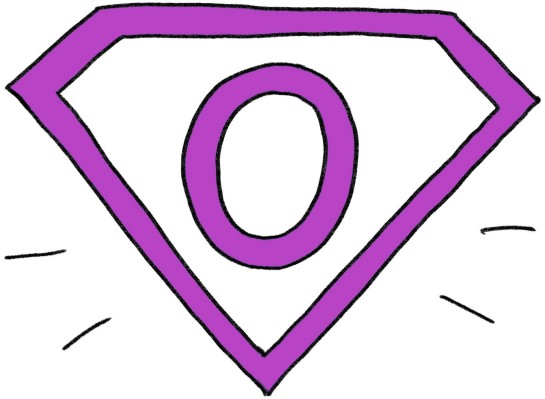
Managing to your SLO (amidst chaos!)

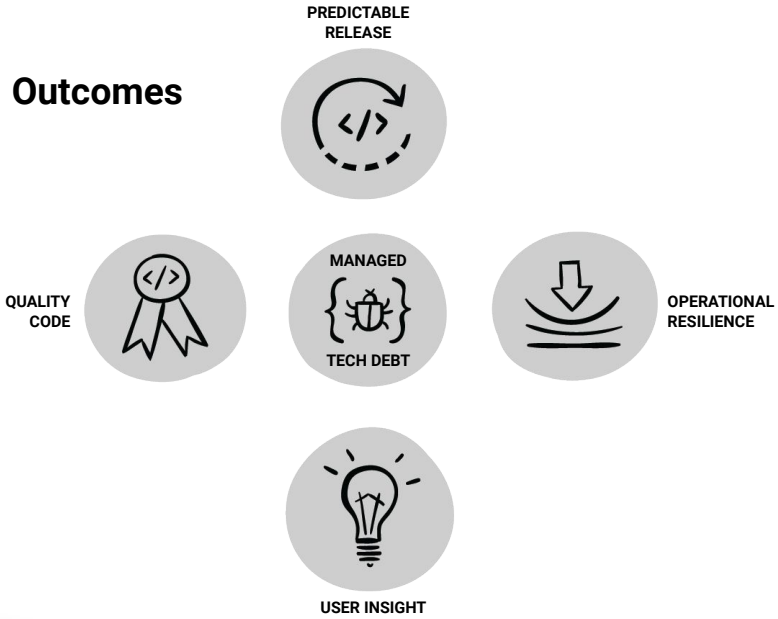
Liz Fong-Jones
Field CTO, Honeycomb.io
YOW! 2022



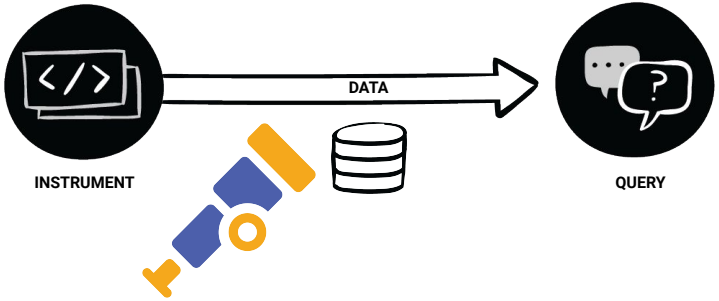
Observability is evolving quickly.

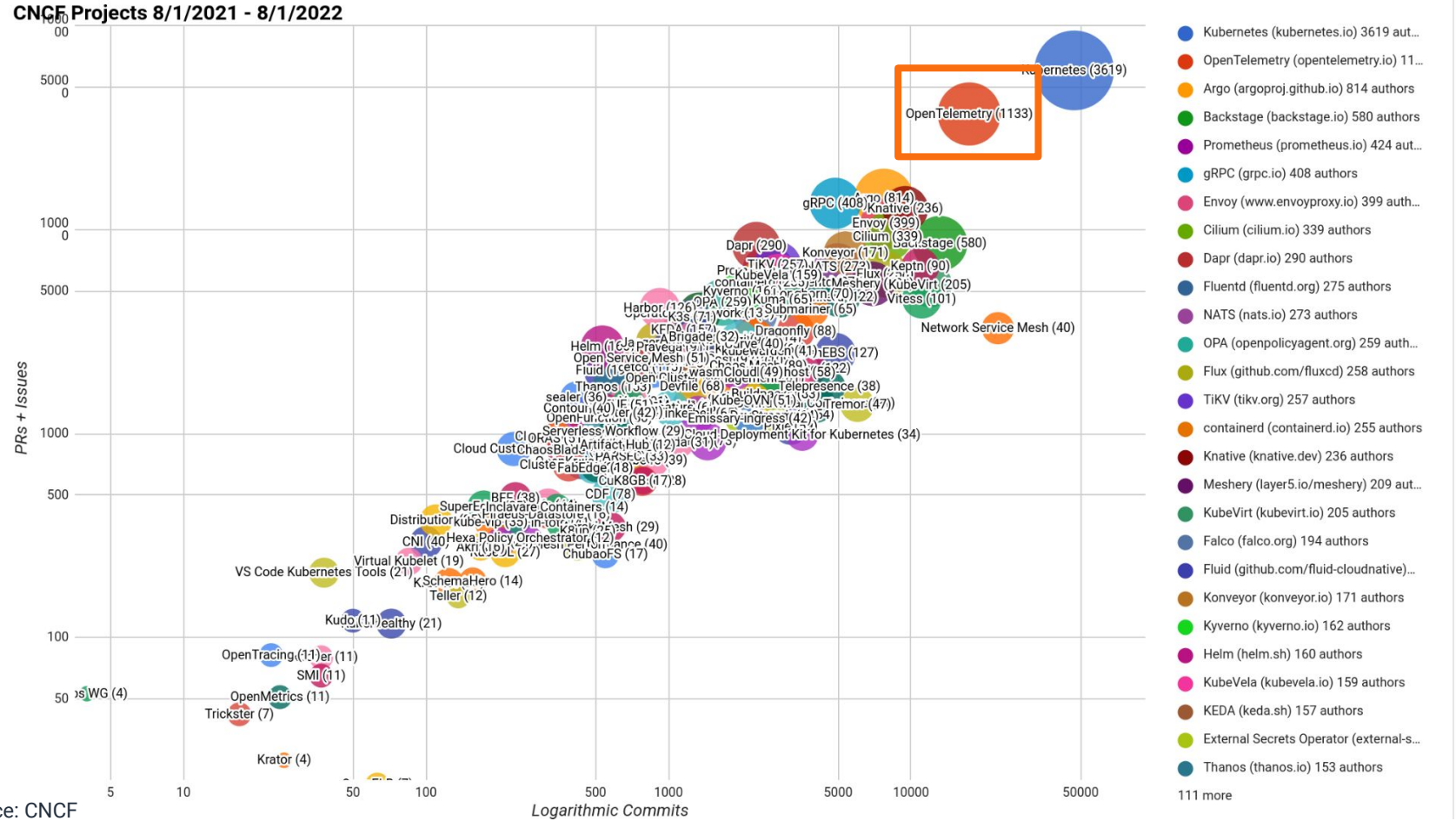
And the problem space is complex.

Outcomes

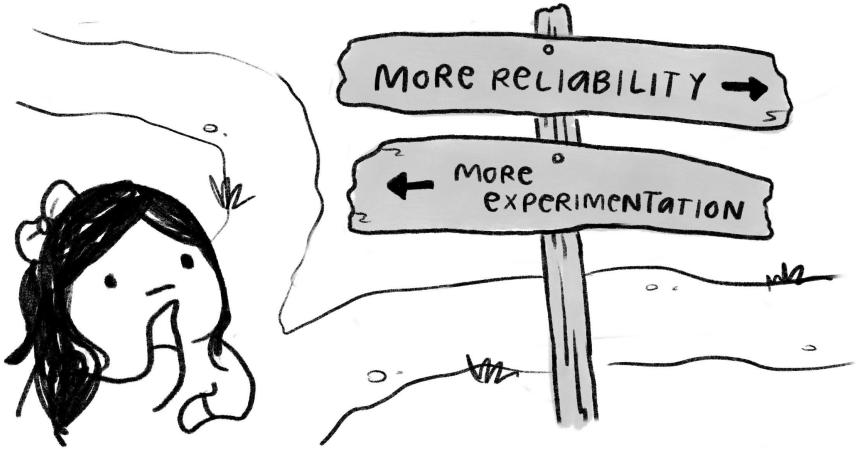


Actions



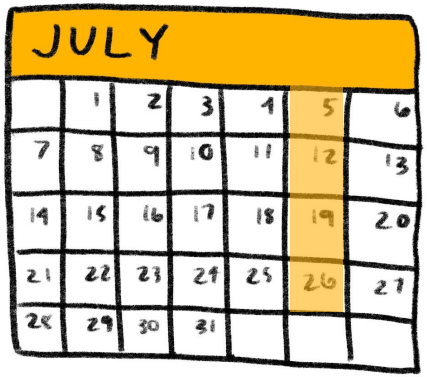


Source: CNCF



Practitioners need velocity, reliability, & scalability.

A small but growing team builds Honeycomb.



We deploy with confidence.

VISUALIZE	WHERE	GROUP BY
COUNT_DISTINCT(global.build_id)	trace.parent_id does-not-exist	None; don't segment
+ ORDER BY	+ LIMIT	+ HAVING

Run Query

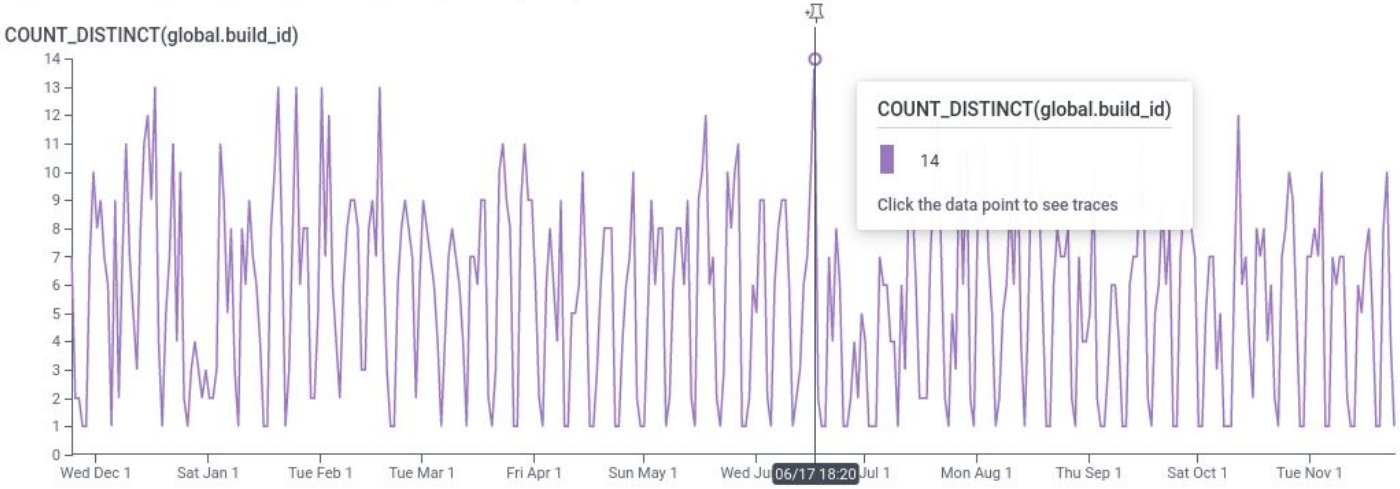
Run 2 minutes ago

Results | BubbleUp | Metrics | Traces | Raw Data

Compare to Previous time range

Graph Settings

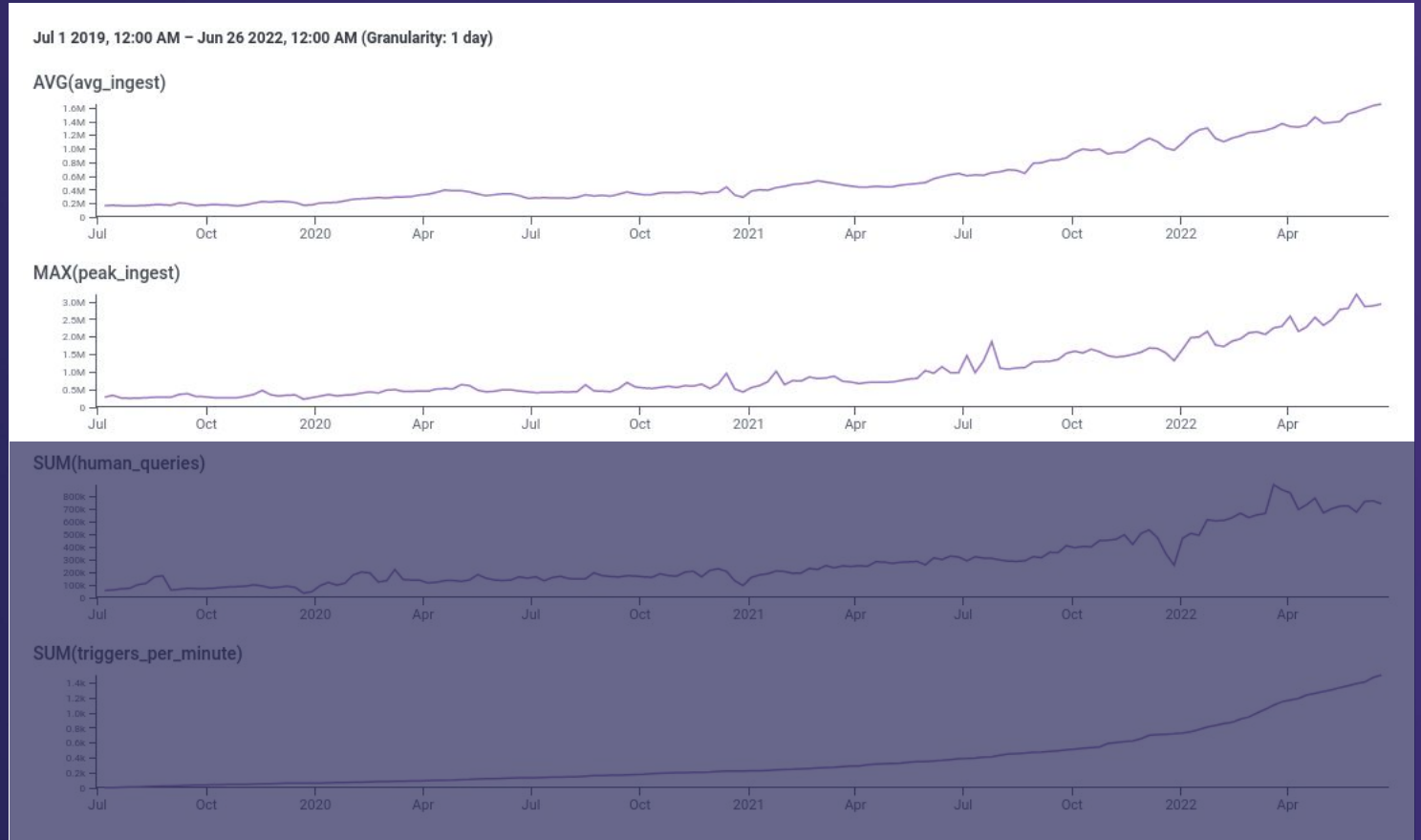
Nov 24 2021 19:13:24 – Nov 24 2022 19:13:24 UTC+11:00 (Granularity: 1 day)



