An Outage
How Broken is “Too Broken”? 

What do you do when things break?

How bad was this break?
WE WERE DOWN FOR TWELVE MINUTES

TWELVE MINUTES ISN'T VERY LONG

THAT'S TWO YEARS IN INTERNET TIME
Build new features!

We need to improve quality!
In search of a common language

Management

Engineering

Clients and Users

How broken is “too broken”?  

What does “good enough” mean?  

Combatting alert fatigue
How do we do it?

A telemetry system produces events that correspond to real world use.

We can describe some of these events as eligible.

We can describe some of them as good.
SLI: Service Level Indicator

Given an event, is it eligible? Is it good?

Eligible: “Had an http status code”

Good: “… that was a 200, and was served under 500 ms”
Defining Quality
Defining Quality

good events

eligible events
SLO

Minimum **Quality ratio** over a period of time

Error Budget

Number of bad events allowed.
Left over budget

Deploy faster

Room for experimentation

Opportunity to tighten SLO
## Honeycomb SLOs

<table>
<thead>
<tr>
<th>Description</th>
<th>99.99%</th>
<th>99.9%</th>
<th>99%</th>
</tr>
</thead>
<tbody>
<tr>
<td>We <em>always</em> store incoming user data</td>
<td>~4.3 minutes</td>
<td>45 minutes</td>
<td>7.3 hours</td>
</tr>
<tr>
<td>Default dashboards <em>usually</em> load in &lt; 1s</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Queries <em>often</em> return in &lt; 10 s</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
We blew through **three months’ budget** in those 12 minutes.
Ingest Outage

We dropped customer data
Ingest Outage

We dropped customer data
We rolled it back (manually)
We communicated to customers
We halted deploys
What happened?

We checked in code that **didn’t build**.

We had **experimental** CI build wiring.

Our scripts deployed **empty binaries**.

There was **no health check** and rollback.
How we fixed it

We stopped writing new features

We prioritized stability

We mitigated risks
SLOs allowed us to characterize
what went wrong,
how badly it went wrong,
and
how to prioritize repair
This Talk

- Design Thinking
-Expressing and Viewing
- Burndown Alerts and Responding
- Learning from our Experiences
- Success Stories
Design Thinking and Task Analysis

Understand user goals and needs
Learn from informants and experts
Collaborate with internal team
Collect feedback and ideas externally
Displays and Views
See where the burndown was happening, explain why, and remediate.
Expressing SLOs

Event based

“How many events had a duration < 500 ms”

Time based

“How many 5 minute periods, had a P95(duration) < 500 ms”
How do we express SLOs?

Good events
Bad events
How often
Time range
How do we express SLOs?

- Good events
- Bad events
- How often
- Time range
How do we express SLOs?

Good events

Bad events

How often

Time range

Eligible: $name is “run_trigger_detailed”

Good: $app.error does not exist

```
IF(EQUALS($name, "run_trigger_detailed"), NOT(EXISTS($app.error)))
```
How do we express SLOs?

- Good events
- Bad events
- How often
- Time range
How do we express SLOs?

- Good events
- Bad events
- How often
- Time range

Update SLO

Learn more about creating SLOs

Name
Basset triggers query retriever successfully

Description
few errors in run_trigger_detailed
sli_errors field looks at app.error.

SLI Column
sli_errors

IF(EQUALS($name, "run_trigger_detailed"), NOT(EXISTS($app.error)))

Time Period (in days)
7

Target Percentage
99.99
Status of an SLO

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

57.5%
How have we done?

Historical SLO Compliance

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

95% of eligible events from the user-events column 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.

46.7%

Historical SLO Compliance

For each day of the past 30, how often this SLI has succeeded over the preceding 30 days.
Where did it go?

99.5% of eligible events from the `poodle` column in the `sli_on_home_page_load_noerr` 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.
When did the errors happen?
When did the errors happen?

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.

Distribution of Events failing SLI by duration_ms
Feb 13 2020, 10:06 PM – Feb 14 2020, 10:06 PM
What went wrong?

High dimensional data

High cardinality data
Dimensions
Try to GROUP BY columns that look most different between the successful and failed.

Measures
Try asking WHERE the successful and failed are most different.
Why did it happen?
See where the burndown was happening, explain why, and remediate
User Feedback

“The Bubble Up in the SLO page is really powerful at highlighting what is contributing the most to missing our SLIs, it has definitely confirmed our assumptions.”
“Your customers have to be happy... we have to have an understanding of the customer experience. ... **To the millisecond we knew what our percentage was of success versus failure.**”

-Josh Hull, Site Reliability Engineering Lead, Clover Health
“The historical SLO chart also **confirms a fix** for a performance issue we did greatly contributed to the SLO compliance by showing a nice upward trend line. :)

User Feedback
User Feedback

“I’d love to drive alerts off our SLOs. Right now we don’t have anything to draw us in and have some alerts on the average error rate but they’re a little spiky to be useful. It would be great to get a better sense of when the budget is going and define alerts that way.”
Burndown Alerts
How is my system doing?

Am I over budget?

When will my alarm fail?
When will I fail?

