Detangling complex systems

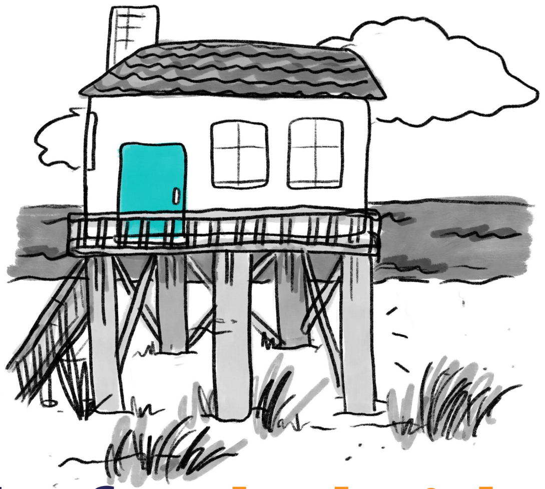
with compassion & production excellence

Liz Fong-Jones
@lizthegrey
#DevOpsDays MSP
August 6, 2019

honeycomb.io

w/ illustrations by @emilywithcurls!

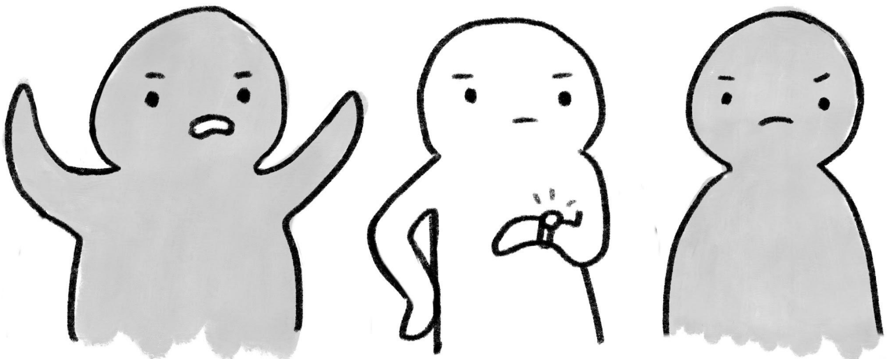
Production is increasingly complex.



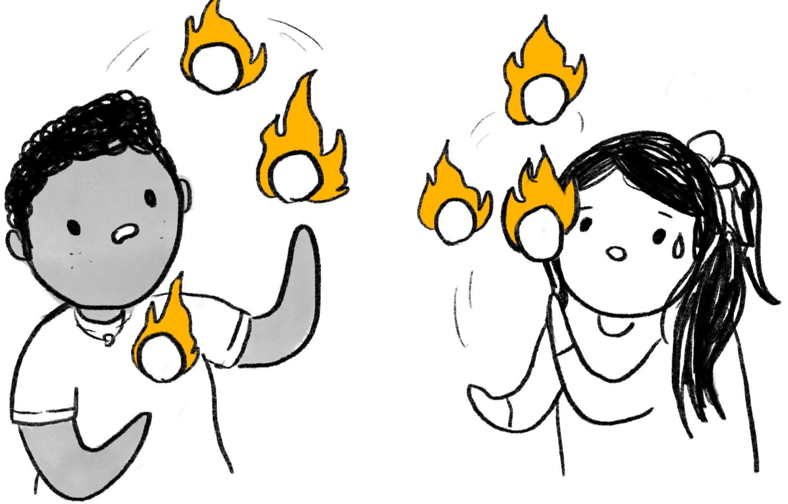
Especially for hybrid systems.

What does uptime mean?

Is it measured in servers?

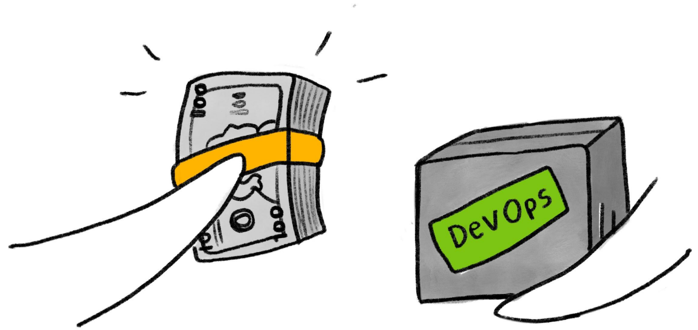


Is it measured in complaints?

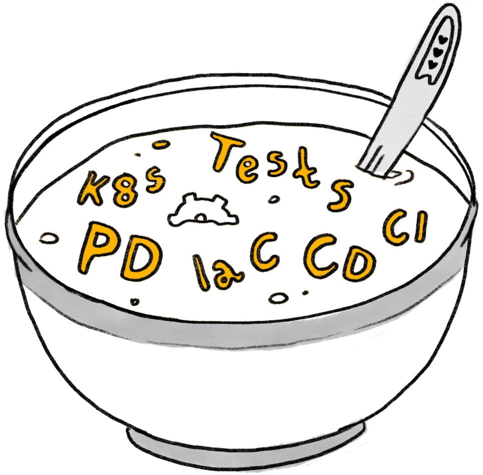


How about juggling everything else?

Our strategies need to evolve.