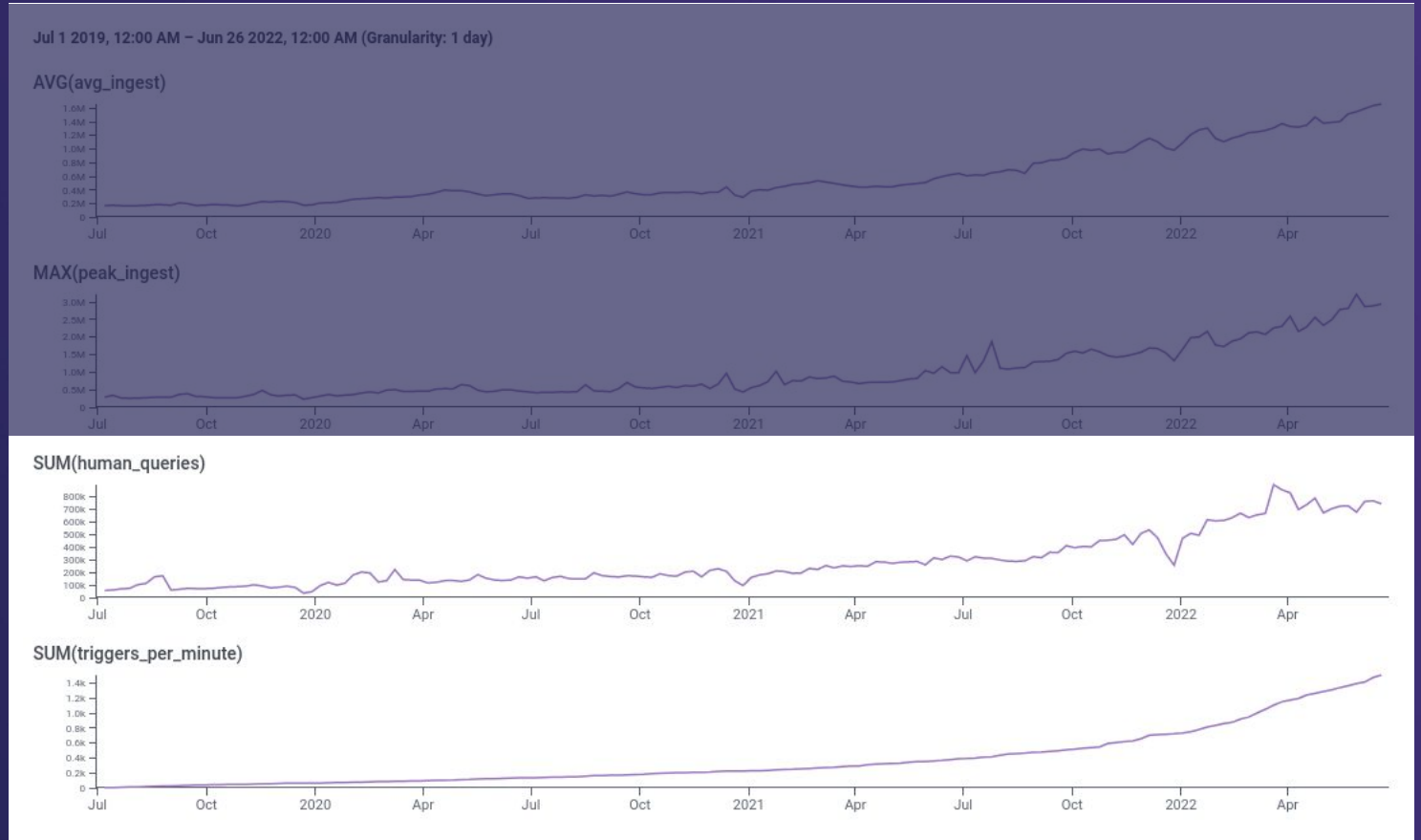
COUNT_DISTINCT(global.build_id)
1,564

elapsed query time: 4.951435582s rows examined: 13,301,219,529 nodes reporting: 100%

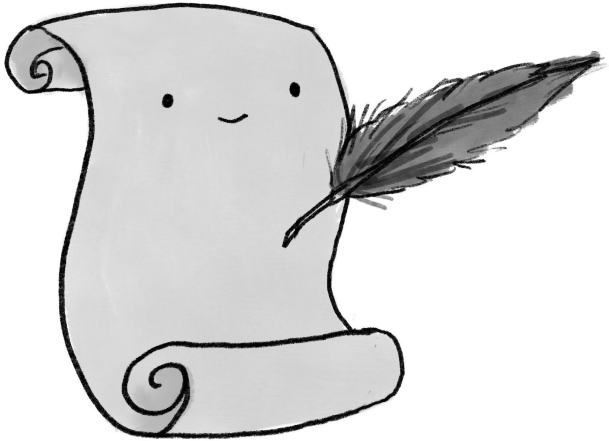
10x growth in three years



10x growth in three years



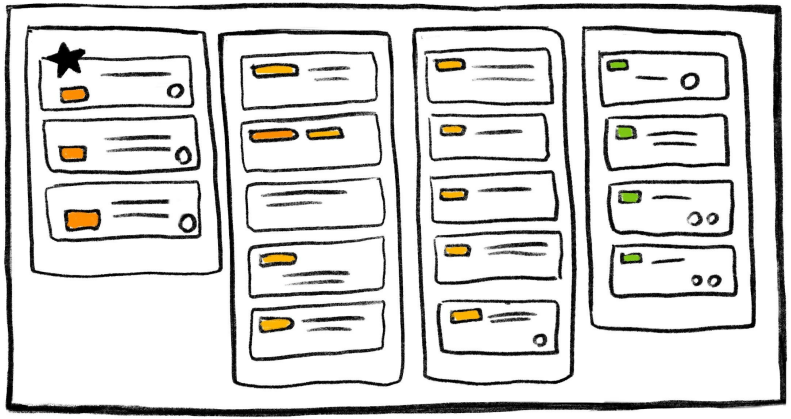
Our confidence recipe:



Quantify reliability.

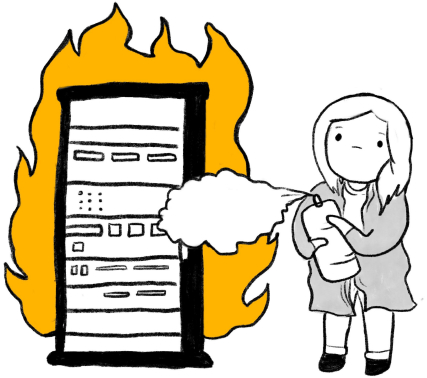
Identify potential areas of risk.

Design experiments to probe risk.



Prioritize addressing risks.

Measuring reliability:

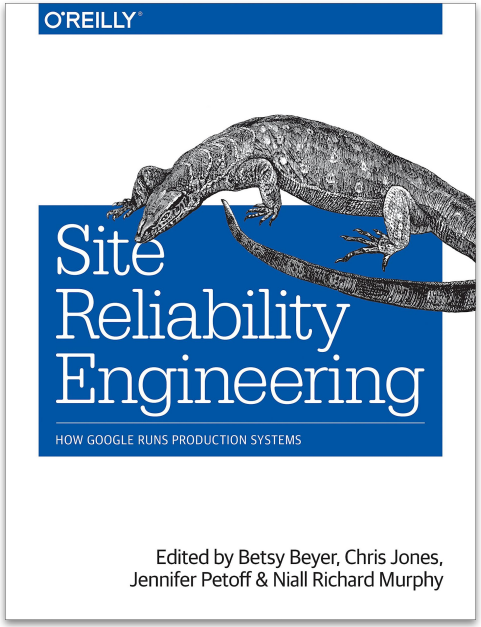


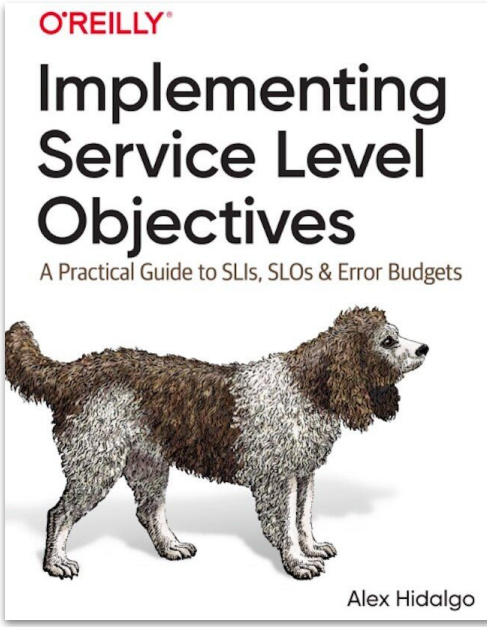
How broken is “too broken”?

Service Level Objectives (SLOs)

Define and measure success!

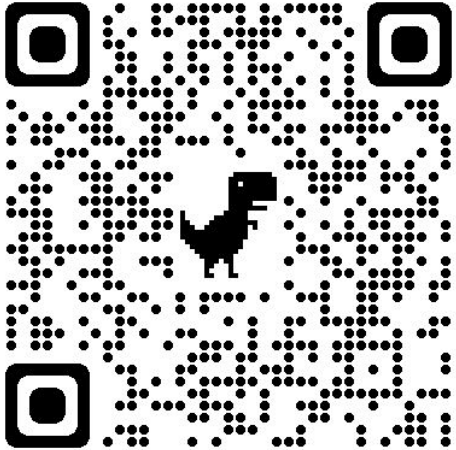
Popularized by Google, widely adopted now!





Service-Level Objectives

- Described in Chapter 4 of the Google SRE Book
- Based on what users actually want, not what's in a contract
- Don't plan for 100% uptime.
- SLOs are an opportunity to negotiate!
- See also: #YOW19 talk by yours truly



Service-Level Objectives

- Negotiation:
  - What's important to your customers?
  - What can everyone agree on? If it's not important, maybe it's not worth measuring in an SLO?

- Paging only on SLOs helps reduce pager fatigue.

DATA GENERATION → STREAMING INGEST → STORAGE + PROCESSING → VISUAL ANALYSIS LOOP

Logging Data

- Amazon Relational Database Service (Amazon RDS)
- AWS Elastic Load Balancing
- AWS Elastic Beanstalk
- Amazon Elastic Kubernetes Service (Amazon EKS)

Metrics Data

- Host Metrics
- App Metrics
- Amazon CloudWatch
- Prometheus

OpenTelemetry

Trace Span Data

- JavaScript
- Go
- Jaeger
- Java
- Ruby
- Python
- .NET
- Front-End

OpenTelemetry

Realtime Ingest
Auth and validation

Honeycomb API

Refinery
User-controlled dynamic tail sampling

Unpacked Into Columns

Query Engine

Unlimited users can store thousands of dimensions at no additional cost and query any arbitrary combinations without pre-aggregation.

Proprietary distributed computing and parallelized processing returns query results in < 3 seconds across billions of rows of data.

Columnar Datastore

High-cardinality column file

Segment with time range

Amazon Simple Storage Service (Amazon S3)

Context
Column
Segment

AWS Lambda

ui.honeycomb.io

ui.honeycomb.io

ui.honeycomb.io

Events contain many fields and distinct values

"Wide events" packed with as much context as you need for debugging

Graph Rendering
Click through visuals based on granular data

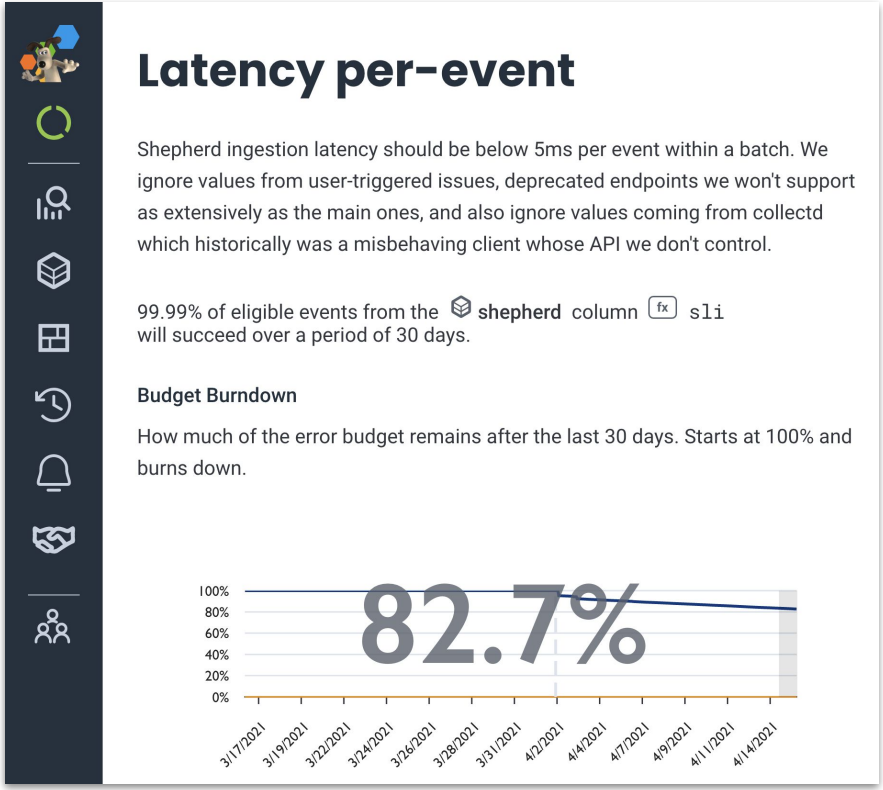
Query Builder UI
Intuitive GUI with multiple group-by

Query Result History
Shared team intelligence

SLOs are user flows

Honeycomb's SLOs

- home page loads quickly
- user-run queries are fast
- customer data gets ingested fast



Service-Level Objectives

- Example Service-Level Indicators:
  - 99.99% of events succeed without error over a period of 30 days.
  - 99.99% of events are processed in 5ms over a period of 30 days.

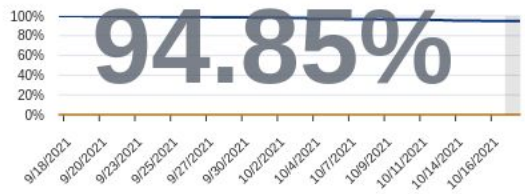
- Translates to about 4.5 minutes of violation in a month.

99.99% of eligible events from the shepherd column

fx error_sli will succeed over a period of 30 days.

Budget Burndown

How much of the error budget remains after the last 30 days. Starts at 100% and burns down.



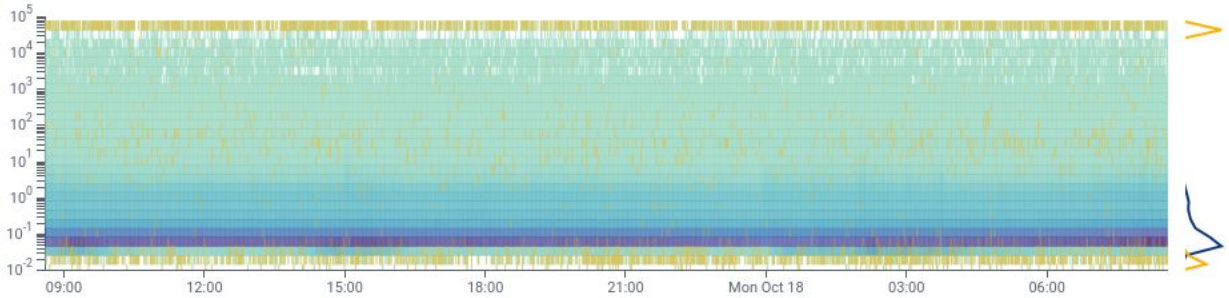
Historical SLO Compliance

For each day of the past 30, how often this SLI has succeeded over the preceding 30 days.

Reset

Distribution of Events failing SLI by filt duration_ms Make Default Last 24 hours

Oct 17 2021, 8:36:05 AM – Oct 18 2021, 8:36:05 AM



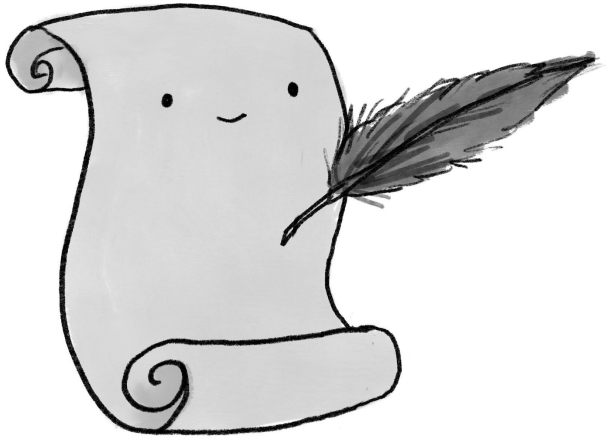
How to stay within SLO

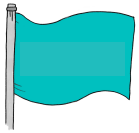
Minimise time to mitigate & recover.

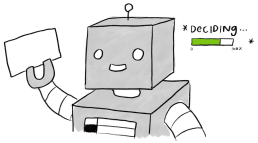
Software delivery performance metric	Elite	High	Medium	Low
Deployment frequency For the primary application or service you work on, how often does your organization deploy code to production or release it to end users?	On-demand (multiple deploys per day)	Between once per week and once per month	Between once per month and once every 6 months	Fewer than once per six months
Lead time for changes For the primary application or service you work on, what is your lead time for changes (i.e., how long does it take to go from code committed to code successfully running in production)?	Less than one hour	Between one day and one week	Between one month and six months	More than six months
Time to restore service For the primary application or service you work on, how long does it generally take to restore service when a service incident or a defect that impacts users occurs (e.g., unplanned outage or service impairment)?	Less than one hour	Less than one day	Between one day and one week	More than six months
Change failure rate For the primary application or service you work on, what percentage of changes to production or released to users result in degraded service (e.g., lead to service impairment or service outage) and subsequently require remediation (e.g., require a hotfix, rollback, fix forward, patch)?	0%-15%	16%-30%	16%-30%	16%-30%

Accelerate: State of DevOps 2021

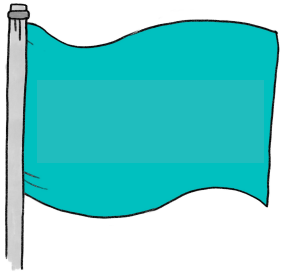
What's our recipe?

