- User : “I am seeing CPU spike”
- AI : Find the mutations running
- AI: Kills it at the users approval



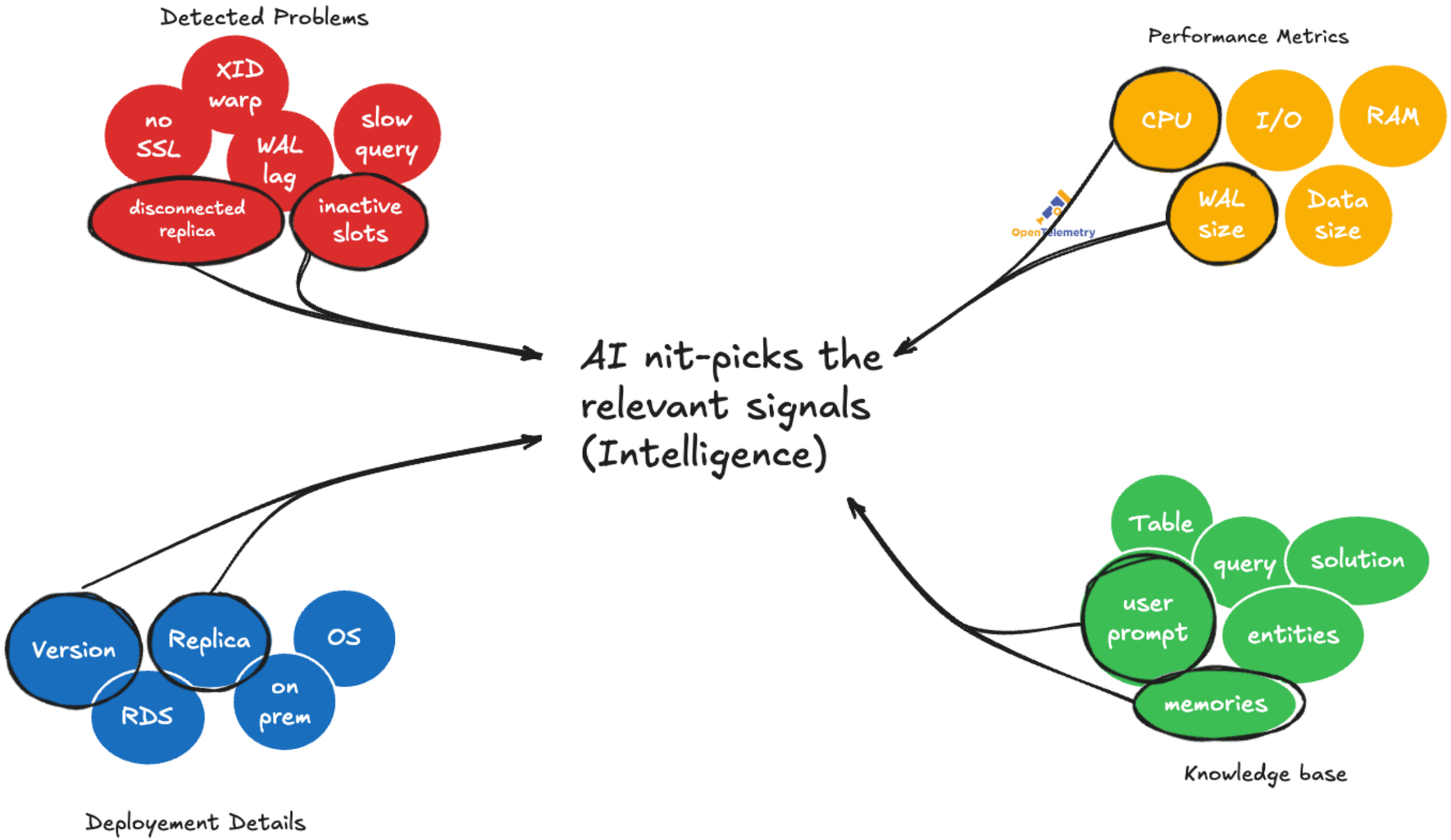
How do design AI Debugger

Context Engine

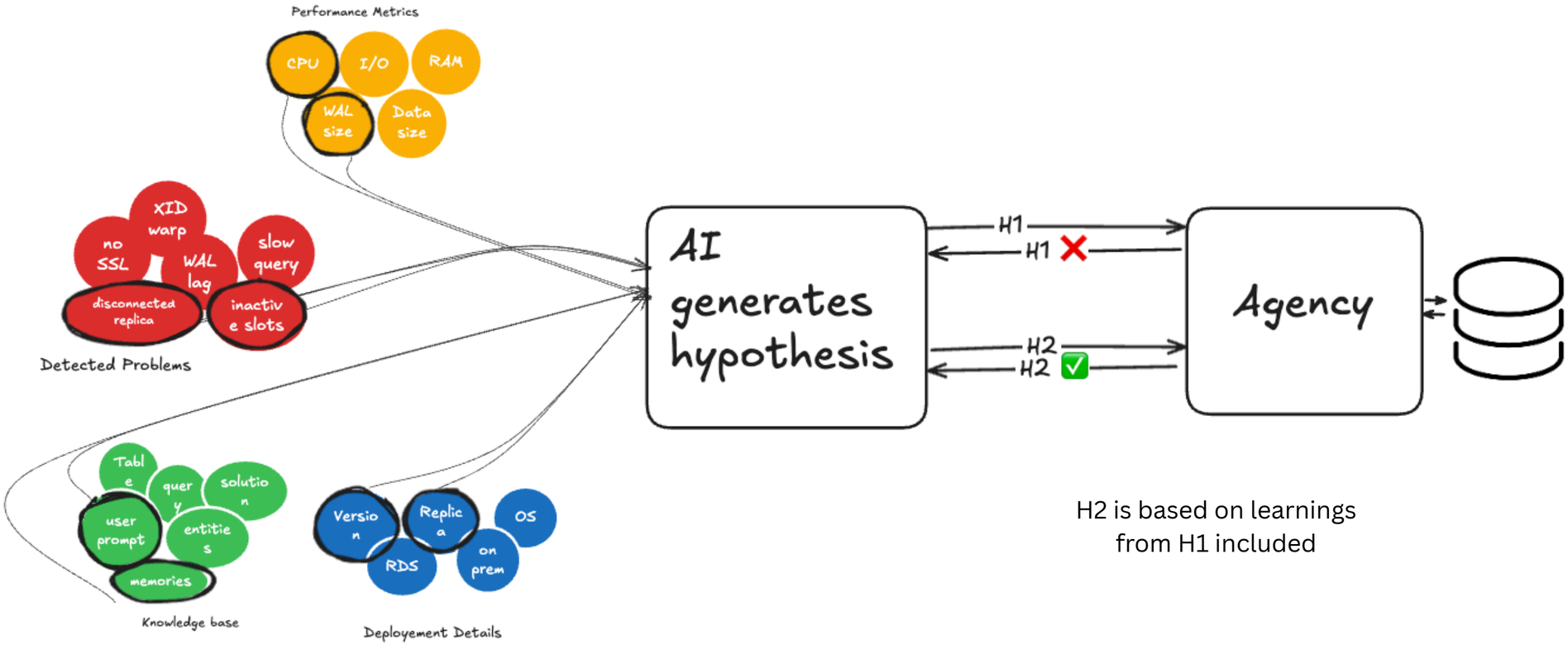


incerto

AI picks up the right signals



AI generates & verifies hypothesis



H2 is based on learnings from H1 included

incerto

Don’t trust anyone who tells you

“AI can do everything!”

Lets look at some human actions…

Distributed systems is hard!