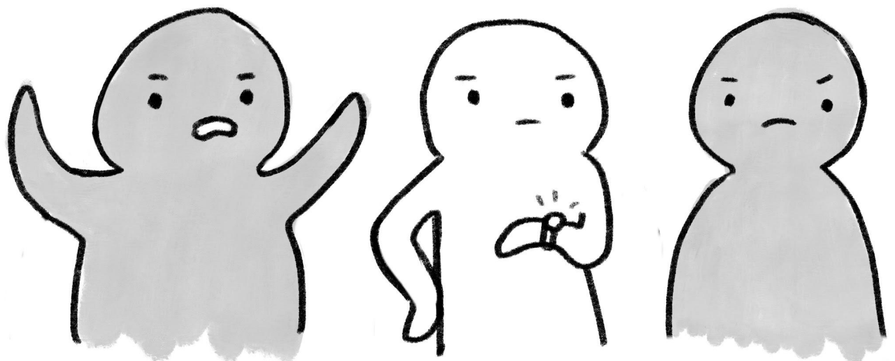
Don't "buy" DevOps.



When we order the alphabet soup...

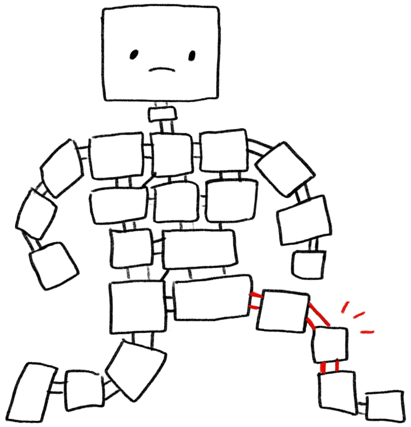
Noisy alerts. Grumpy engineers.

Walls of meaningless dashboards.

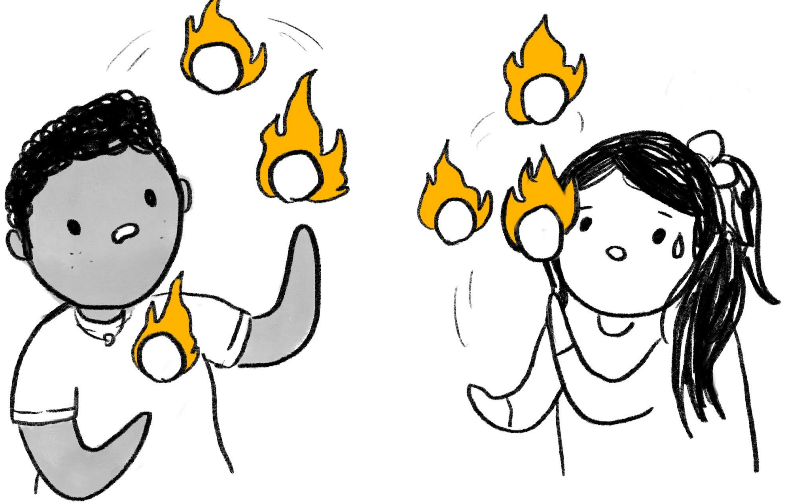


Incidents take forever to fix.

Everyone bugs the "expert".



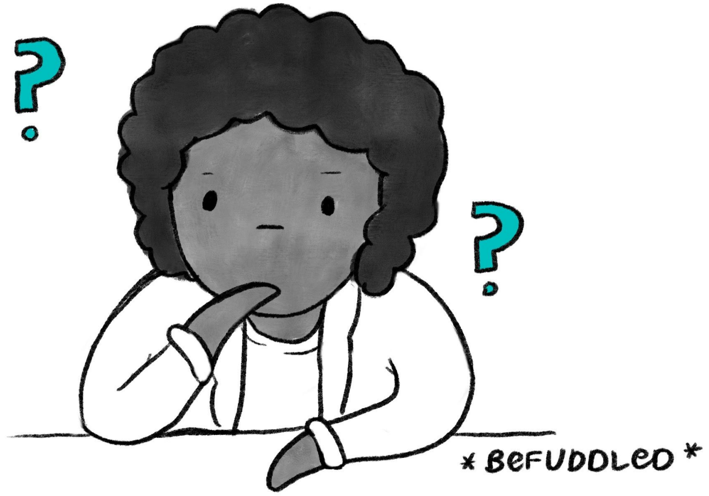
Deploys are unpredictable.



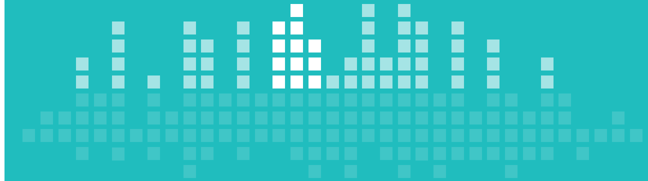
There's no time to do projects...

and when there's time, there's no plan.

The team is struggling to hold on.



What are we missing?

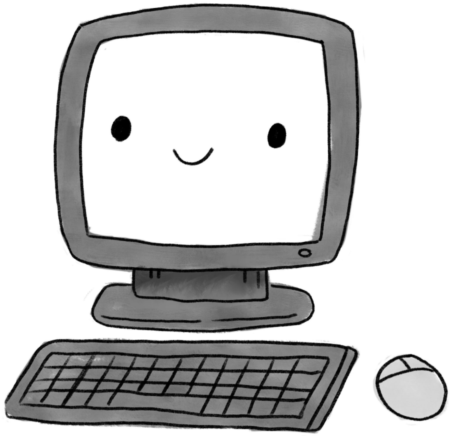


We forgot who operates systems.