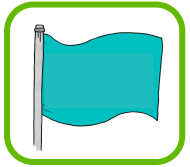


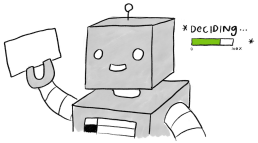




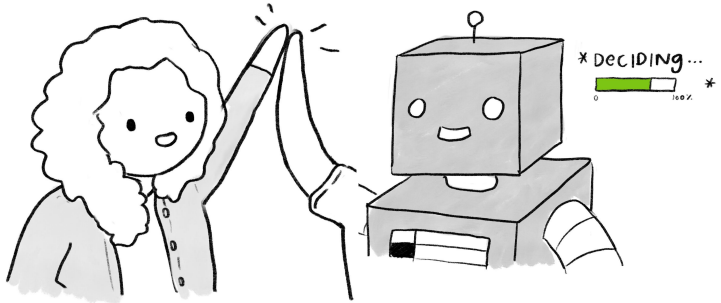
Design for feature flag deployment.

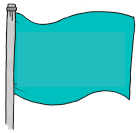


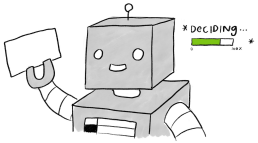




Automated integration.

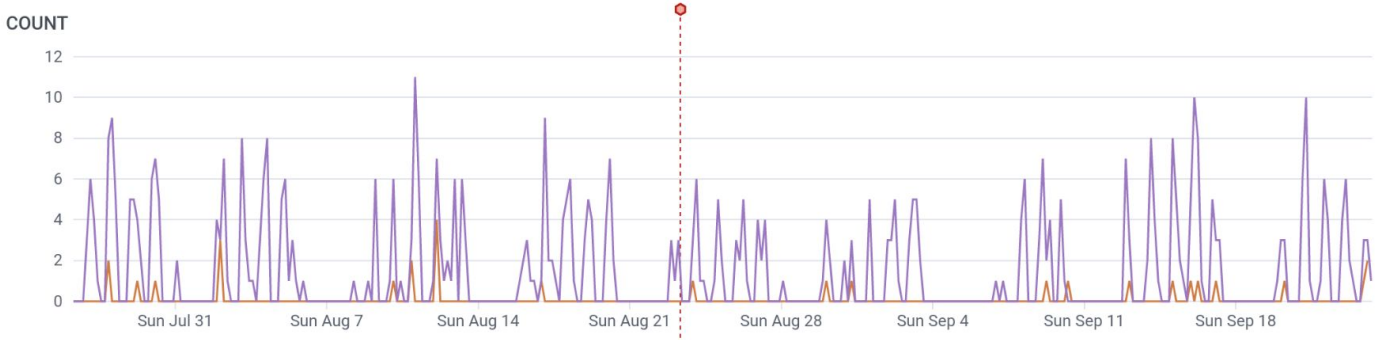




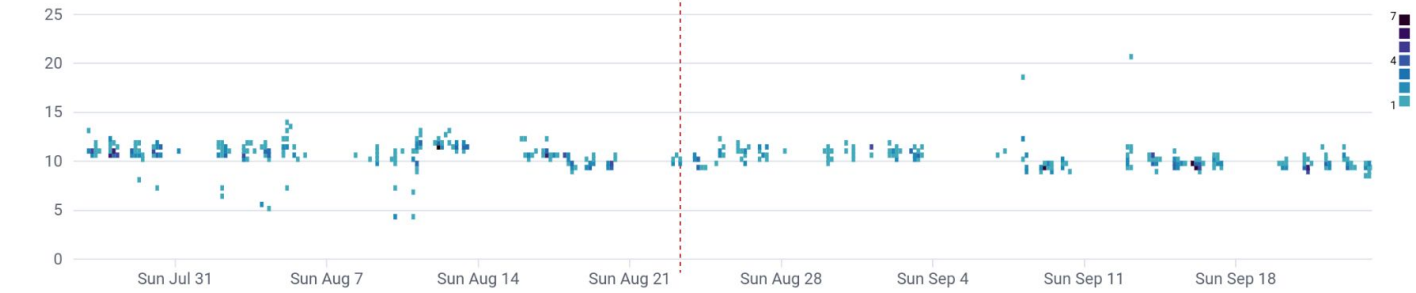


Jul 26 2022 07:07:58 – Sep 24 2022 07:07:58 UTC+10:00 (Granularity: 4 hr)

COUNT



HEATMAP(duration_min)



status	COUNT	HEATMAP(duration_min)
success	521	
failed	29	

CI/C >

VISUALIZE	WHERE	GROUP BY
HEATMAP(duration_min)	trace.parent_id does-not-exist branch = main status = success repo = git@github.com:honeycombio/hound.git	None; don't segment
+ ORDER BY	+ LIMIT	+ HAVING

Run Query

Run a few seconds ago

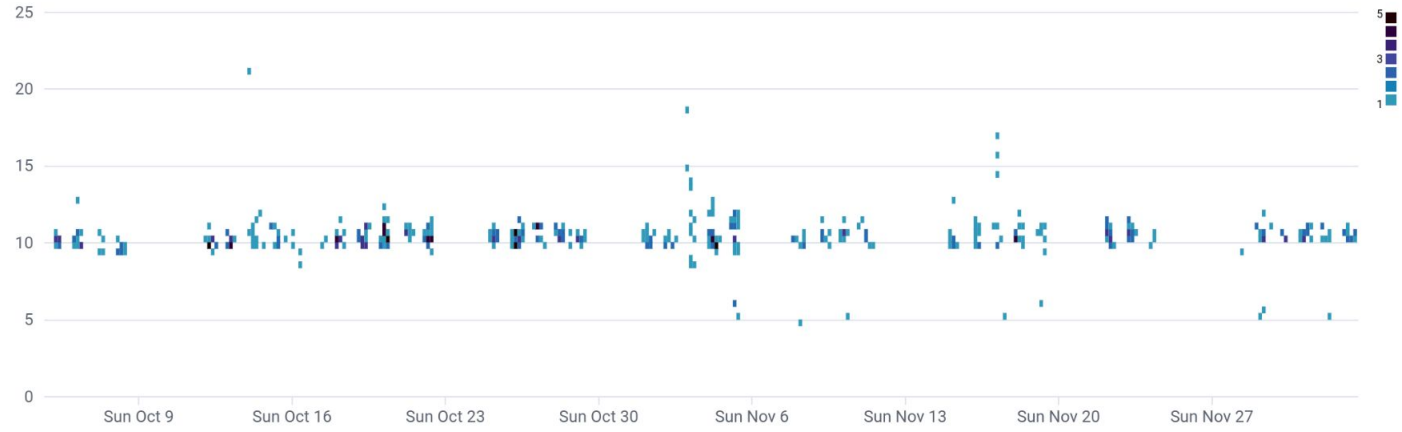
Results | BubbleUp | Metrics | Traces | Raw Data

Compare to Previous time range

Graph Settings

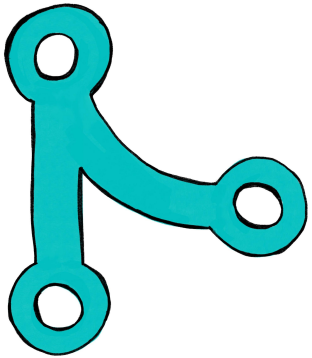
Oct 4 2022 16:31:20 – Dec 3 2022 16:31:20 UTC+11:00 (Granularity: 4 hr)

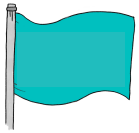
HEATMAP(duration_min)

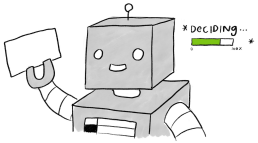


@lizthegrey at #YOW22

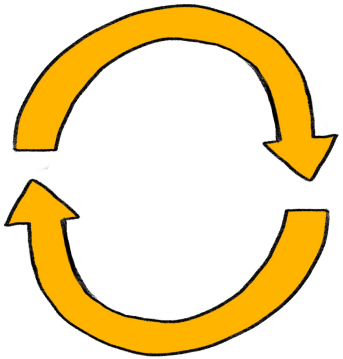
Green button merge.

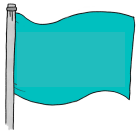


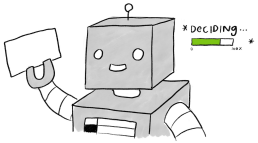




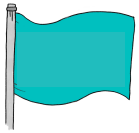
Auto-updates, rollbacks, & pins.

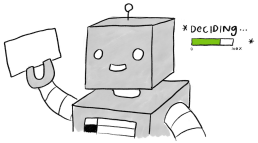




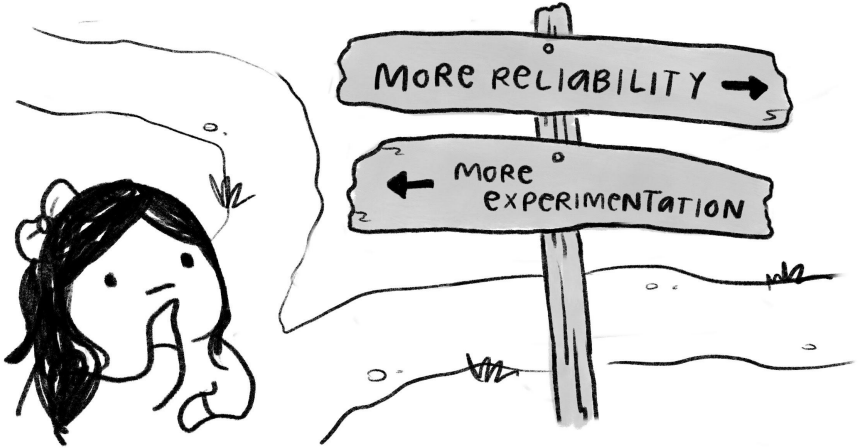


Observe behavior in prod.



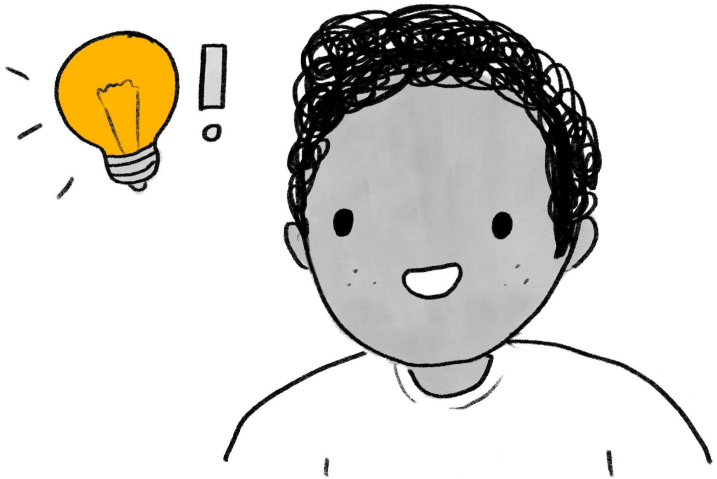


Validating our expectations

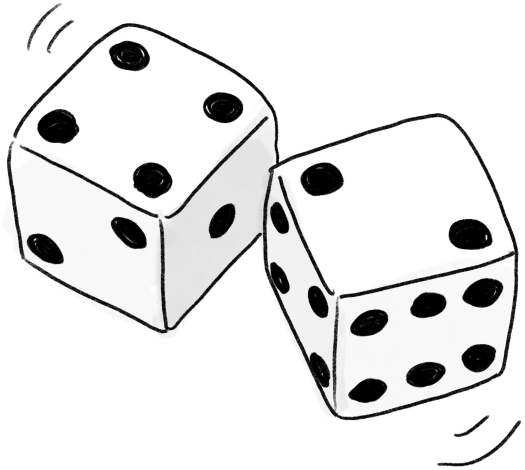


Experiment using error budgets.

(only when there's error budget left)



Keep learning objectives in mind.



Always ensure safety.