Resiliency depends on
how well you
test **Failure scenarios**

Some Chaos tests:

1. Introduce OS Kernel panic
2. Artificial Network latency
3. Stop the io channels abruptly
4. Kill pod, VM, power off abruptly

Human war story - 1

Clickhouse server failed to come up

Code: 722. DB::Exception: Waited job failed: Code: 696. DB::Exception: Load job 'startup table db1.table1' -> Code: 695. DB::Exception: Load job 'load table db1.table1' failed: Code: 231. DB::Exception: Suspiciously many (119 parts, 0.00 B in total) broken parts to remove while **maximum allowed broken parts count is 100**. You can change the maximum value with merge tree setting 'max_suspicious_broken_parts' in <merge_tree> configuration section or in table settings in .sql file (don't forget to return setting back to default value): Cannot attach table `db1`.`table1` from metadata file /var/lib/clickhouse/store/402/40286535-a77d-4d49-890c-fea3bf2f2d2e/table1.sql from query

.....

(**TOO_MANY_UNEXPECTED_DATA_PARTS**),. (ASYNC_LOAD_WAIT_FAILED) (version 24.8.8.17 (official build))

max_suspicious_broken_parts

1. For the write througput of modern hardware, default value of **100 is too low**!
2. We changed it to 1OOK
3. We prefer self-healing over data loss
    a. Getting access to cluster is very hard
    b. Also, its observability data

https://clickhouse.com/docs/operations/settings/merge-tree-settings#max_suspicious_broken_parts