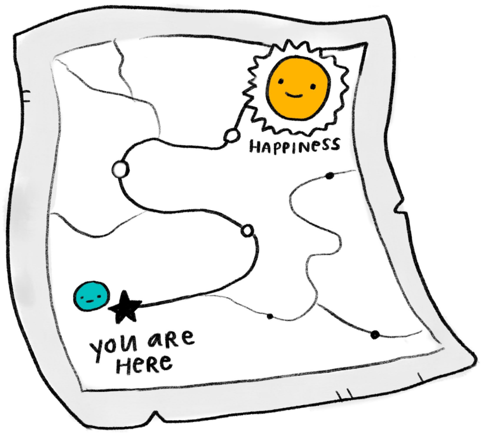
Tools aren't magical.

Invest in people, culture, & process.

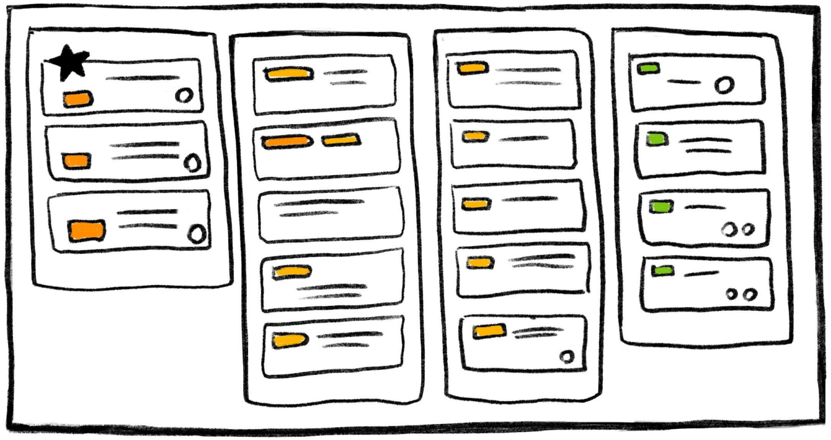
Enter the art of **Production Excellence.**



Make systems more reliable & friendly.



ProdEx takes planning.

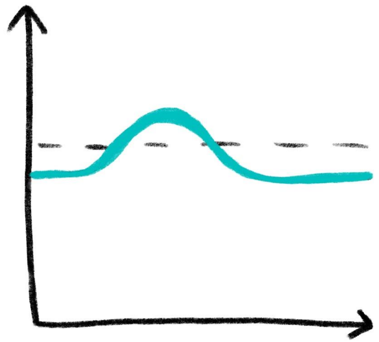


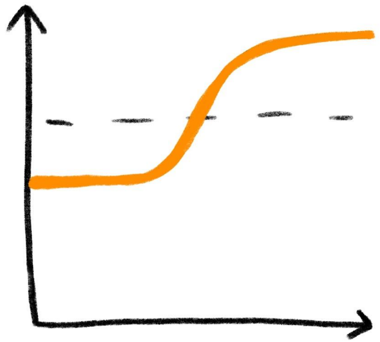
Measure and act on what matters.

Involve everyone.

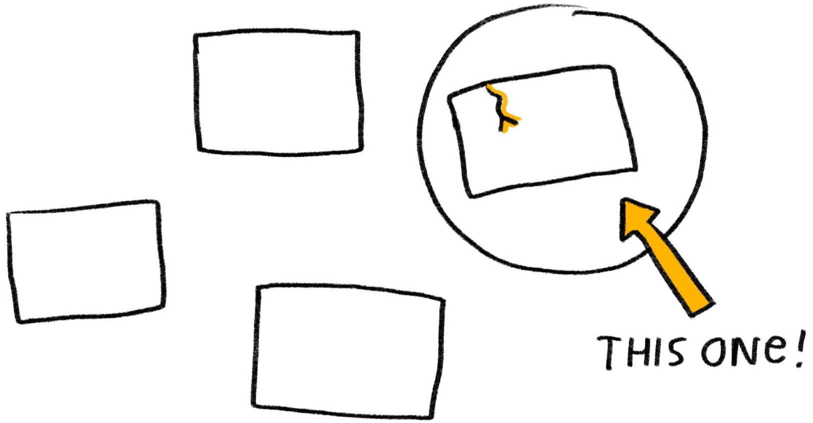
Build everyone's confidence. Encourage asking questions.

How do we get started?





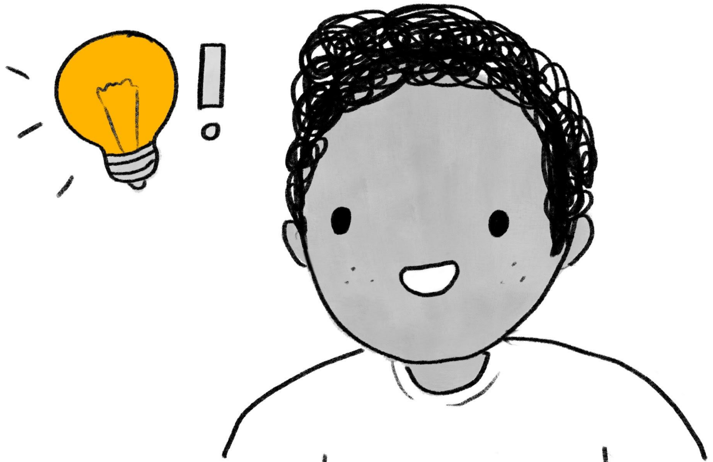
Know when it's too broken.



& be able to debug, together when it is.

Eliminate (unnecessary) complexity.