Targeting Insights Variations History Code references

Targeting ON

Prerequisites

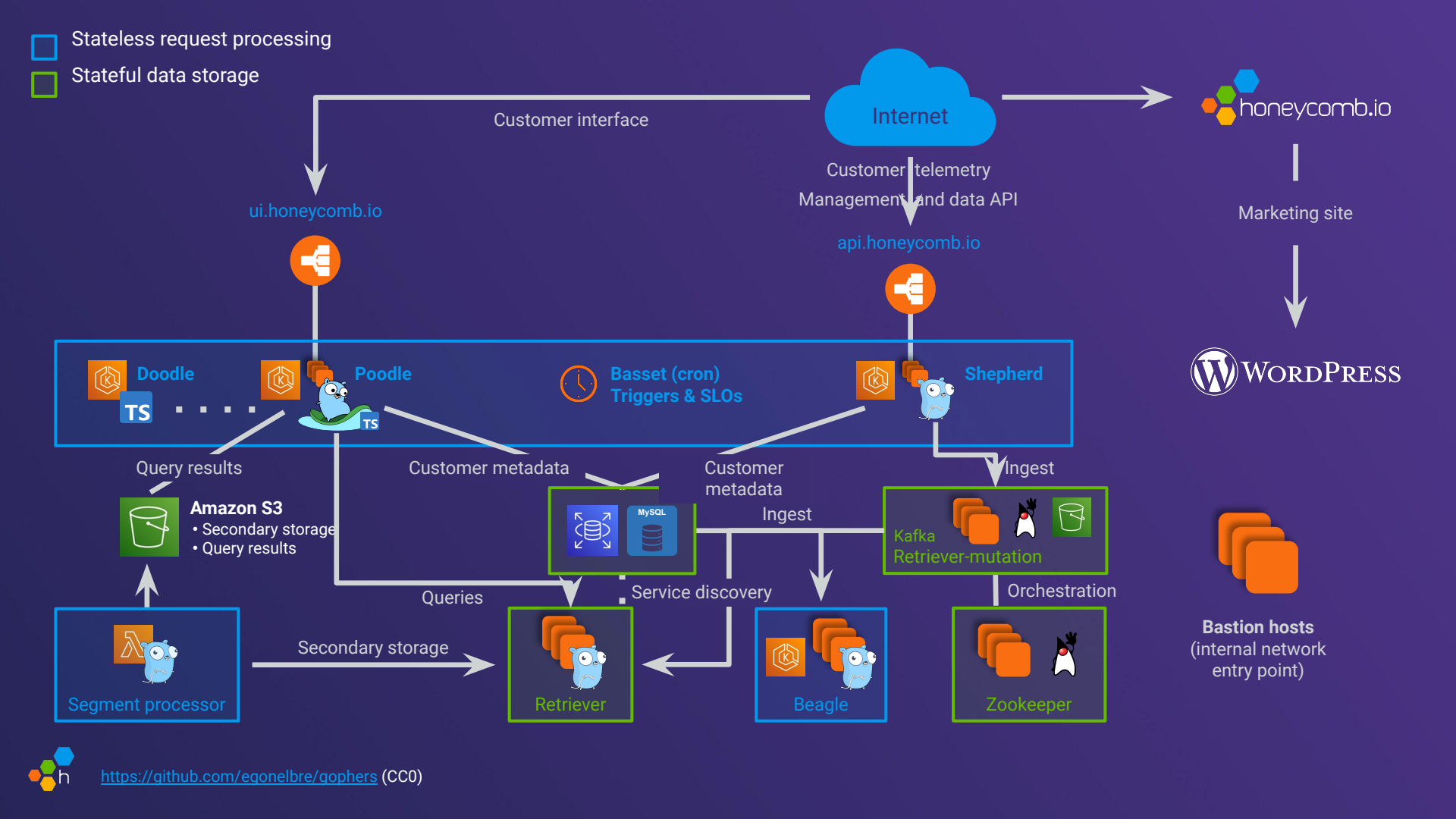
Target individual users

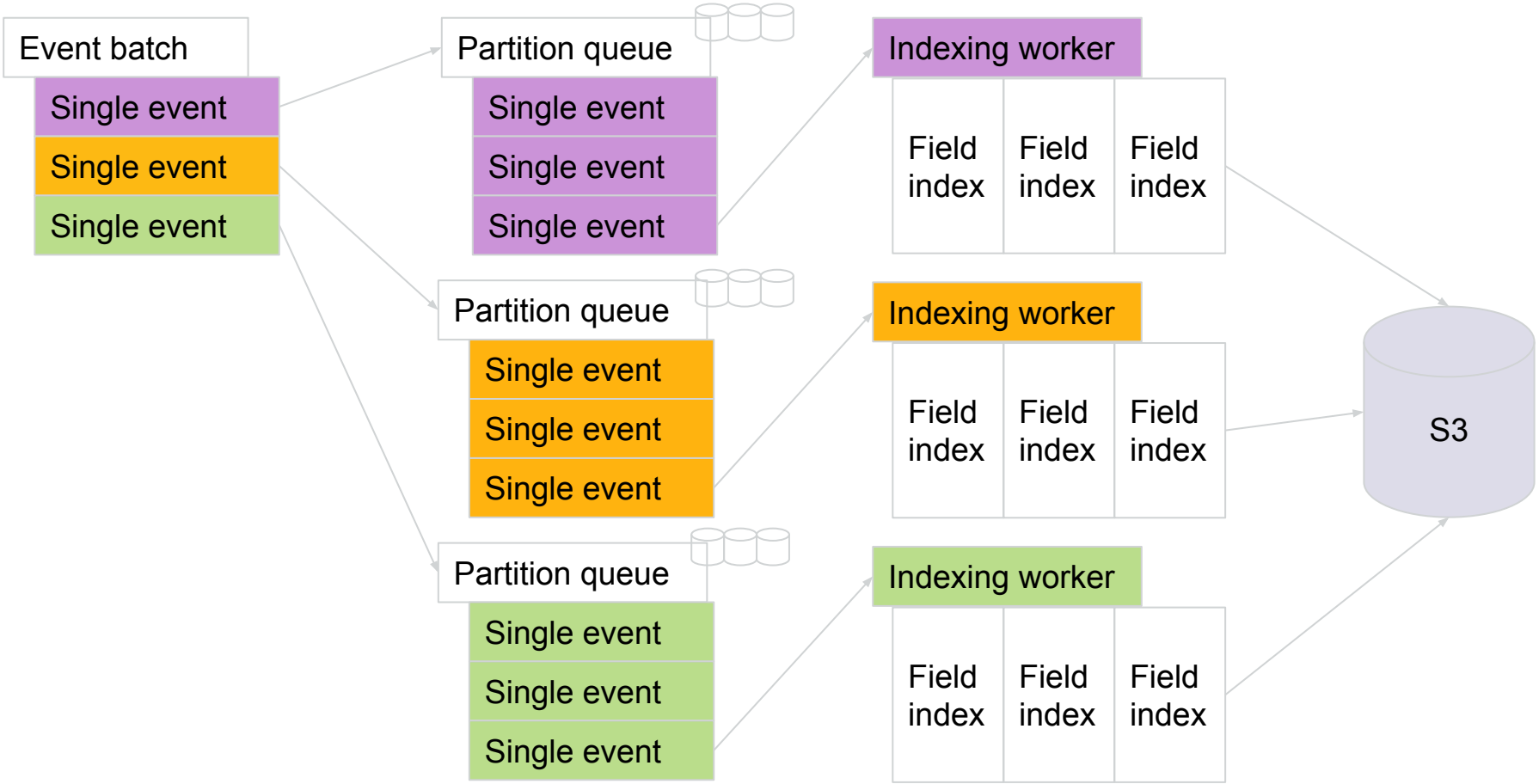
Target users who match these rules

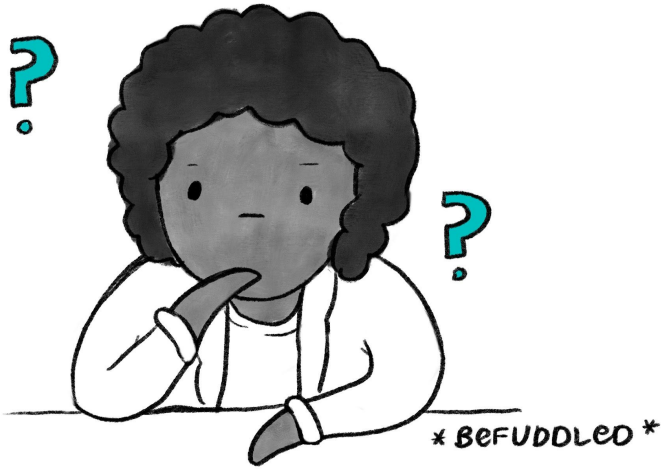
IF team_id is one of

SERVE true

Data persistence is tricky.

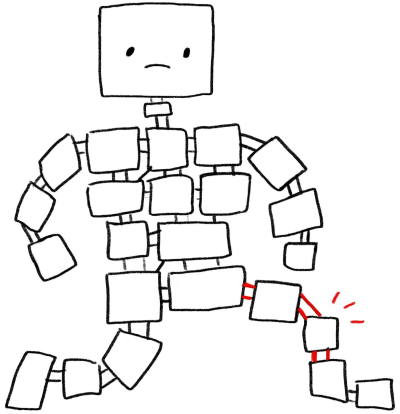

Stateless request processing
Stateful data storage
Internet
Customer interface
honeycomb.io
Customer telemetry
Management and data API
ui.honeycomb.io
api.honeycomb.io
Marketing site
WordPress
Doodle
TS
Poodle
TS
Basset (cron)
Triggers & SLOs
Shepherd
Query results
Customer metadata
Customer metadata
Ingest
Ingest
Amazon S3
• Secondary storage
• Query results
MySQL
Kafka
Retriever-mutation
Queries
Service discovery
Orchestration
Secondary storage
Segment processor
Retriever
Beagle
Zookeeper
Bastion hosts
(internal network entry point)
https://github.com/egonelbre/gophers (CC0)


Event batch
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Indexing worker
Field index
Field index
Field index
Indexing worker
Field index
Field index
Field index
Indexing worker
Field index
Field index
Field index
S3

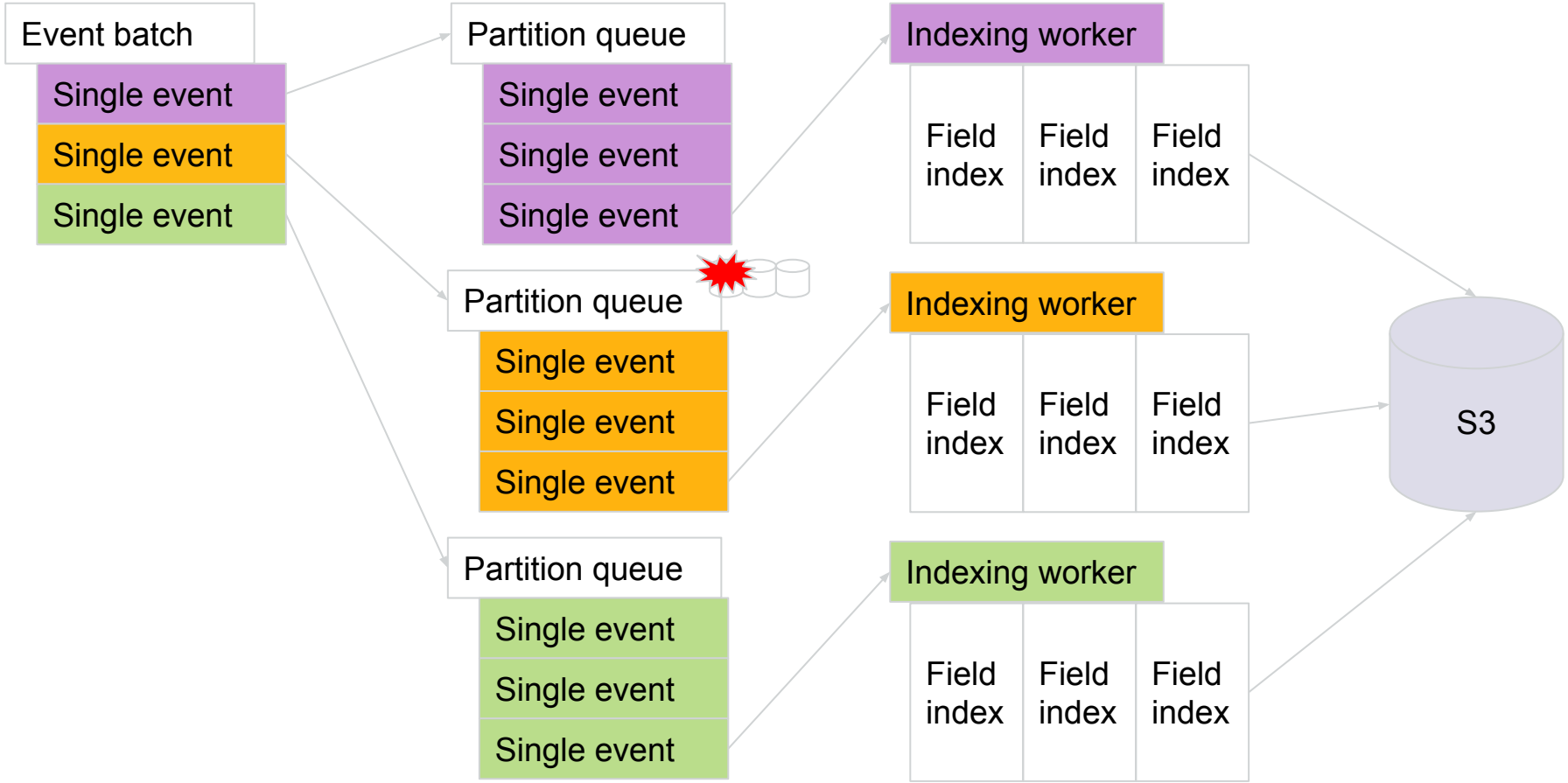


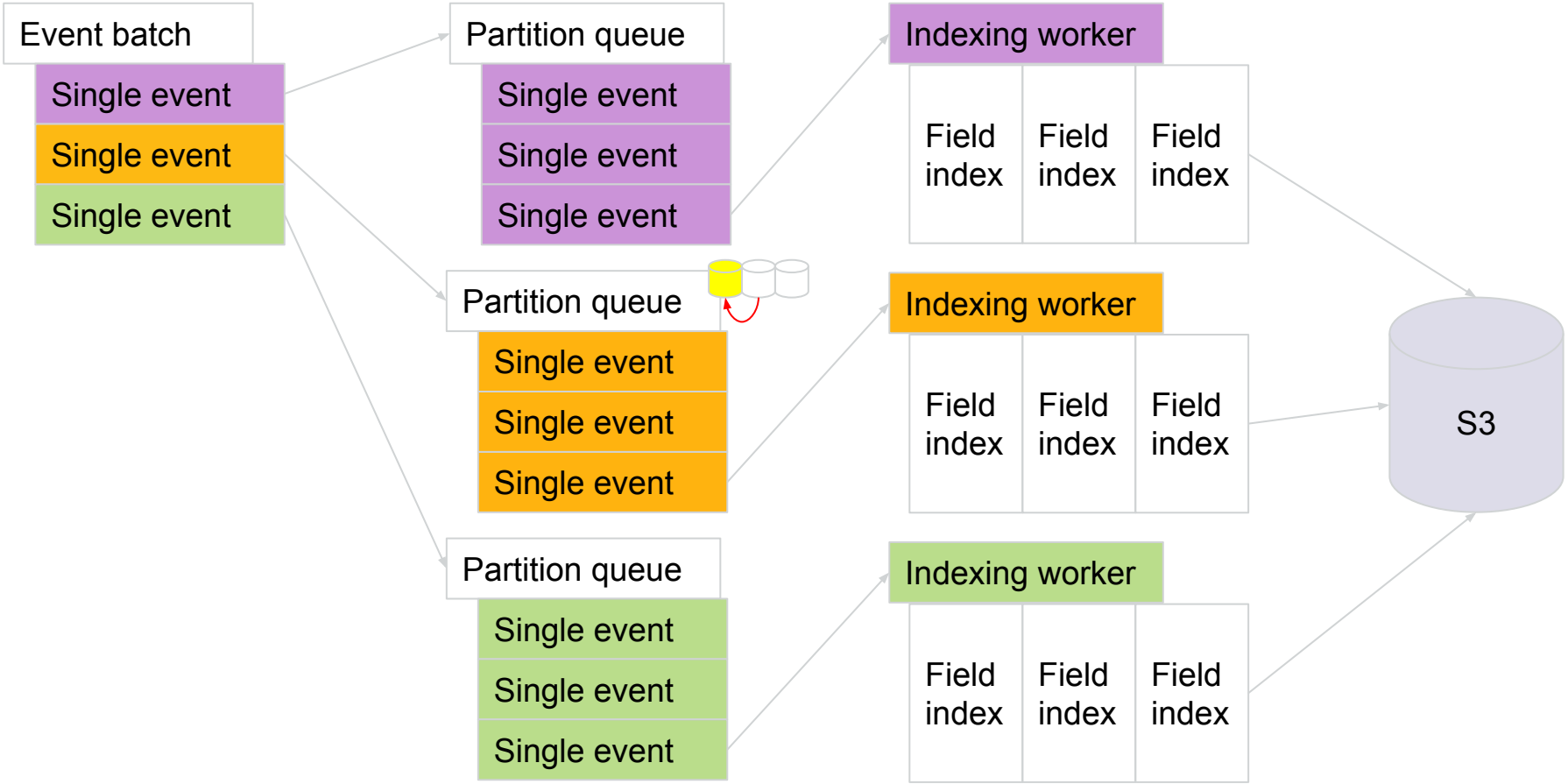
Infrequent changes.

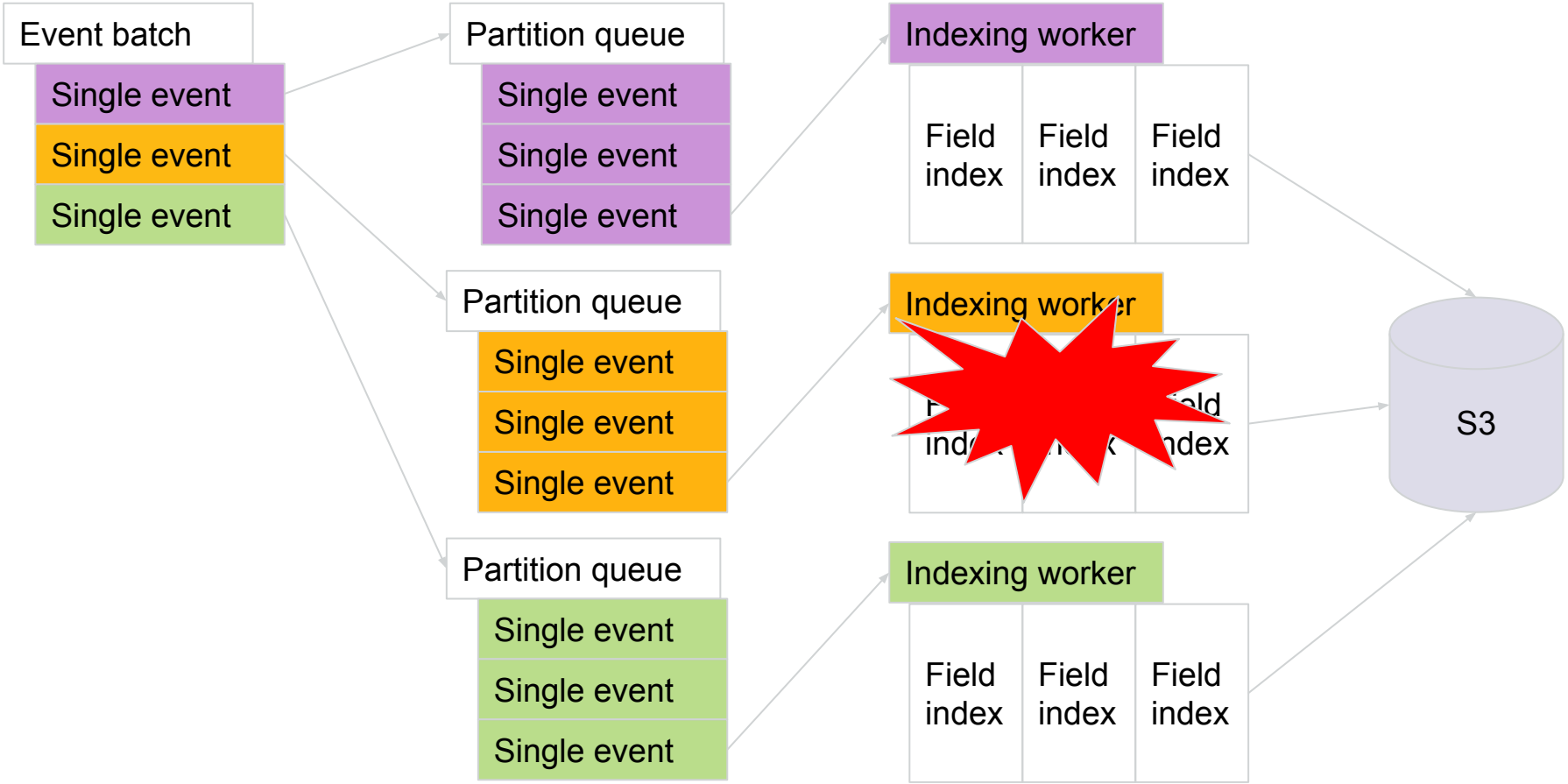
Data integrity and consistency.