Human war story - 2

Queries began failing with

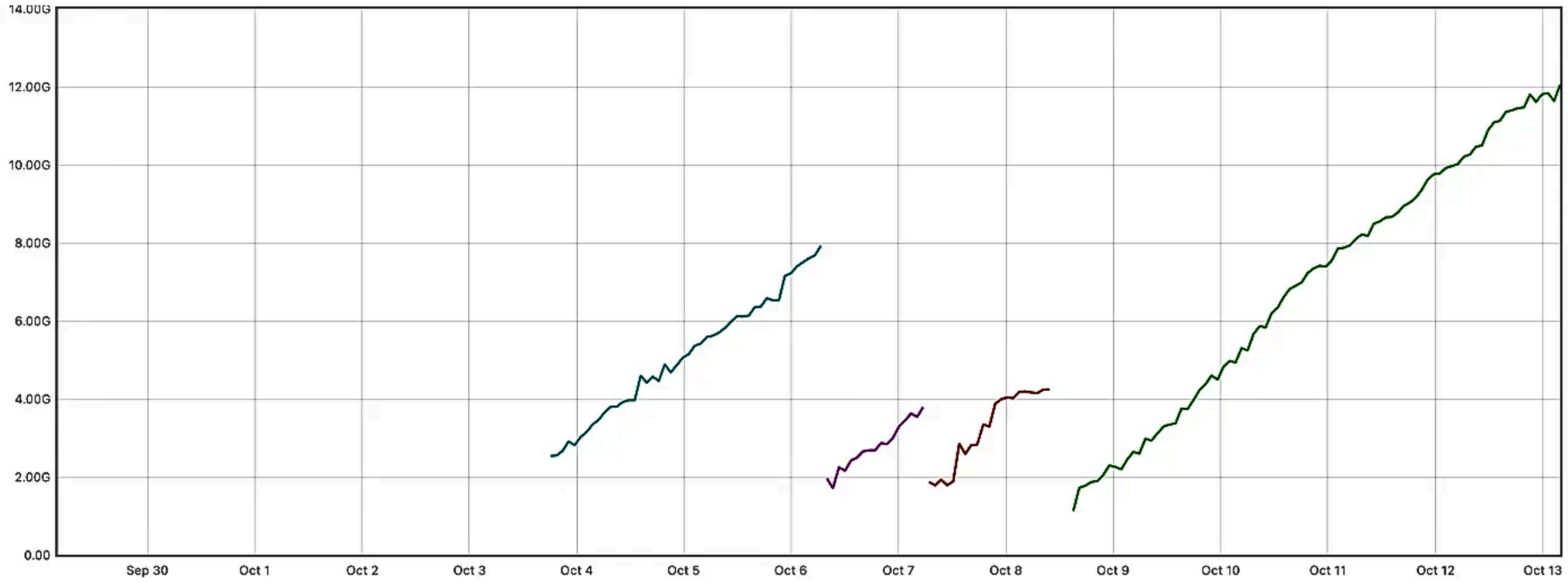
“Memory limit exceeded” errors

on ClickHouse v25.3.5.42

Every 5 to 6 days, clickhouse restarts 

thanks to the kubernetes gods 

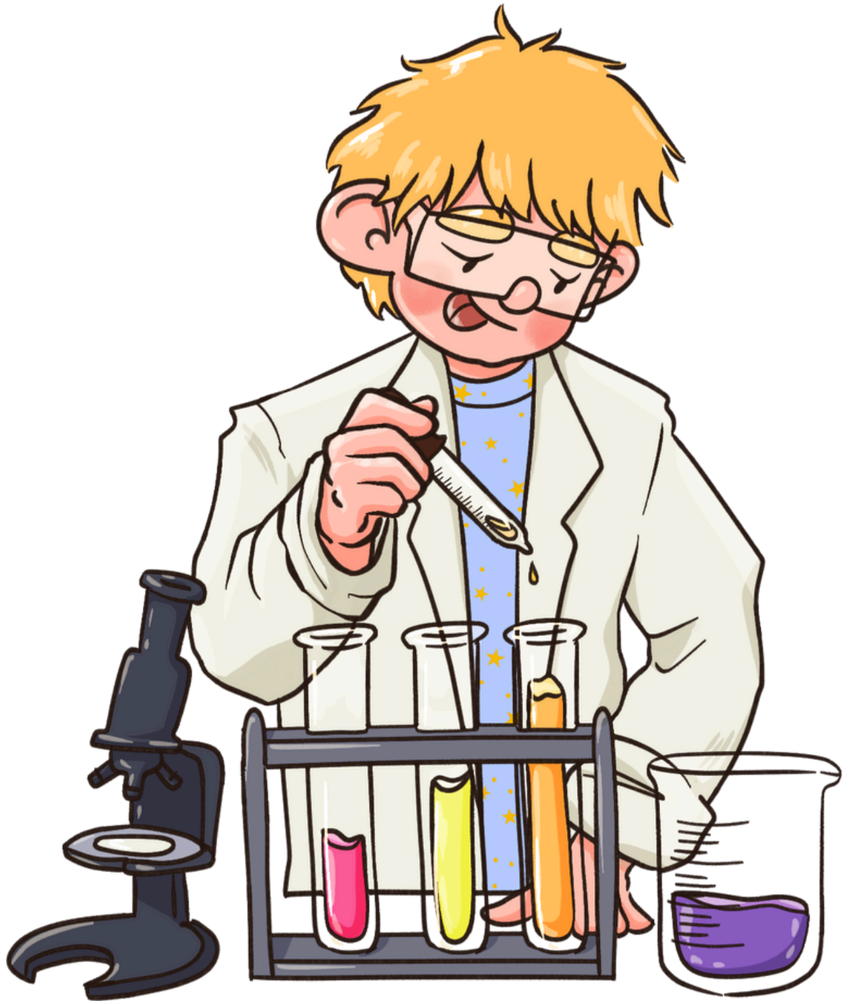
The dreaded graph no dev wants to see (ouch!)