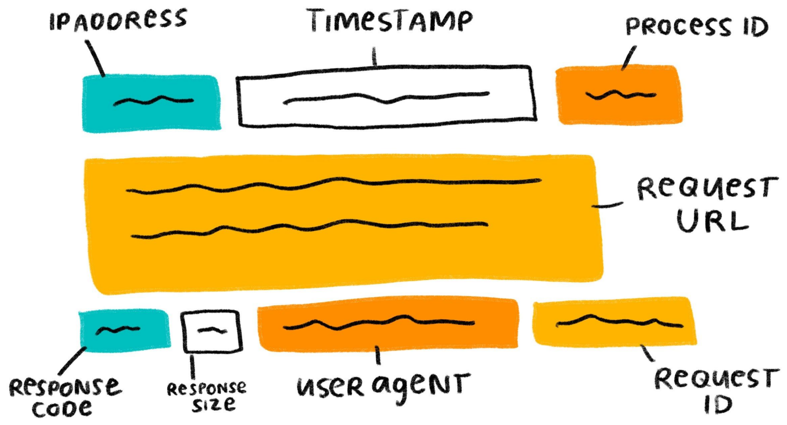
Our systems are always failing.



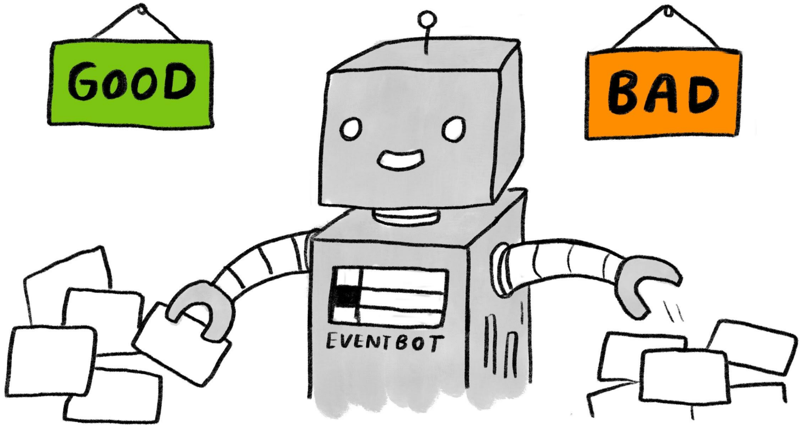
What if we measure too broken?

We need Service Level Indicators

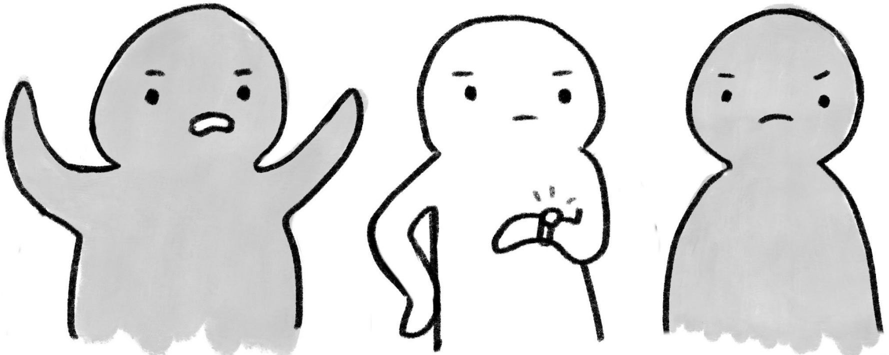
SLIs and SLOs are common language.



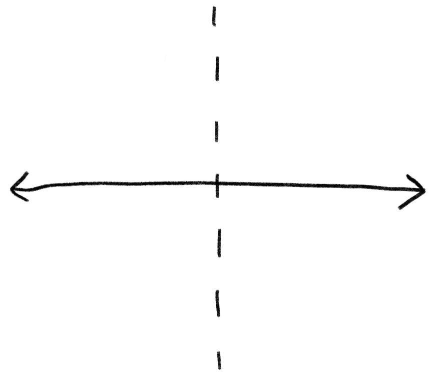
Think in terms of events in context.



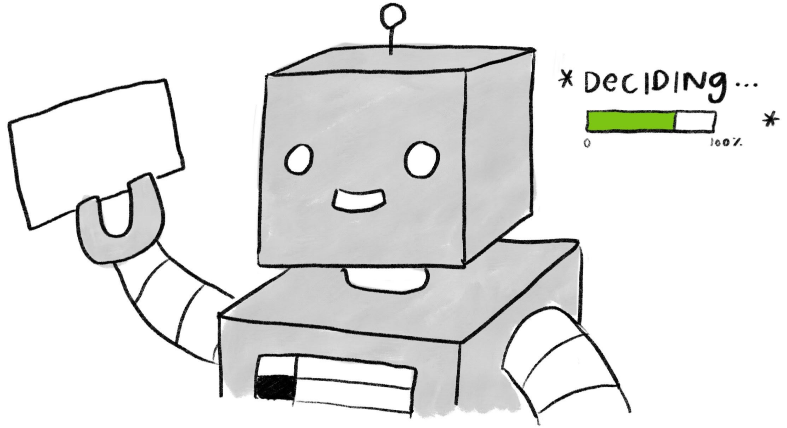
Is this event good or bad?



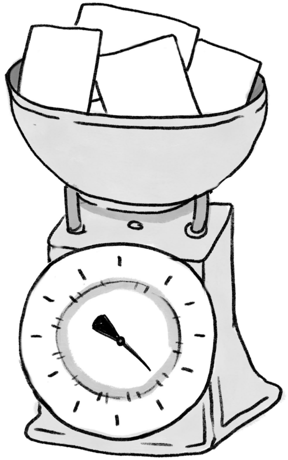
Are users grumpy? Ask your PM.