User goal: get alerts to exhaustion time

Human-digestible units

24 hours: “I’ll take a look in the morning”

4 hours: “All hands on deck!”
How is my system doing?
Am I over budget?
When will my alarm fail?
Implementing Burn Alerts

Run a 30 day query

at a 5 minute resolution

every minute
**SQL Query**

```sql
SELECT SUM(amortizedCost) AS SUM_amortizedCost, resourceTags/user:Role, product/productFamily
FROM aws_costusage
GROUP BY resourceTags/user:Role, product/productFamily
ORDER BY SUM(amortizedCost) DESC
LIMIT None
```

### Results

<table>
<thead>
<tr>
<th>resourceTags/user:Role</th>
<th>product/productFamily</th>
<th>SUM(amortizedCost)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Data Transfer</td>
<td>9,744.61691</td>
</tr>
<tr>
<td>kafka</td>
<td>Compute Instance</td>
<td>6,492.72813</td>
</tr>
<tr>
<td>retriever</td>
<td>Storage</td>
<td>2,374.18905</td>
</tr>
<tr>
<td>kafka</td>
<td></td>
<td>1,991.24955</td>
</tr>
</tbody>
</table>

---

Danyel Fisher  @fisherdanyel
Caching is Fun!
Fun with Caching

Vital to cache results

... but not incomplete results

... ... at what resolution of cache?
Flappy Alerts

“It’ll expire at 3:55”

“Wait, make that 4:05”

“Nope, 3:55 again!”

(We added a 10%ish buffer)
Recovering from Bankruptcy

A failure a month ago brought us to -169% and still hasn’t aged out?

That means we don’t get alerts anymore

Customer workaround: delete and re-create the SLO, thus blowing the cache
Learning from Experience
Volume is important

Tolerate at least dozens of bad events per day
Faults

```go
err = zReader.Reset(bodyReader)
if err != nil {
    beeline.AddField(ctx, "dropped", "our fault")
    beeline.AddField(ctx, "drop_reason", err.Error())
    return 0, err
}
```

```go
info, partitions, err := a.getReqInfo(r, reqinfo.ResourceEvents, a.prepPartitionFn(ctx))
if err != nil {
    if errors.Is(err, reqinfo.ErrDisabledWriteKey) {
        beeline.AddField(ctx, "dropped", "their fault")
        beeline.AddField(ctx, "drop_reason", "disabled write key")
        return apierr.MsgUnknownTeam
    }
    return err
}
```
SLOs for Customer Service
Blackouts are easy

... but brownouts are much more interesting
Triggered: **Shepherd ALB timeout/error** will violate SLO in 4h0m0s

**Status**
The error budget for SLO **Shepherd ALB timeout/error** is in danger of being exhausted.
Description: `99.995% of ALB response codes should match what Shepherd gave us.
If these differ, it's either due to client misbehavior or due to Shepherd failing to respond.`

View SLO
Timeline

1:29 am

SLO alerts. “Maybe it’s just a blip”
1.5% brownout for 20 minutes
Timeline

1:29 am: SLO alerts. “Maybe it’s just a blip”

4:21 am: Minor incident. “It might be an AWS problem”

6:25 am: SLO alerts again. “Could it be ALB compat?”

9:55 am: “Why is our system uptime dropping to zero?”

It’s out of memory
We aren’t alerting on that crash

COUNT DISTINCT(hostname)

0 5 10 15 20 25 30 35 40
05:00 06:00 07:00 08:00 09:00 10:00 11:00 12:00 UTC

Danyel Fisher  @fisherdanyel
Timeline

1:29 am  SLO alerts. “Maybe it’s just a blip”
         1.5% brownout for 20 minutes
4:21 am  Minor incident. “It might be an AWS problem”
Timeline

1:29 am  SLO alerts. “Maybe it’s just a blip”
1.5% brownout for 20 minutes

4:21 am  Minor incident. “It might be an AWS problem”

6:25 am  SLO alerts again. “Could it be ALB compat?”
Timeline

1:29 am  SLO alerts. “Maybe it’s just a blip”
        1.5% brownout for 20 minutes
4:21 am  Minor incident. “It might be an AWS problem”
6:25 am  SLO alerts again. “Could it be ALB compat?”
9:55 am  “Why is our system uptime dropping to zero?”
        It’s out of memory
        We aren’t alerting on that crash
1:29 am
SLO alerts. “Maybe it’s just a blip”

4:21 am
Minor incident. “It might be an AWS problem”

6:25 am
SLO alerts again. “Could it be ALB compat?”

9:55 am
“Why is our system uptime dropping to zero?”

It’s out of memory
We aren’t alerting on that crash
Timeline

1:29 am  SLO alerts. “Maybe it’s just a blip”
         1.5% brownout for 20 minutes
4:21 am  Minor incident. “It might be an AWS problem”
6:25 am  SLO alerts again. “Could it be ALB compat?”
9:55 am  “Why is our system uptime dropping to zero?”
         It’s out of memory
         We aren’t alerting on that crash
10:32 am Fixed
How we fixed it

We stopped writing new features

We prioritized stability

We mitigated risks

... and we promoted our SLO burn alerts
Cultural Change

It’s hard to replace alerts with SLOs

But a clear incident can help
Reduce Alarm Fatigue

Focus on **user-affecting** SLOs

Focus on **actionable** alarms
Conclusion
SLOs allowed us to characterize what went wrong, how badly it went wrong, and how to prioritize repair.
You can do it too
And maybe avoid our mistakes
Pitfalls in Measuring SLOs

Email:  danyel@honeycomb.io
Twitter: @fisherdanyel

Visit our booth on the 5th floor to kick the SLO tires, and to learn how we debug in high-res

hny.co/danyel