container_memory_working_set_bytes{container=~"clickhouse-pod"}

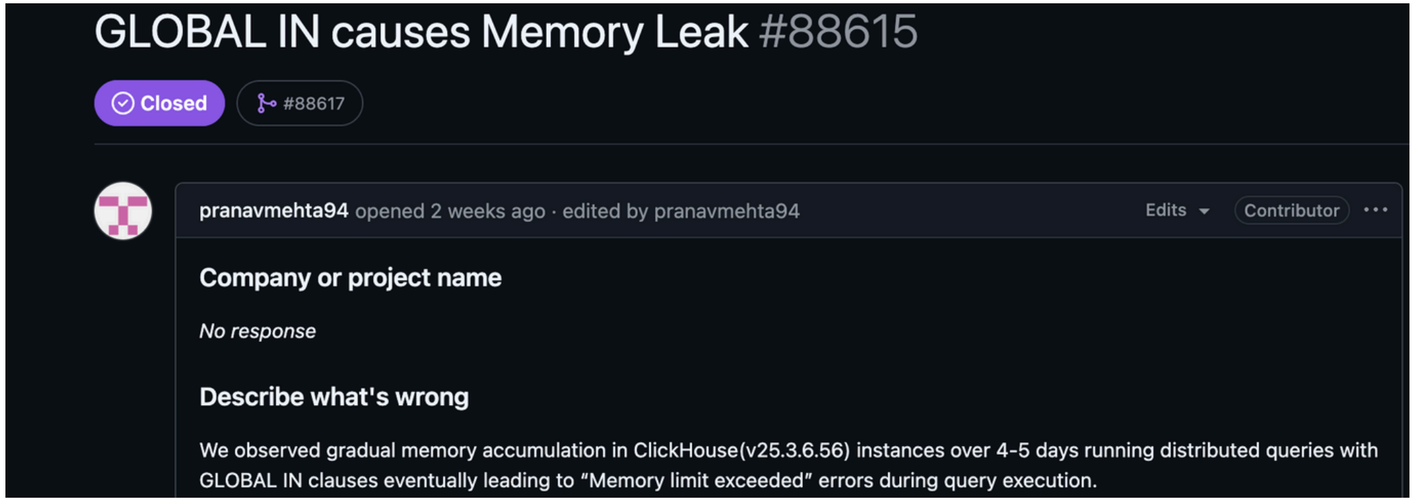
Stared with the usual suspects

1. check the query load
2. Investigate background merges & caches
3. Suspecting the Page cache
4. Nothing helped