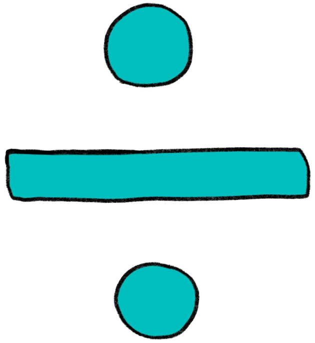
What threshold buckets events?



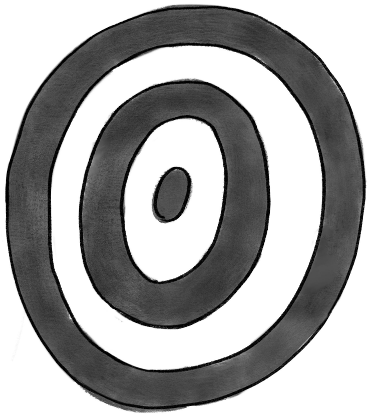
HTTP Code 200? Latency < 300ms?



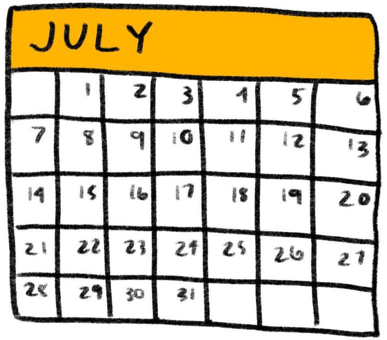
How many eligible events did we see?

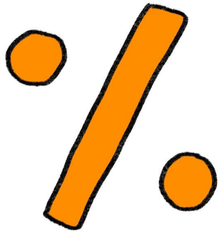


Availability: Good / Eligible Events



Set a target Service Level Objective.





Use a window and target percentage.

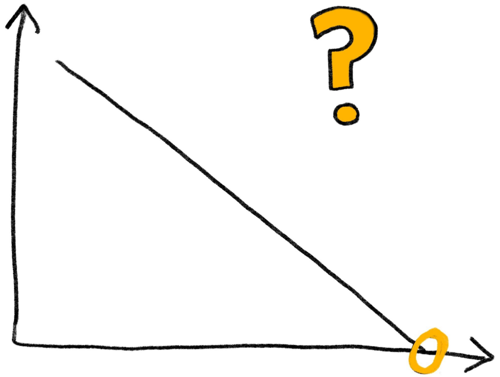
99.9% of events good in past 30 days.

A good SLO barely keeps users happy.

Drive alerting with SLOs.

$$1 - x$$

Error budget: allowed unavailability



How long until I run out?

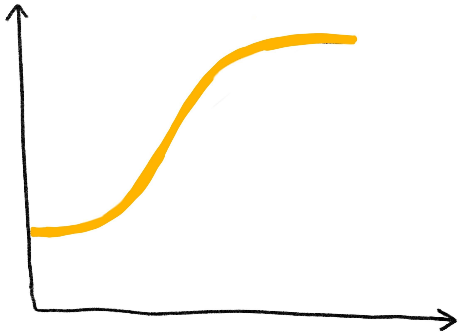
Page if it's hours.

Ticket if it's days.

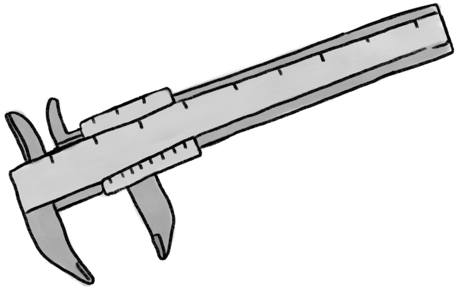
Data-driven business decisions.

Is it safe to do this risky experiment?

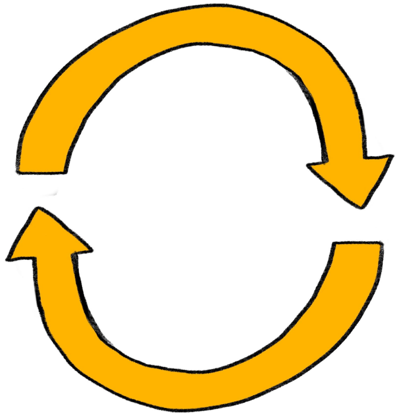
Should we invest in more reliability?



Perfect SLO > Good SLO >>> No SLO



Measure what you can today.

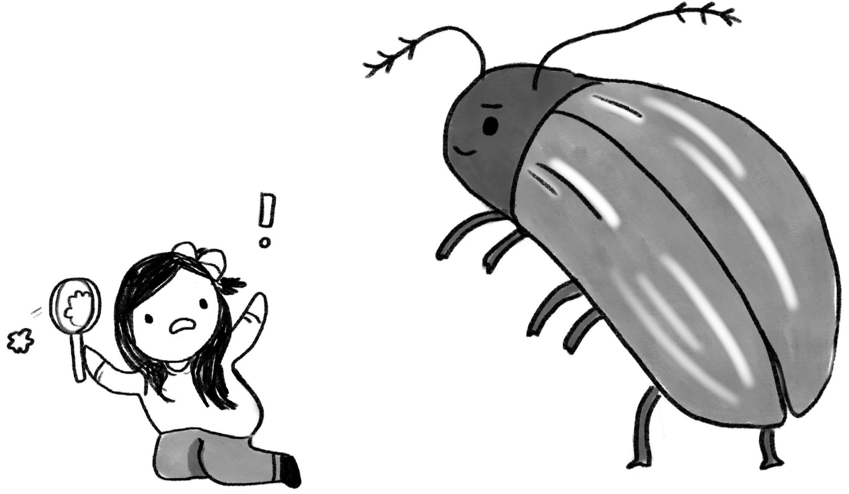


Iterate to meet user needs.