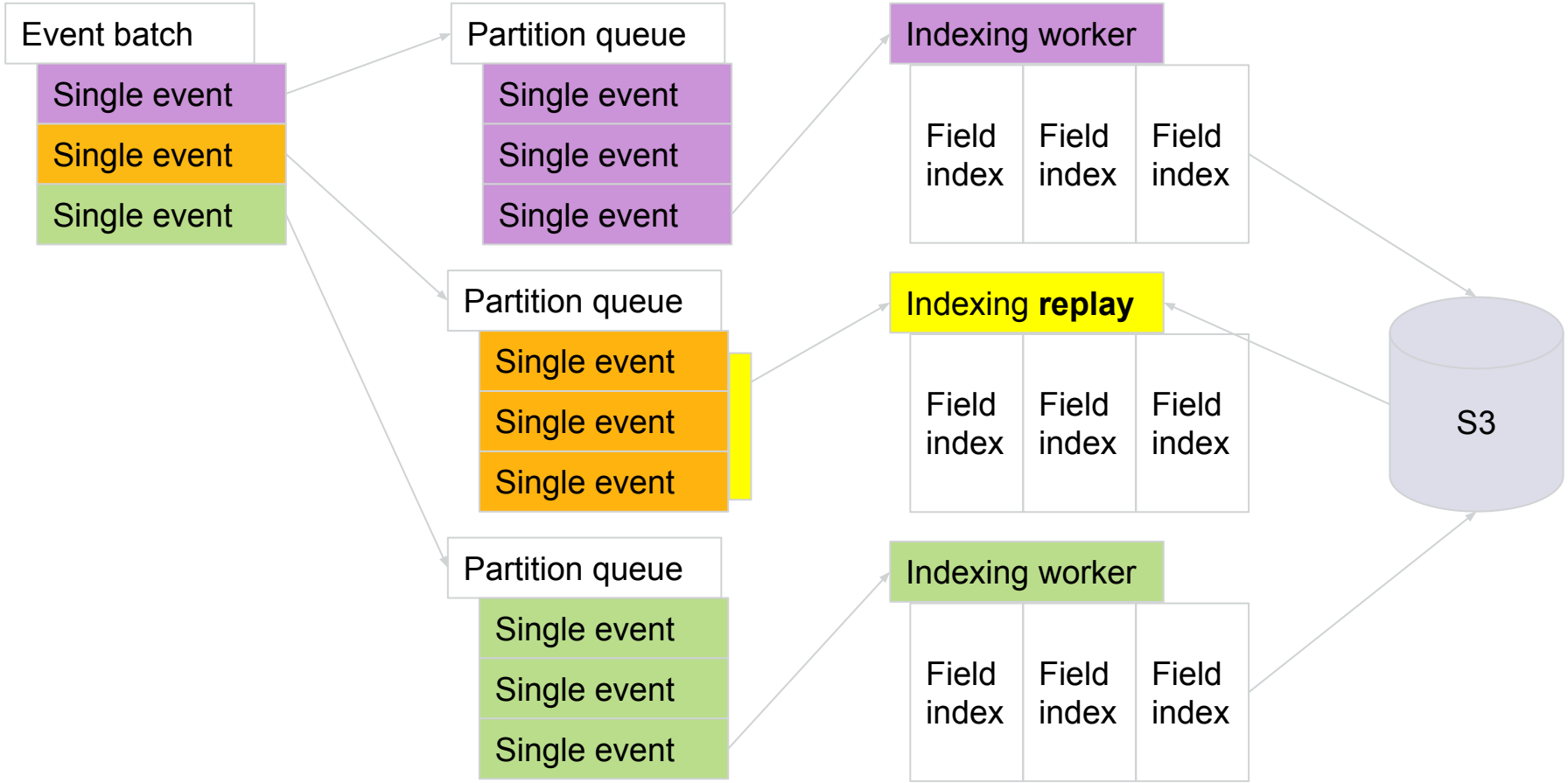


Delicate failover dances

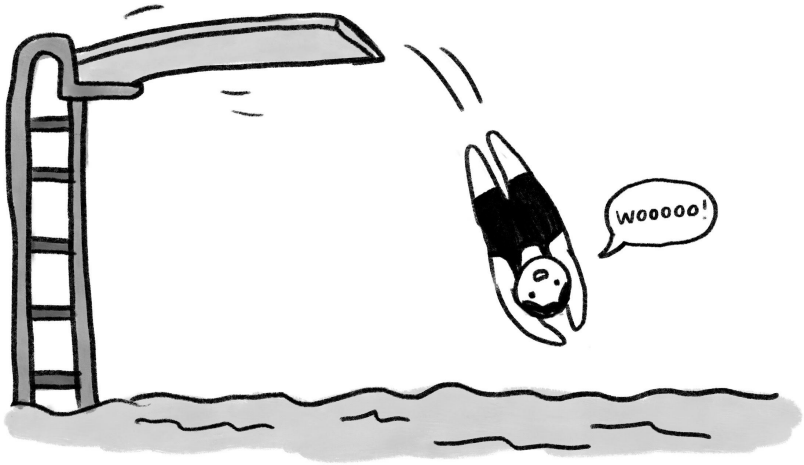

Event batch
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Indexing worker
Field index
Field index
Field index
Indexing worker
Field index
Field index
Field index
Indexing worker
Field index
Field index
Field index
S3

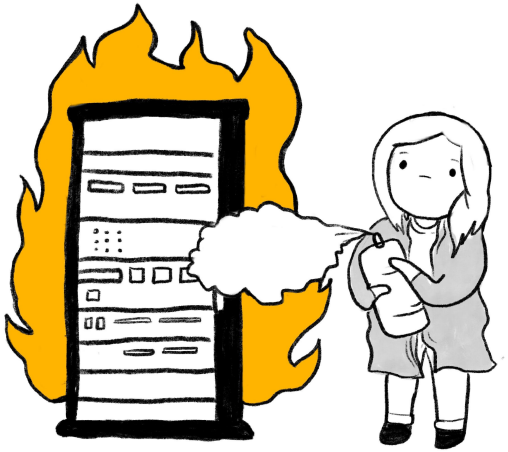

Event batch
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Indexing worker
Field index
Field index
Field index
Indexing worker
Field index
Field index
Field index
Indexing worker
Field index
Field index
Field index
S3


Event batch
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Indexing worker
Field index
Field index
Field index
Indexing worker
Indexing worker
Field index
Field index
Field index
S3

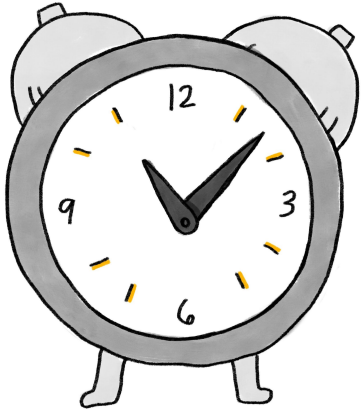

Event batch
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Indexing worker
Field index
Field index
Field index
Indexing replay
Field index
Field index
Field index
Indexing worker
Field index
Field index
Field index
S3

Experimenting in prod



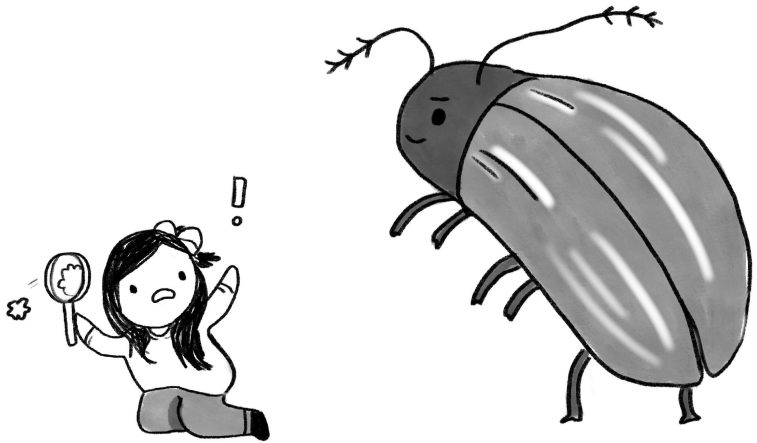


Restart one server & service at a time.

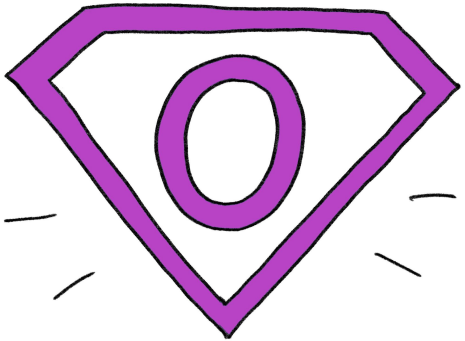


At 3pm, not at 3am.

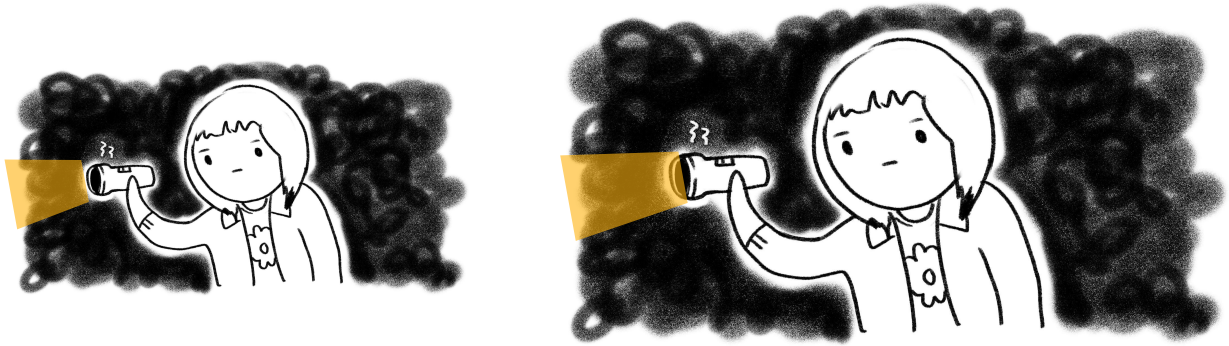
"Bugs are shallow with more eyes."



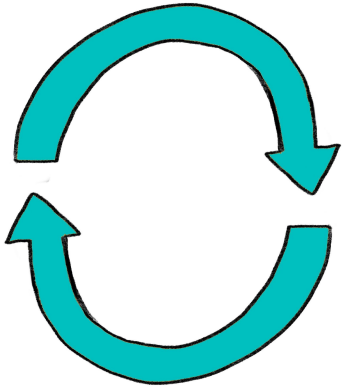
Monitor for changes using SLIs.



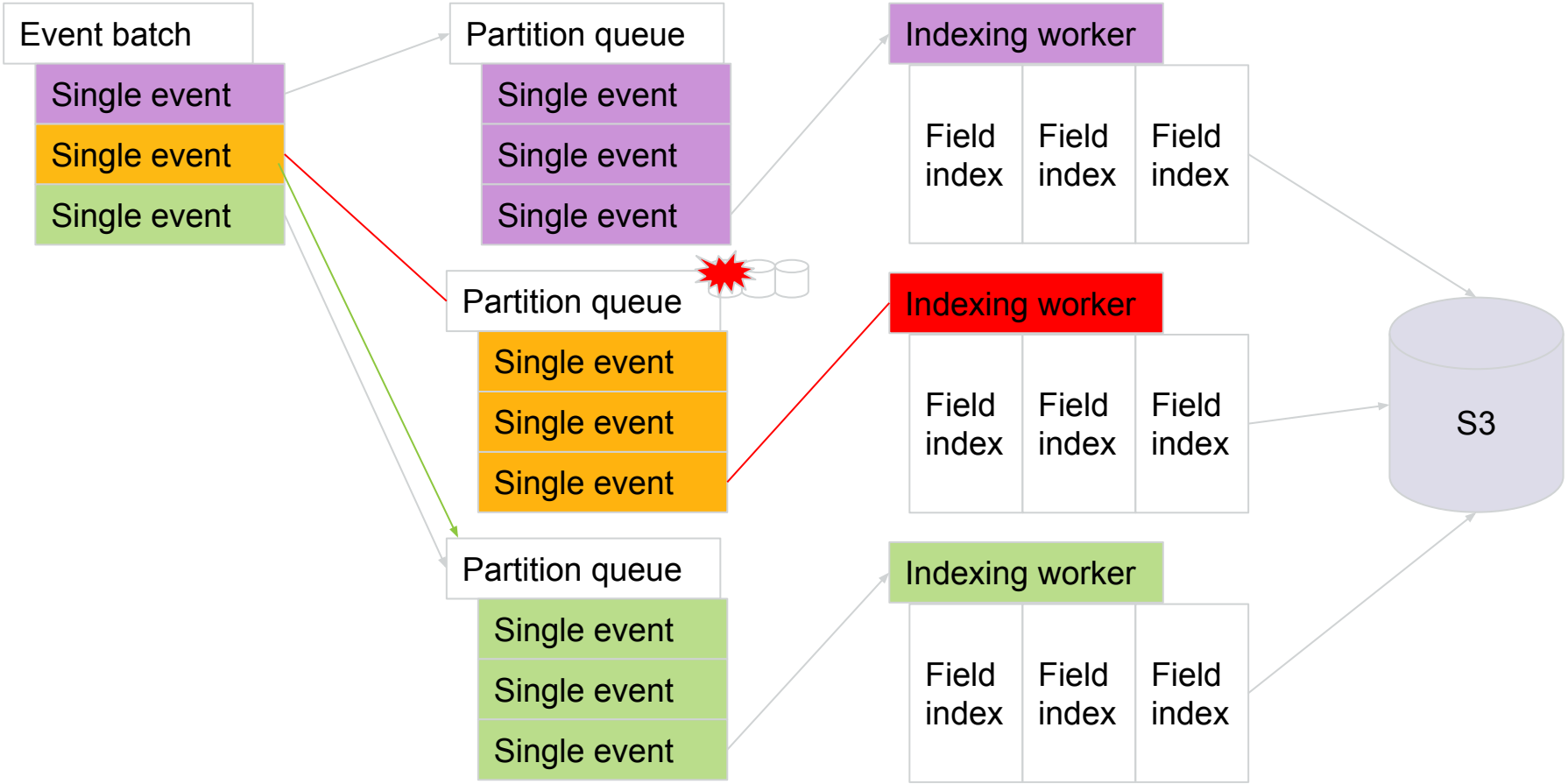
Debug with observability.

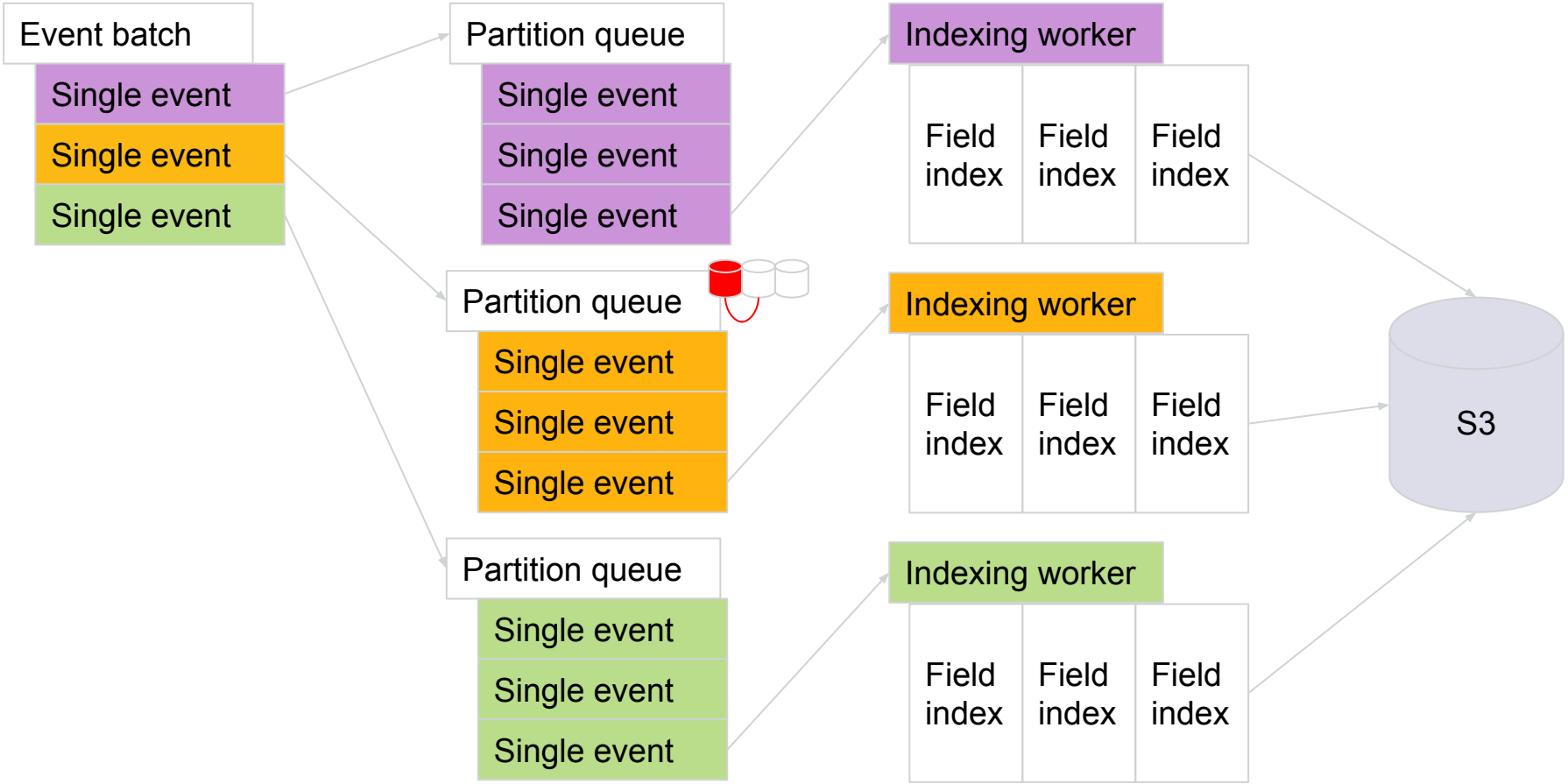


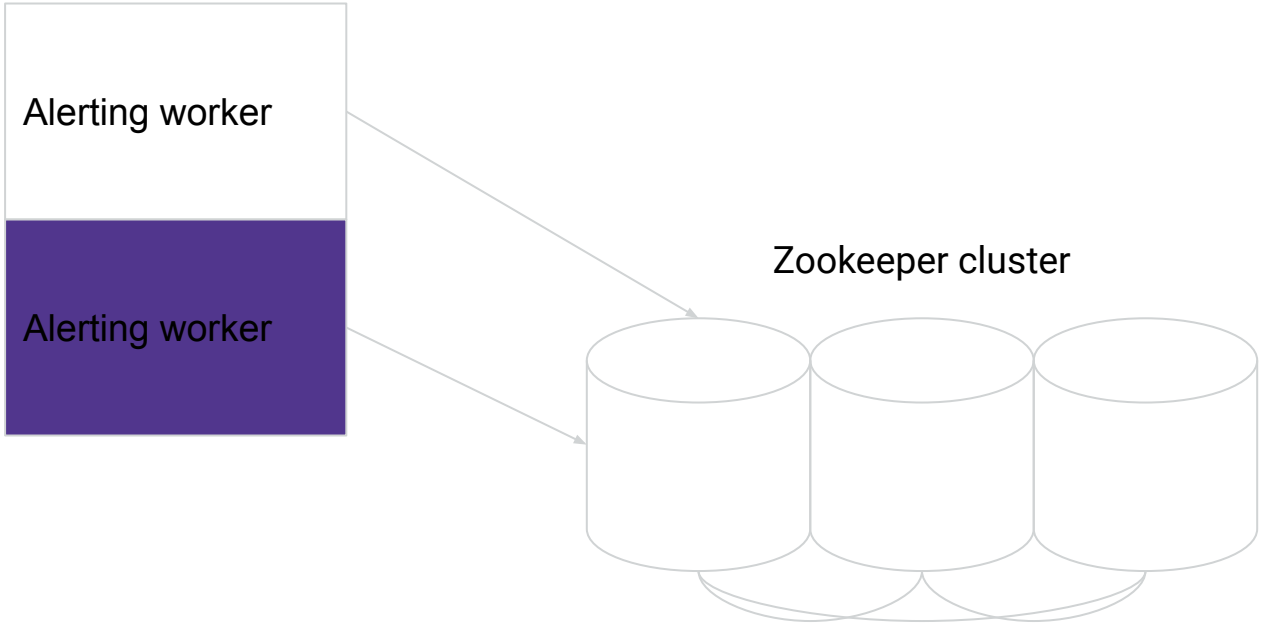
Test the telemetry too!