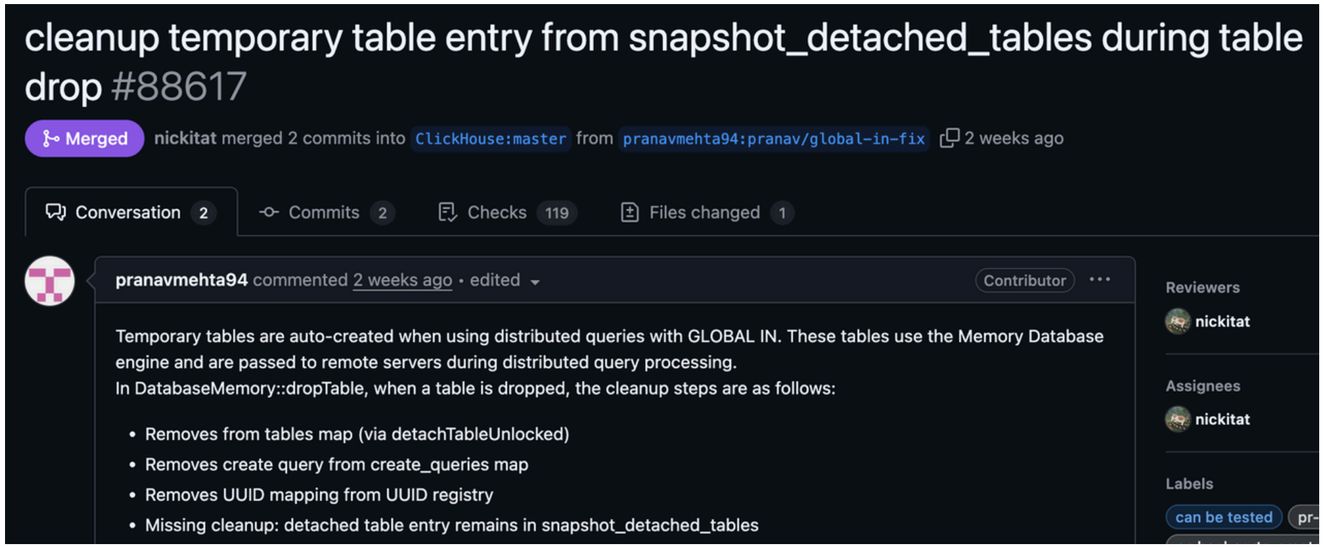


Finally, accept the worst nightmare & profile **Jemalloc** with *jeprof*

Fix server and merge upstream - thanks to the OSS gods!



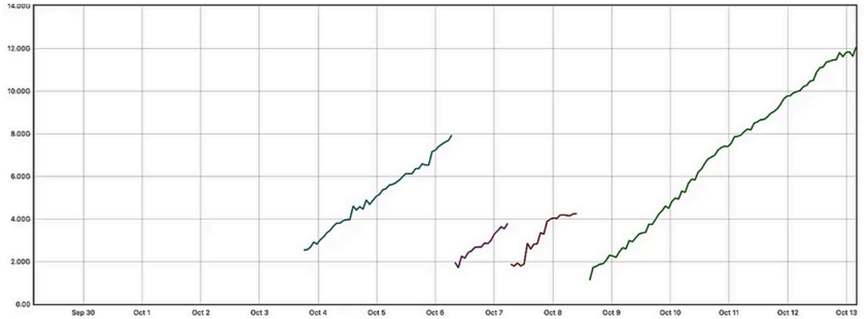
https://github.com/ClickHouse/ClickHouse/issues/88615