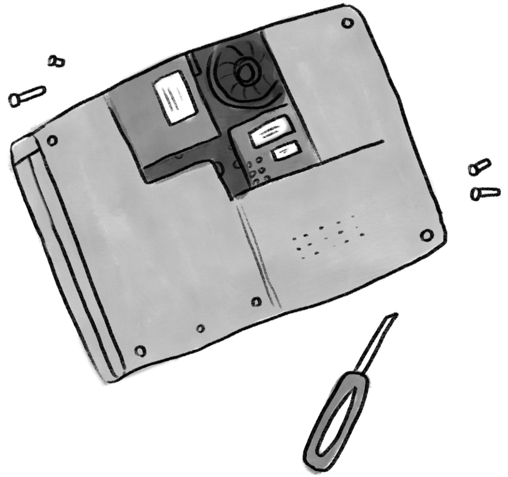
Only alert on what matters.

SLIs & SLOs are only half the picture...

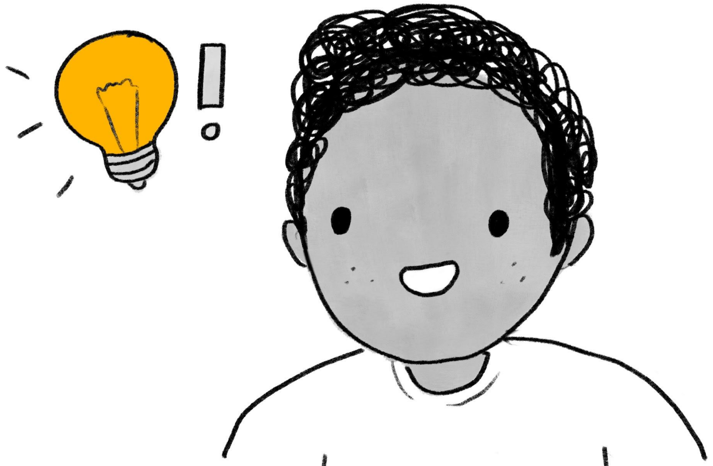
Our outages are never identical.



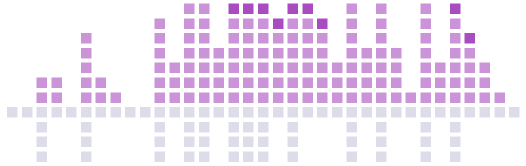
Failure modes can't be predicted.

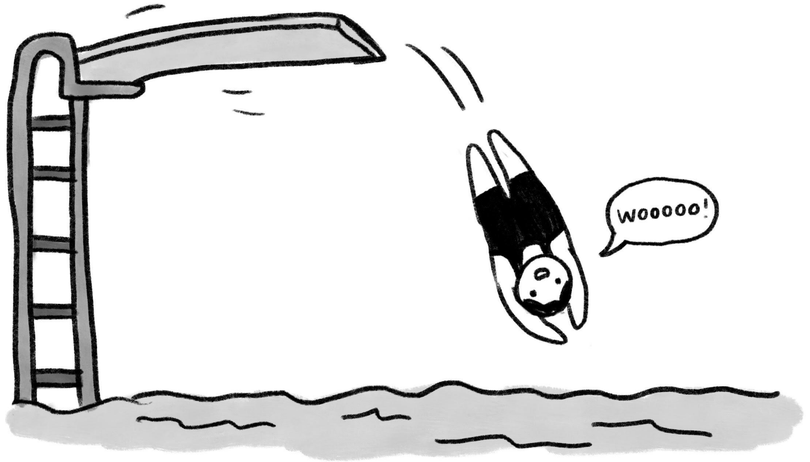


Support debugging novel cases.
In production.

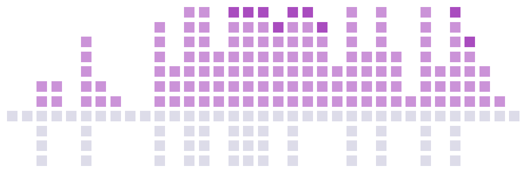


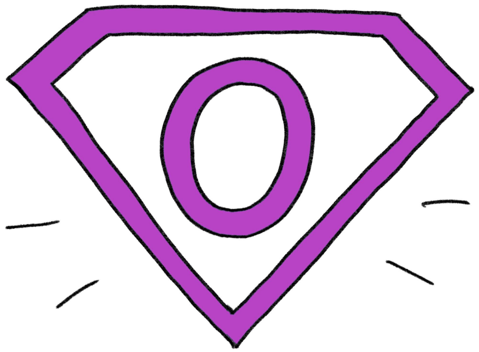
Allow forming & testing hypotheses.