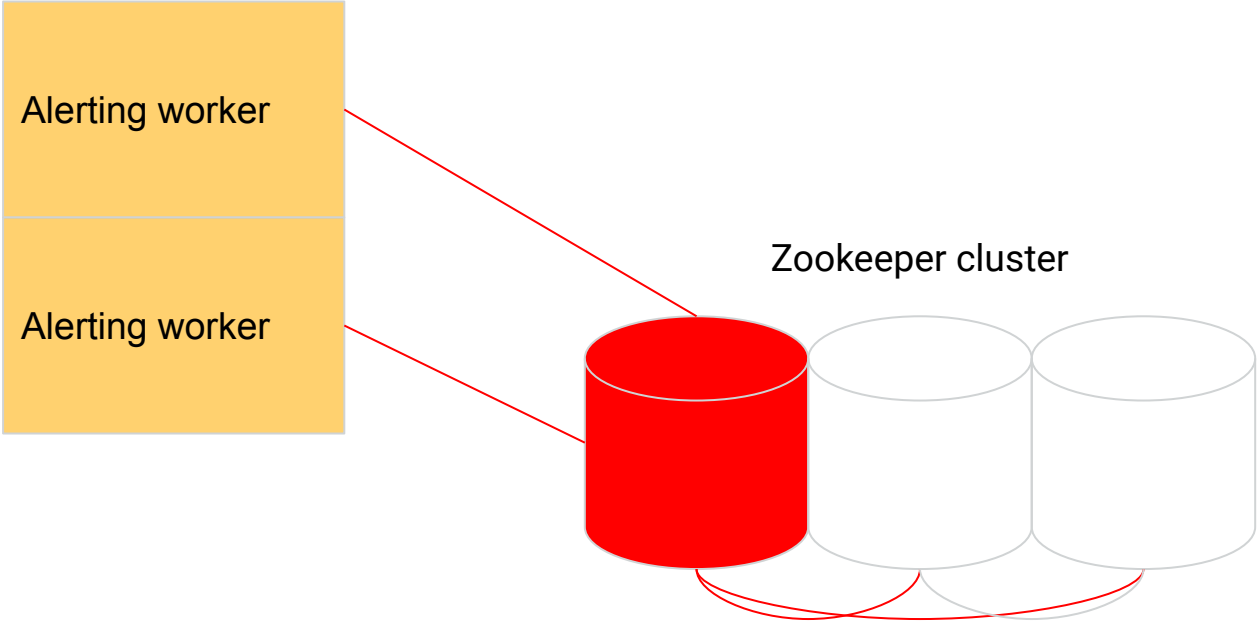


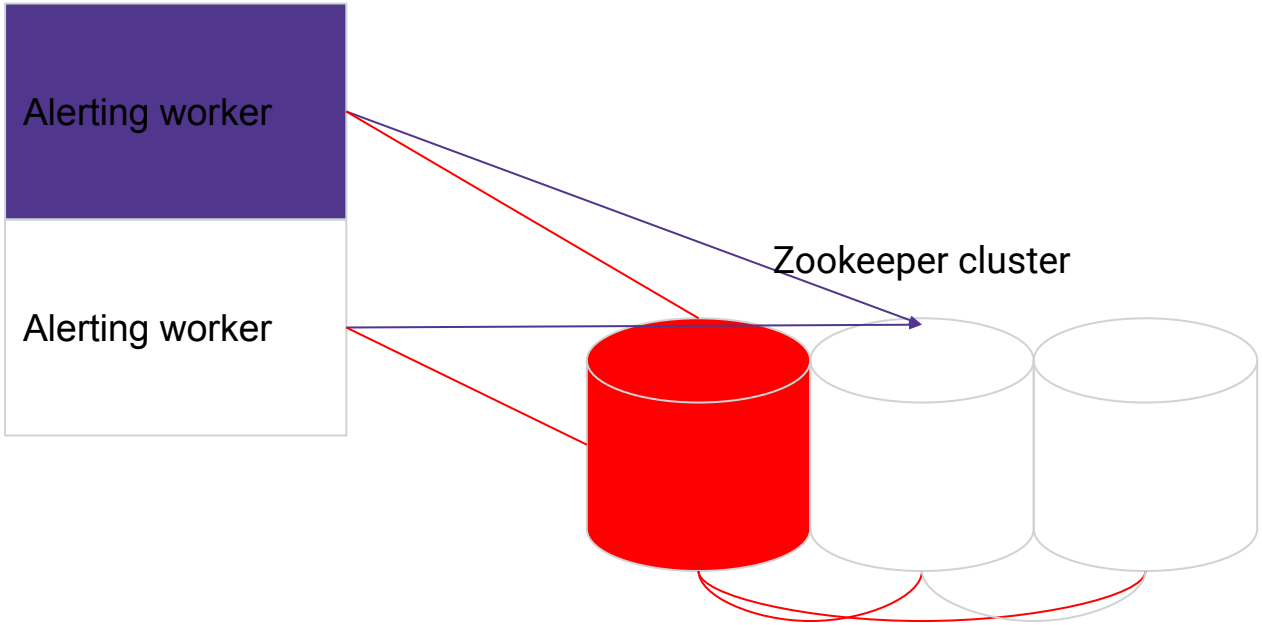
Verify fixes by repeating.

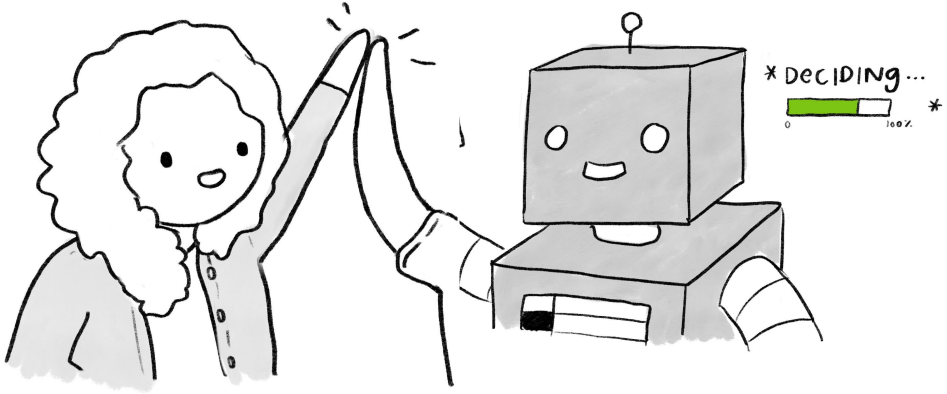

Event batch
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Indexing worker
Field index
Field index
Field index
Indexing worker
Field index
Field index
Field index
Indexing worker
Field index
Field index
Field index
S3


Event batch
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Partition queue
Single event
Single event
Single event
Indexing worker
Field index
Field index
Field index
Indexing worker
Field index
Field index
Field index
Indexing worker
Field index
Field index
Field index
S3

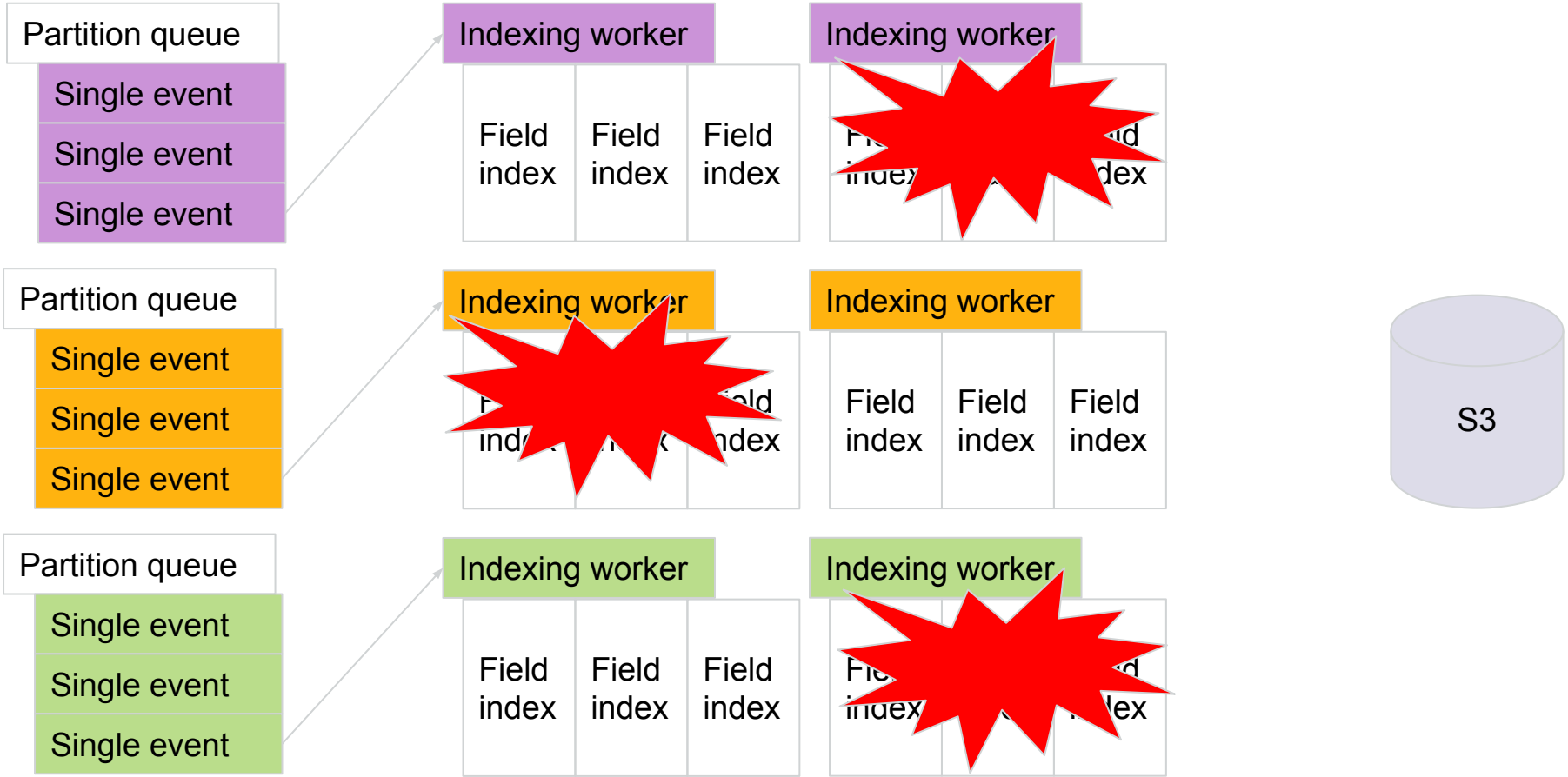

Alerting worker
Alerting worker
Zookeeper cluster

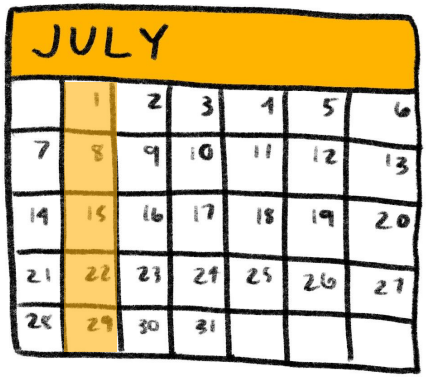

Alerting worker
Alerting worker
Zookeeper cluster


Alerting worker
Alerting worker
Zookeeper cluster



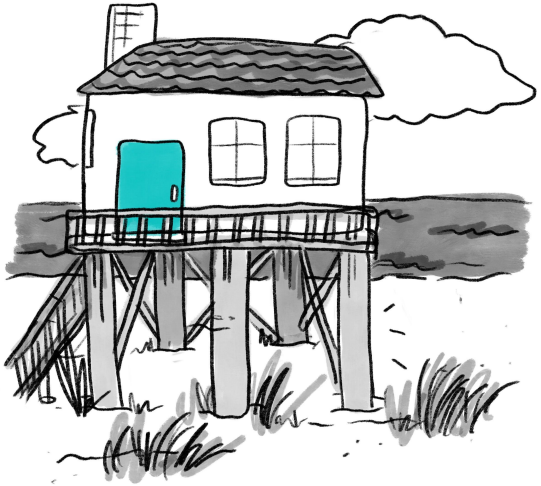
De-risk with design & automation.


Partition queue
Single event
Single event
Single event
Indexing worker
Field index
Field index
Field index
Indexing worker
Partition queue
Single event
Single event
Single event
Indexing worker
Indexing worker
Field index
Field index
Field index
S3
Partition queue
Single event
Single event
Single event
Indexing worker
Field index
Field index
Field index
Indexing worker



Continuously verify to stop regression.

Spot and Kubernetes cause (un)planned chaos.



Chaos testing creates stronger platforms.

Not every experiment succeeds.

But you can mitigate the risks.

Three case studies of failure

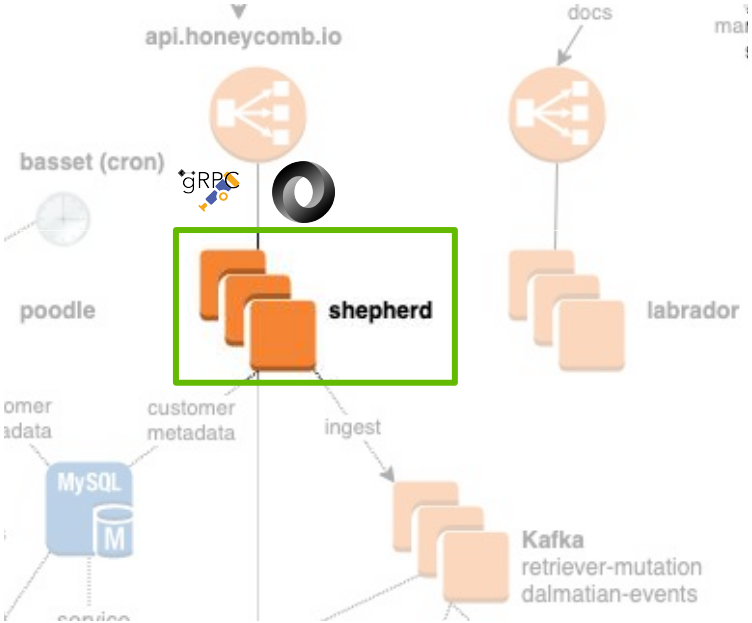
- Ingest service crash
- Kafka instability
- Query performance degradation

and what we learned from each.

1) Shepherd: ingest API service

Shepherd is the gateway to all ingest

- highest-traffic service
- stateless service
- cares about throughput first, latency close second
- used compressed JSON
- gRPC was needed.