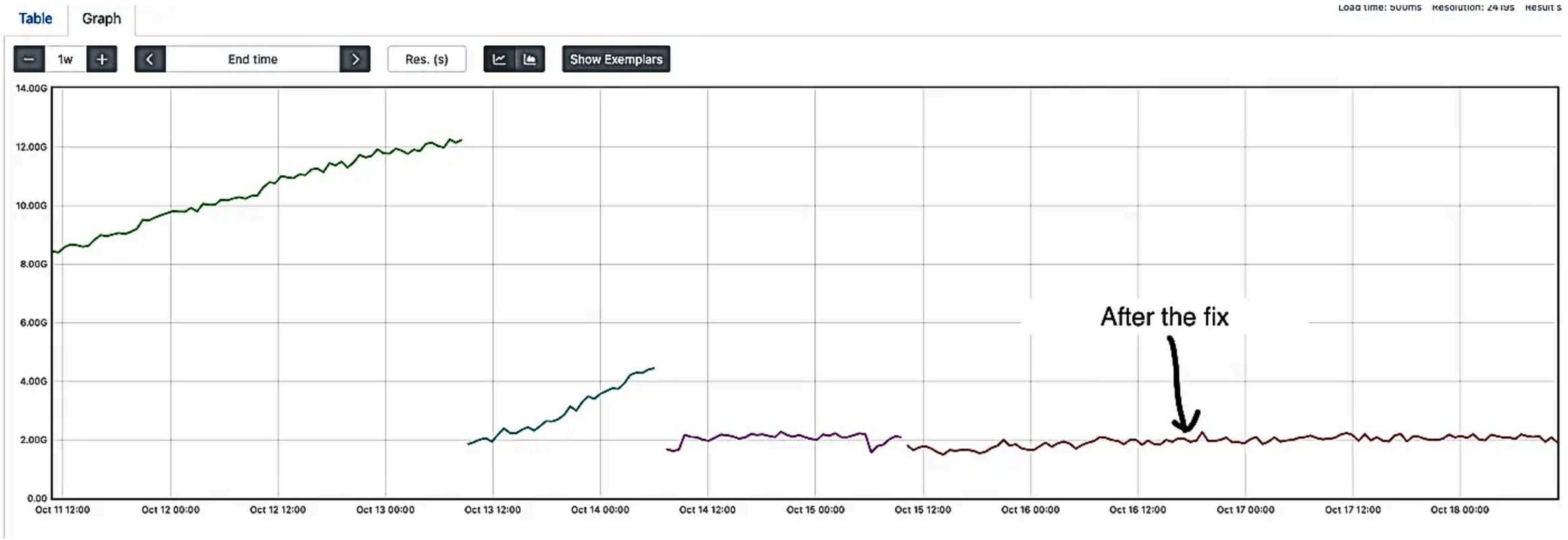


https://github.com/ClickHouse/ClickHouse/pull/88617

Blog post - https://medium.com/@pranavmehta94/a-clickhouse-memory-leak-story-768399955424

Memory leak =>





container_memory_working_set_bytes{container=~"clickhouse-pod"}

Human war story - 3

Network errors in single pod Clickhouse logs

Error type 1:

Response: {"message": "Invalid Argument: 6
:Internal Error: 5
:clickhouse: Code: 279. DB::NetException: ***All connection tries failed****. Log:*
Code: 210. DB::NetException: I/O error: ***Broken pipe****, while writing to socket (127.0.0.1:40882 -> 127.0.0.1:9000).*
*(****NETWORK_ERROR****) (version 24.3.5.46 (official build))*
*Code: 210. DB::NetException: I/O error: Broken pipe, while writing to socket (****127.0.0.1:40896 -> 127.0.0.1:9000****).*
(NETWORK_ERROR) (version 24.3.5.46 (official build))

github.com/ClickHouse/ClickHouse/issues/64954#issuecomment-2365285608

Closed **The NETWORK_ERROR occurs frequently** #64954

alexey-milovidov on Sep 22, 2024

The diagnostic in the logs is correct. The error messages are correct.

👎 4

alexey-milovidov closed this as completed on Sep 22, 2024

You have a flaky network,
go run clickhouse in a stable setup please!!

Network errors in single pod Clickhouse logs

Error type 2:

2025.03.05 04:53:13.804881 [1153] {c349a22c-7450-4f49-85d7-17e430386702} <Information> ***HedgedConnectionsFactory****: Connection failed at try №1, reason: Code: 210. DB::NetException: I/O error: Broken pipe, while writing to socket (**127.0.0.1:47536 -> 127.0.0.1:9000**). (**NETWORK_ERROR**) (version 24.8.8.17 (official build))*

How Hedged Requests Work:

1. ClickHouse starts by sending a query to one replica.
2. If the response is delayed beyond a certain threshold, it sends the same query to another replica.
3. The first response to return is used, and the other request(s) are discarded.

Network errors in single pod Clickhouse logs

Error Type 3:

stack_trace: 0. DB::Exception::Exception(DB::Exception::MessageMasked&&, int, bool) @ 0x000000000d17749b
1. DB::NetException::NetException<String&>(int, FormatStringHelperImpl<std::type_identity<String&>::type>, String&) @ 0x0000000011c35703
2. ***PoolWithFailoverBase<DB::IConnectionPool>****::getMany(unsigned long, unsigned long, unsigned long, unsigned long, bool, bool,*

Connection Pools related to Distributed query processing exhausted

Found connection configs, reviewed and updated

Clickhouse config	Default	Updated values	Description
tcp_keep_alive_timeout	290s	1s	The time in seconds the connection needs to remain idle before TCP starts sending keepalive probes
use_hedged_requests	1	0	Use hedged requests for distributed queries
connections_with_failover_max_tries	3	10	The maximum number of connection attempts with each replica for the Distributed table engine.
distributed_connections_pool_size	1024	4096	The maximum number of connection attempts with each replica for the Distributed table engine.
max_distributed_connections	1024	64	The maximum number of connections for distributed processing of one query (should be greater than max_threads).

Caution: Values depend on the nature of **your** environment

Questions?

Thank you!