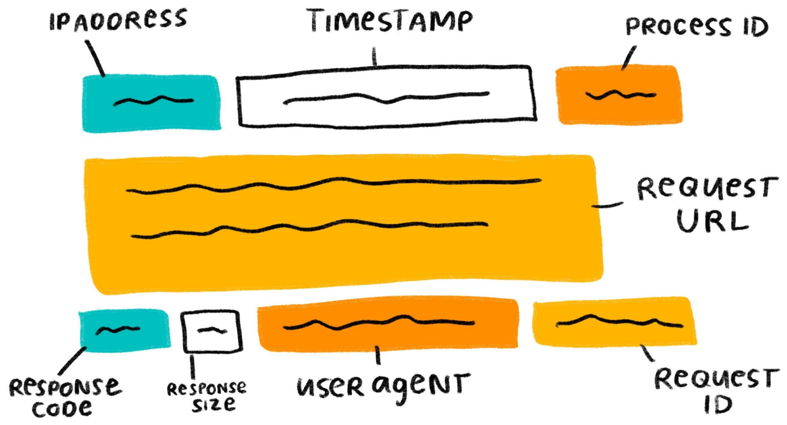


Dive into data to ask new questions.

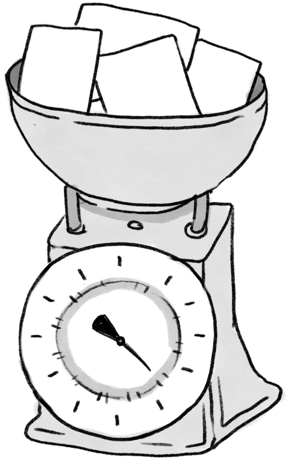




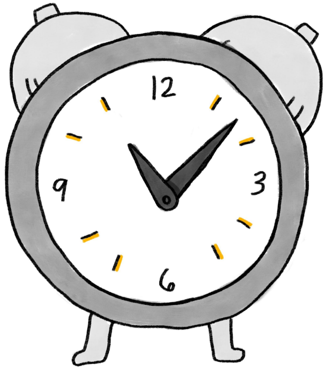
Our services must be observable.



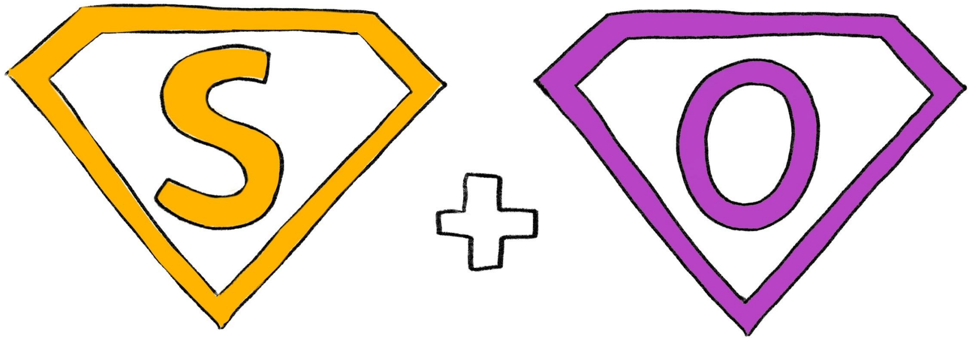
Can you examine events in context?



Can you *explain* the variance?



Can you mitigate impact & debug later?

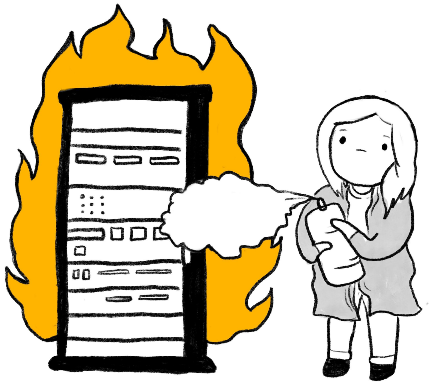


SLOs and Observability go together.

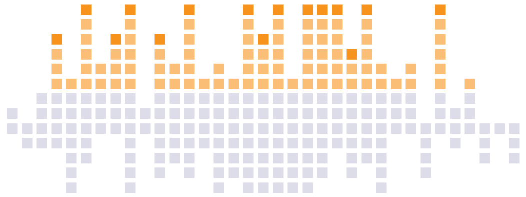
But they alone don't create collaboration.

Heroism isn't sustainable.

Debugging is not a solo activity.

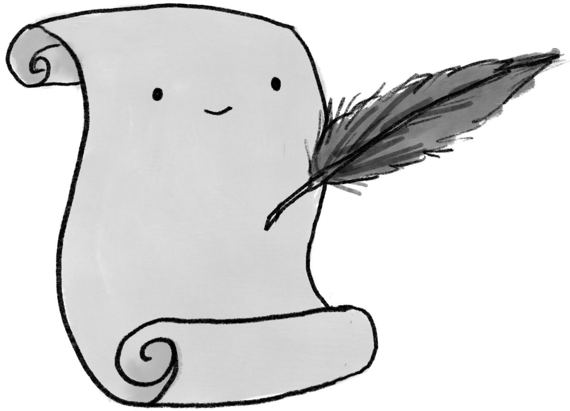


Debugging is for everyone.



Collaboration is interpersonal.

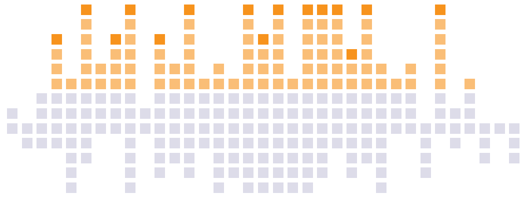
Lean on your team.