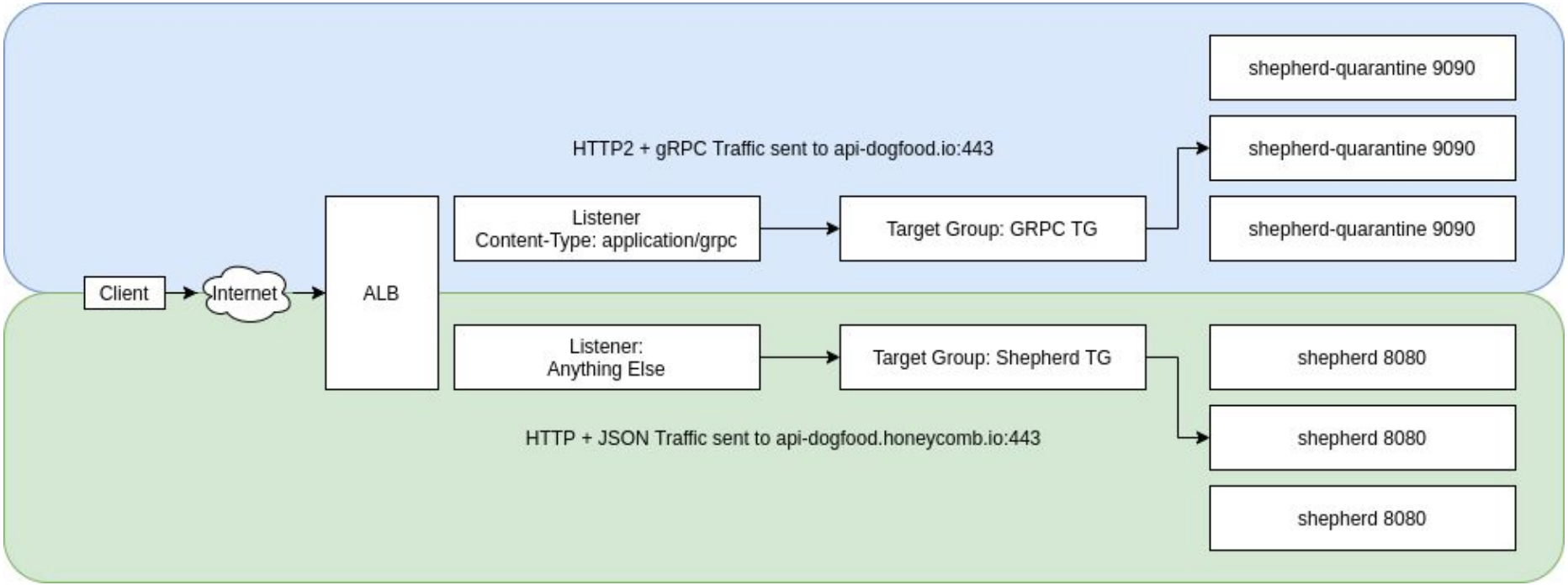
Honeycomb Ingest Outage

- In November 2020, we were working on OTLP and gRPC ingest support
- Let a commit deploy that attempted to bind to a privileged port
- Stopped the deploy in time, but scale-ups were trying to use the new build
- Latency shot up, took more than 10 minutes to remediate, blew our SLO

Now what?

- We could freeze deploys (oh no, don’t do this!)
- Delay the launch? We considered this...
- Get creative!

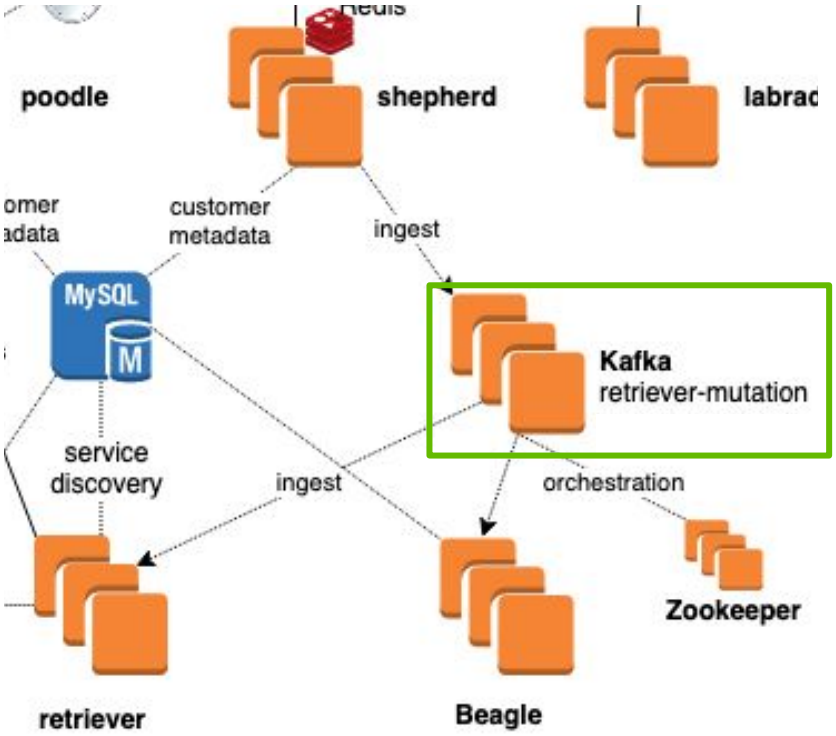
Reduce Risk



2) Kafka: data bus

Kafka provides durability

- Decoupling components provides safety.
- But introduces new dependencies.
- And things that can go wrong.



Our month of Kafka pain

Longtime Confluent Kafka users

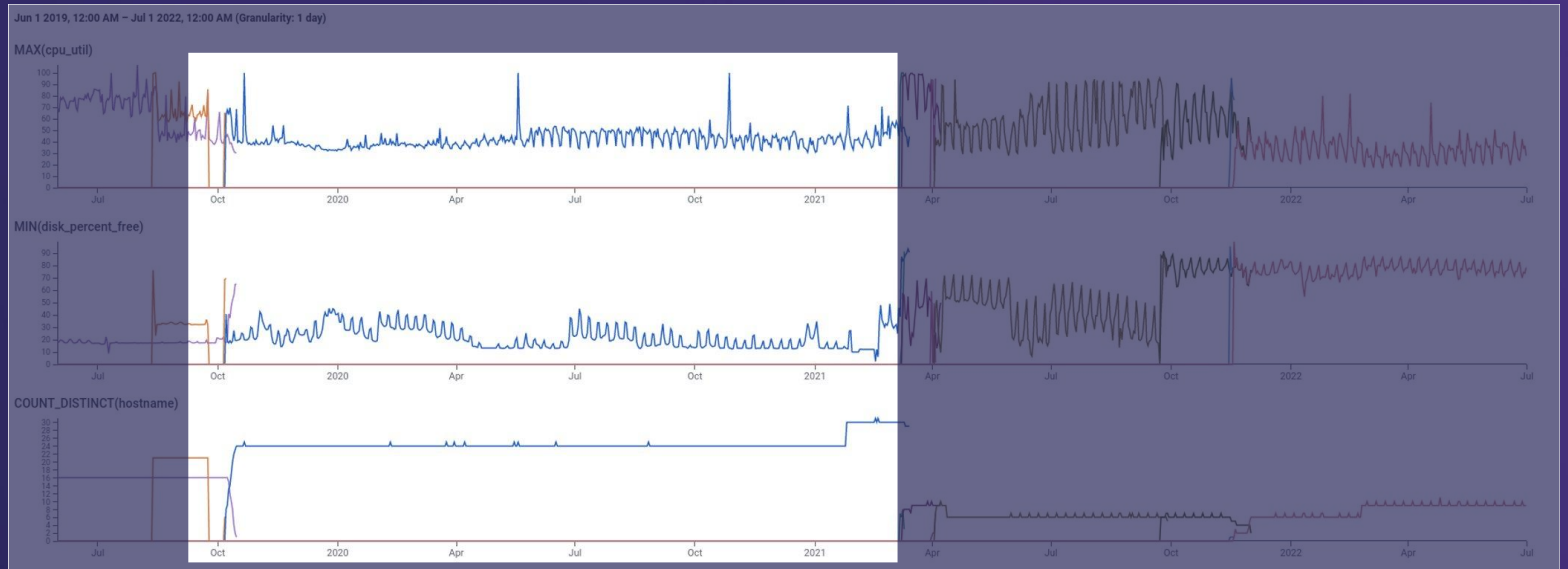
First to use Kafka on Graviton2 at scale

Changed multiple variables at once

- move to tiered storage
- i3en → c6gn
- AWS Nitro

Read more: go.hny.co/kafka-lessons

Finding the right way to migrate Kafka



instance_type	MAX(cpu_util)	MIN(disk_percent_free)	COUNT_DISTINCT(hostname)
c5.xlarge	107	8.93199	16
i3en.large	100.1	23.21438	27
i3en.xlarge	100.1	2.41057	43
m6g.large	100.1	0.32752	9
c6gn.xlarge	99.8	1.01042	17
is4gen.2xlarge	95.7	76.5197	1
i3en.2xlarge	95.6	2.8939	29
c6gn.2xlarge	95.1	1.48059	10
i3en.3xlarge	95	62.79987	13
im4gn.4xlarge	81.9	54.73766	43

elapsed query time: 2.050535309s # results: 10 rows examined: 453,669,920 nodes reporting: 100%

Unexpected constraints

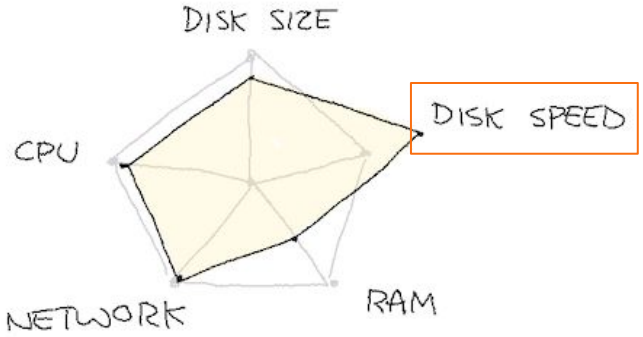
We thrashed multiple dimensions.

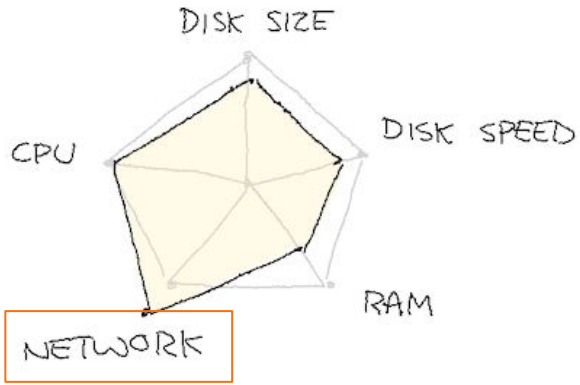
We tickled hypervisor bugs.

We tickled EBS bugs.

Burning our people out wasn't worth it.

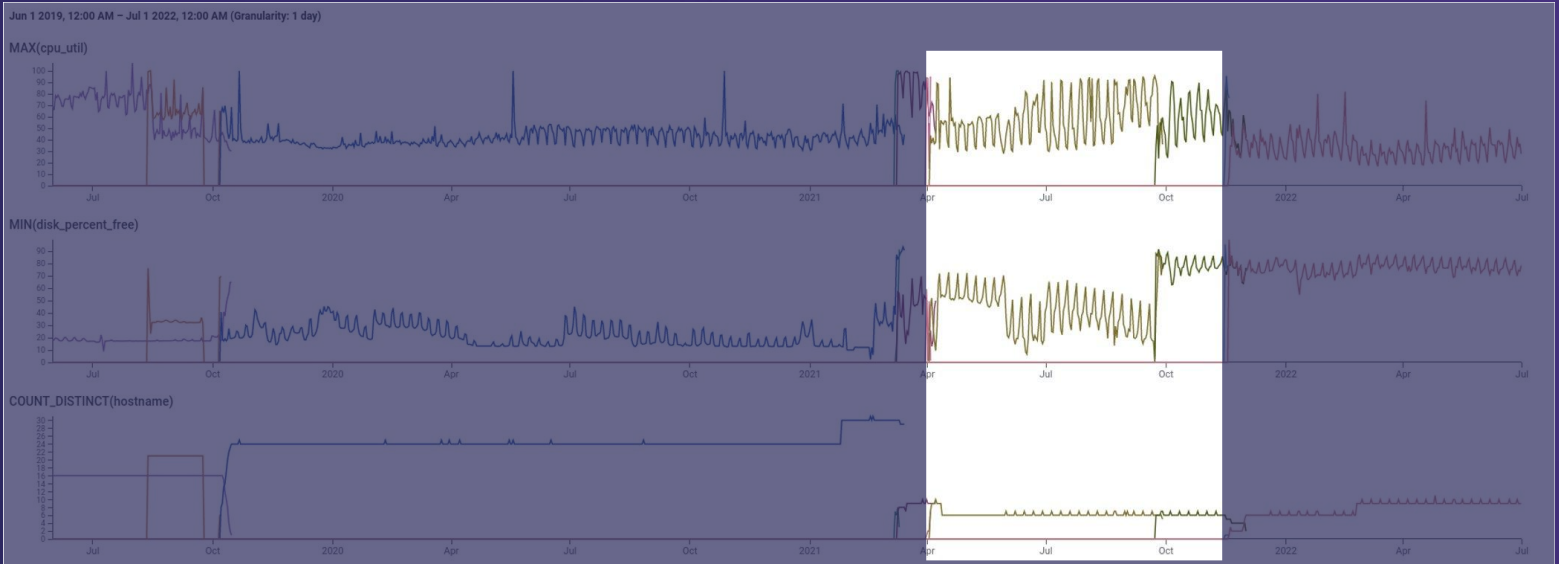
But we were finally able to move forward in Dec 2021 with im4gn!





Read more: go.hny.co/kafka-lessons

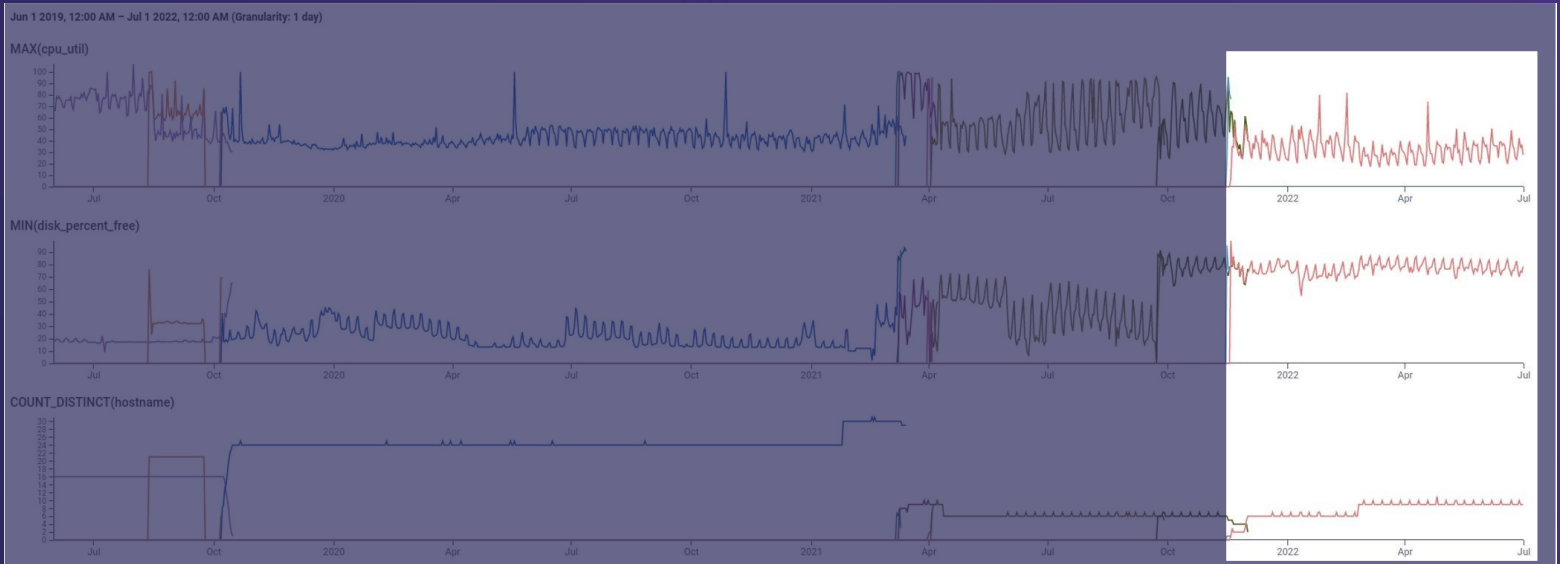
Finding the right way to migrate Kafka



instance_type	MAX(cpu_util)	MIN(disk_percent_free)	COUNT_DISTINCT(hostname)
c5.xlarge	107	8.93199	16
i3en.large	100.1	23.21438	27
i3en.xlarge	100.1	2.41057	43
m6g.large	100.1	0.32752	9
c6gn.xlarge	99.8	1.01842	17
is4gen.2xlarge	95.7	76.5197	1
i3en.2xlarge	95.6	2.8939	29
c6gn.2xlarge	95.1	1.48059	10
i3en.3xlarge	95	62.79987	13
im4gn.4xlarge	81.9	54.73766	43

elapsed query time: 2.050535309s # results: 10 rows examined: 453,669,920 nodes reporting: 100%

Finding the right way to migrate Kafka



instance_type	MAX(cpu_util)	MIN(disk_percent_free)	COUNT_DISTINCT(hostname)
c5.xlarge	107	8.93199	16
i3en.large	100.1	23.21438	27
i3en.xlarge	100.1	2.41057	43
m6g.large	100.1	0.32752	9
c6gn.xlarge	99.8	1.01042	17
is4gen.2xlarge	95.7	76.5197	1
i3en.2xlarge	95.6	2.8939	29
c6gn.2xlarge	95.1	1.48059	10
i3en.3xlarge	95	62.79987	13
im4gn.4xlarge	81.9	54.73766	43

elapsed query time: 2.050535309s # results: 10 rows examined: 453,669,920 nodes reporting: 100%

Take care of your people

Existing incident response practices

- Escalate when you need a break / hand-off
- Remind (or enforce) time off work to make up for off-hours incident response

Official Honeycomb policy

- Incident responders are encouraged to expense meals for themselves and family during an incident

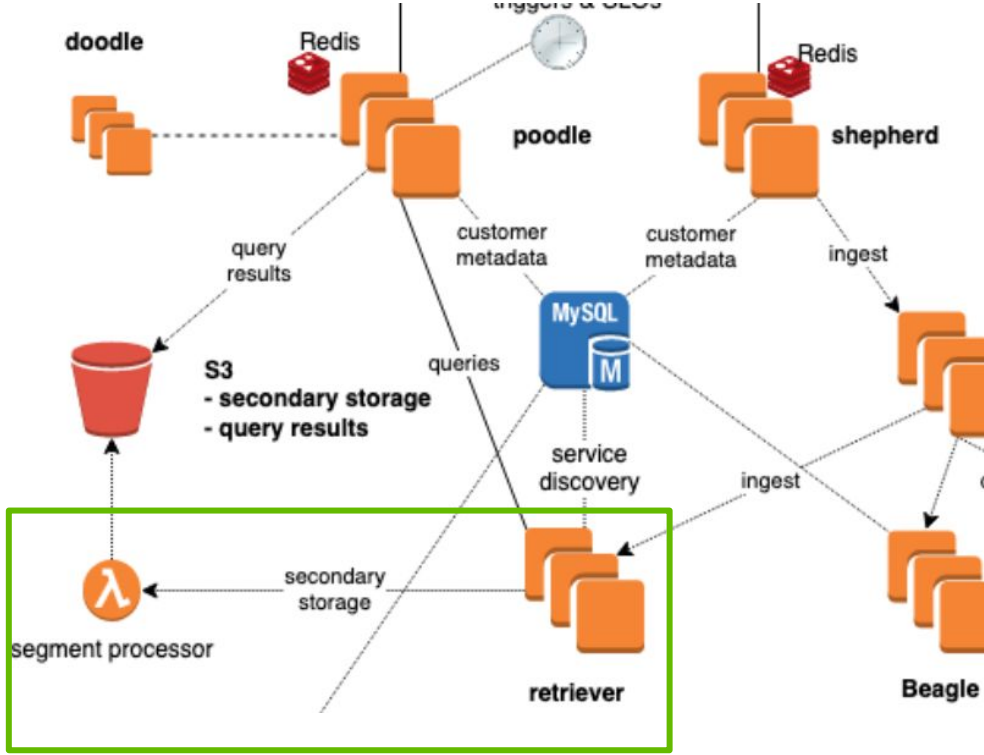
We hire adults.

Pay attention to your mind and body so you can give and get help. All of us wobble, and being transparent about that means we can support each other. Participate fully in collaboration, coaching and management. If any group of us were together in a car on a long road trip, there would be no need for a dividing line in the back seat to keep people from hitting each other.

3) Retriever: query service

Retriever is performance-critical

- It calls to Lambda for parallel compute
- Lambda use exploded.
- Could we address performance & cost?
- Maybe.



If infra is code, we can use CI & flags!

LaunchDarkly APP 6:48 PM

Liz Fong-Jones updated the flag Retriever Lambda ARM Percentage

- Added the variation `1% ARM`

Liz Fong-Jones updated the flag Retriever Lambda ARM Percentage in `Production`

- Changed the default variation from ~~`50% ARM`~~ to `1% ARM`

lizf 6:49 PM

reverting ARM experiment, just keeping a trickle on 1% for validation of non-breakage/dogfooding of the lambda layer on both archs. it was 20% slower at p50 and 100% slower at p99, so we need to roll back.

✅ 1 🤔 1 1

1 reply 17 days ago

COUNT	arch exists name = processSegment	arch

Run a few seconds ago

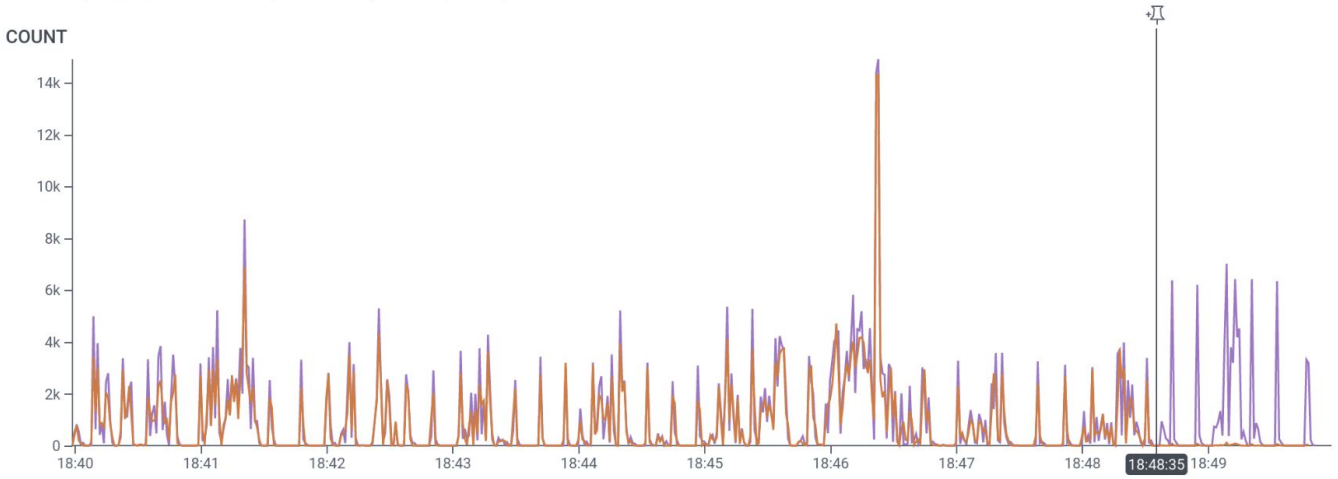
ORDER BY	LIMIT	HAVING
COUNT desc	None	None; include all results

Results | BubbleUp | Metrics | Traces | Raw Data

Compare to 10 minutes prior

Graph Settings

Oct 1 2021, 6:39:59 PM – Oct 1 2021, 6:49:59 PM (Granularity: 1 sec)



arch	COUNT
amd64	583,172
arm64	455,704

Making progress carefully

LaunchDarkly APP 11:06 AM

Liz Fong-Jones turned on the flag Profile Lambda Percent in `Production`

Liz Fong-Jones scheduled changes for the flag Profile Lambda Percent in `Production`

- Changes will occur on `Sat, 16 Oct 2021 18:15:00 UTC`
- Turn off the flag

Liz Fong-Jones scheduled changes for the flag Retriever Lambda ARM Percentage in `Production`

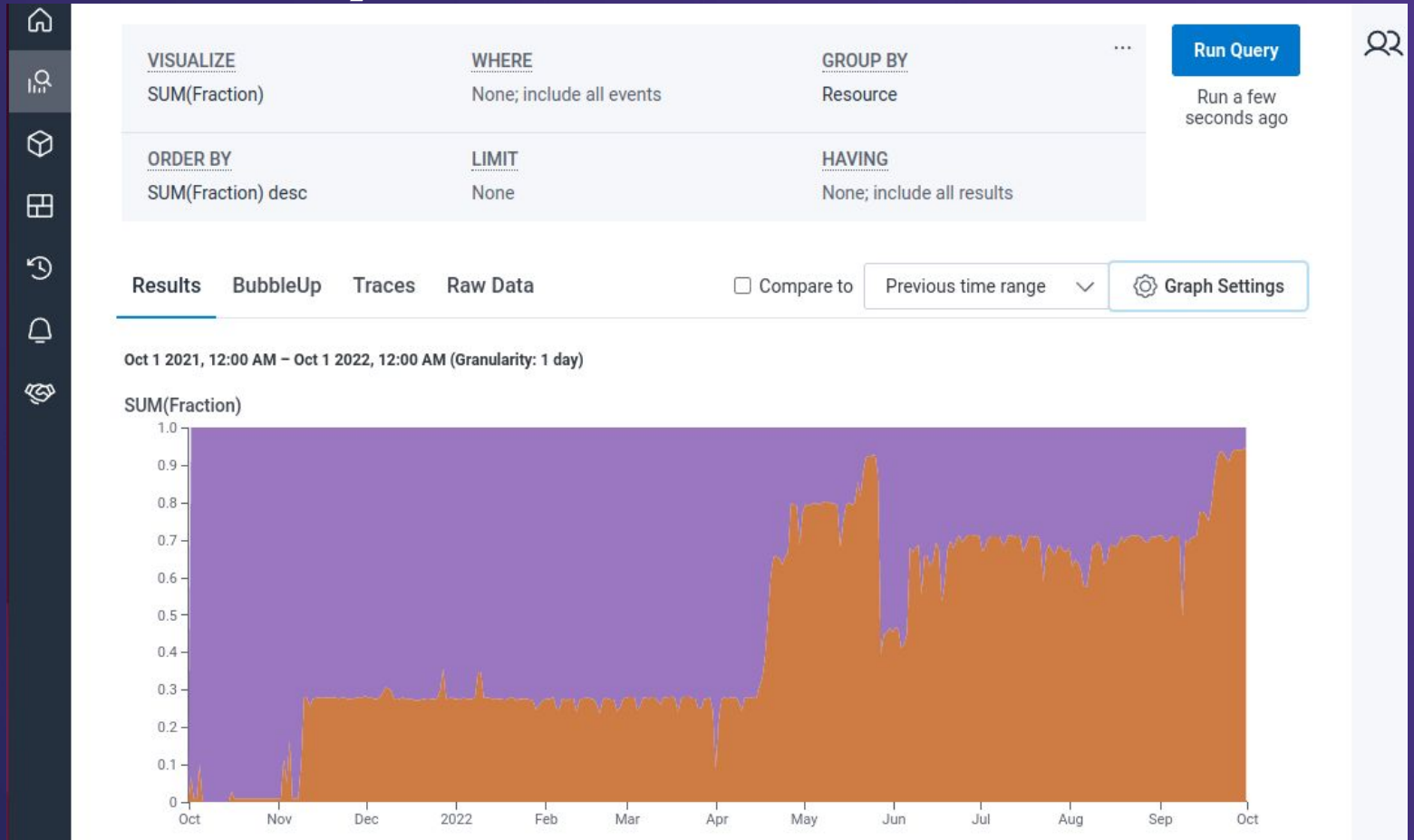
- Changes will occur on `Sat, 16 Oct 2021 18:20:00 UTC`
- Update default variation to **serve** `1% ARM`

LaunchDarkly APP 11:15 AM

Completed scheduled changes to the flag Profile Lambda Percent in `Production` (via API)

- Turned the flag off

Today we're 99%+ ARM lambda



Fast and reliable: pick both!

Go faster, safely.

Takeaways

- Design for reliability through full lifecycle.
- Feature flags can keep us within SLO, most of the time.
- But even when they can't, find other ways to mitigate risk.
- Discovering & spreading out risk improves customer experiences.
- Black swans happen; SLOs are a guideline, not a rule.

Acknowledge hidden risks

Examples of hidden risks

- Operational complexity
- Existing tech debt
- Vendor code and architecture
- Unexpected dependencies
- **SSL certificates**
- **DNS**

Discover early and often through testing.

Takeaways

- We are part of sociotechnical systems. Customers, Engineers, Stakeholders.
- Outages and failed experiments are unscheduled learning opportunities.
- Nothing happens without discussions between different people and teams.
- DevOps is just talking to each other! Figuring out how to put customers first.

PS: we're now 100% Graviton3 k8s nodes.

kubernetes-metrics

Add to Board | Share | Last 60 days

Prod

Add name and description

VISUALIZE	WHERE	GROUP BY
COUNT_DISTINCT(k8s.node.name) HEATMAP(dc.node_cpu_utilization)	k8s.node.name exists label.eks.amazonaws.com/nodegroup does-not-contain retriever label.eks.amazonaws.com/nodegroup does-not-contain stable metrics.source.type = node	instance_family
ORDER BY	**LIMIT**	**HAVING**
COUNT_DISTINCT(k8s.node.name) desc	1000	None; include all results

Run Query

Run a few seconds ago

Results | BubbleUp | Metrics | Traces | Raw Data

Compare to | Previous time range | Graph Settings

Oct 3 2022 22:51:19 – Dec 2 2022 22:51:19 UTC+11:00 (Granularity: 2 hr)

COUNT_DISTINCT(k8s.node.name)

120
100
80
60
40
20
0

Sun Oct 9 | Sun Oct 16 | Sun Oct 23 | Sun Oct 30 | Sun Nov 6 | Sun Nov 13 | Sun Nov 20 | Sun Nov 27

HEATMAP(dc.node_cpu_utilization)

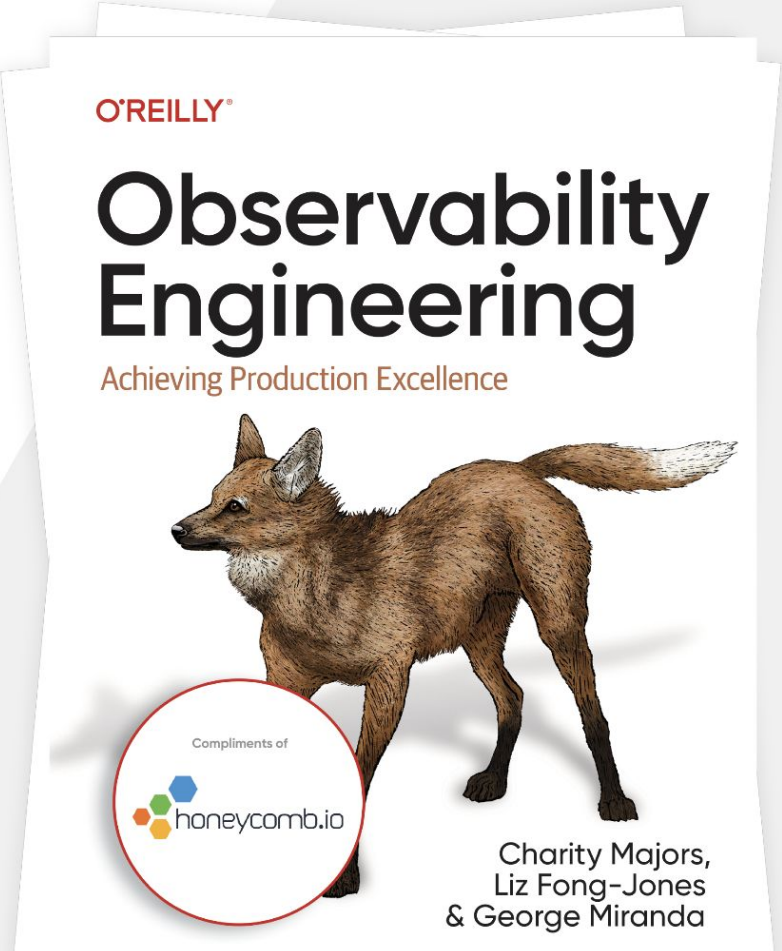
200
150
100
50
0

24000
12000
1

Sun Oct 9 | Sun Oct 16 | Sun Oct 23 | Sun Oct 30 | Sun Nov 6 | Sun Nov 13 | Sun Nov 20 | Sun Nov 27

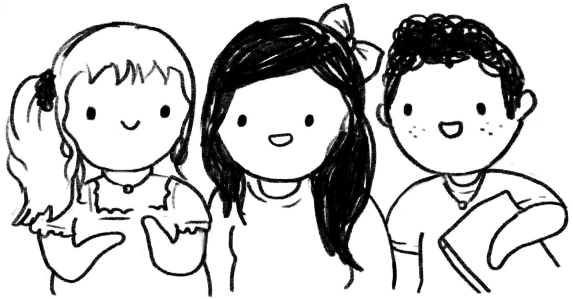
Observability Engineering

Get our new book, free!



@lizthegrey

Understand & control production.



Go faster on stable infra.
Manage risk and iterate.

honeycomb.io/liz

lizthegrey.com; @lizthegrey

www.honeycomb.io