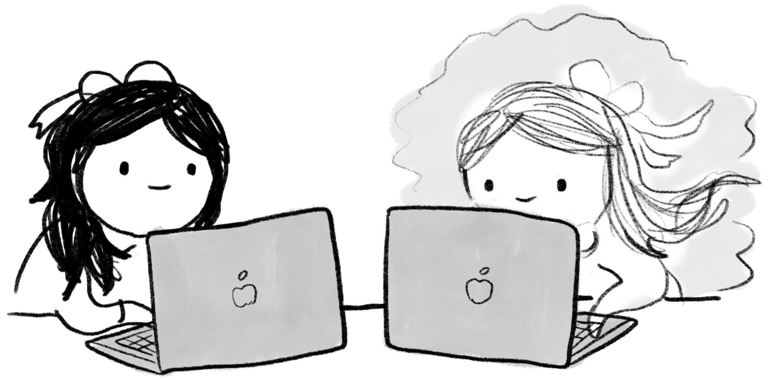


We learn better when we document.

Fix hero culture. Share knowledge.

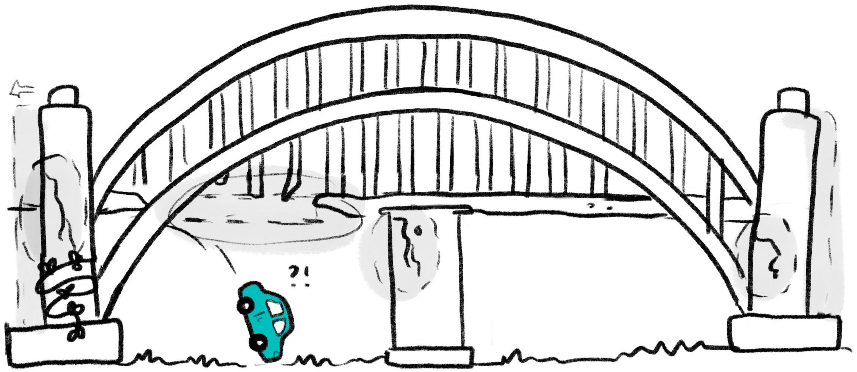
Reward curiosity and teamwork.



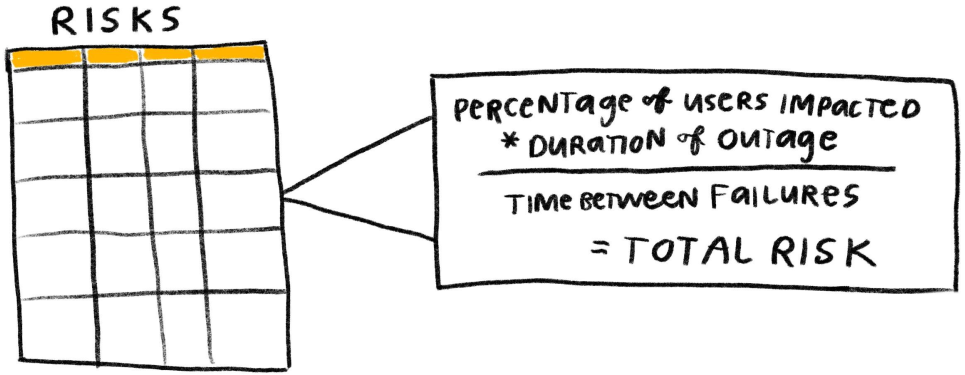


Learn from the past.
Reward your future self.

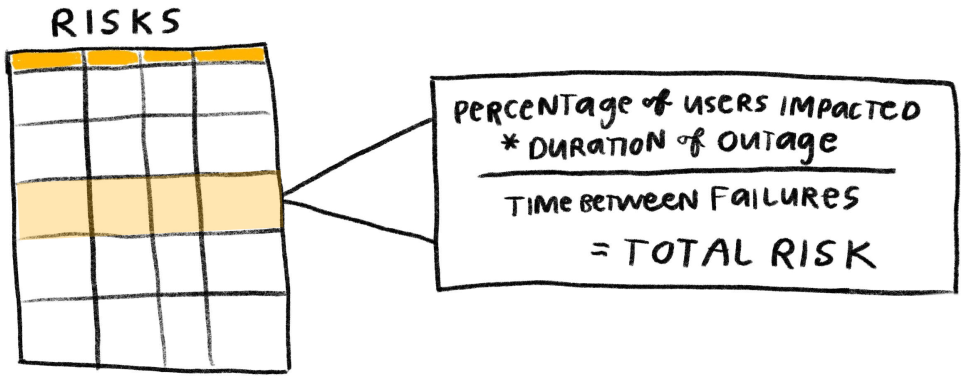
Outages don't repeat, but they rhyme.



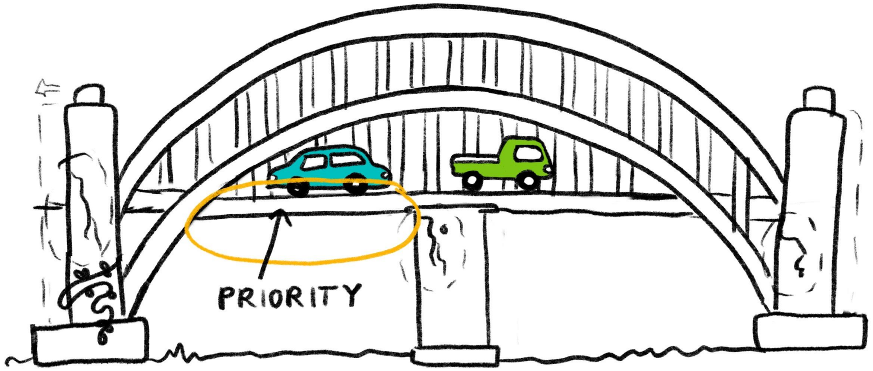
Risk analysis helps us plan.



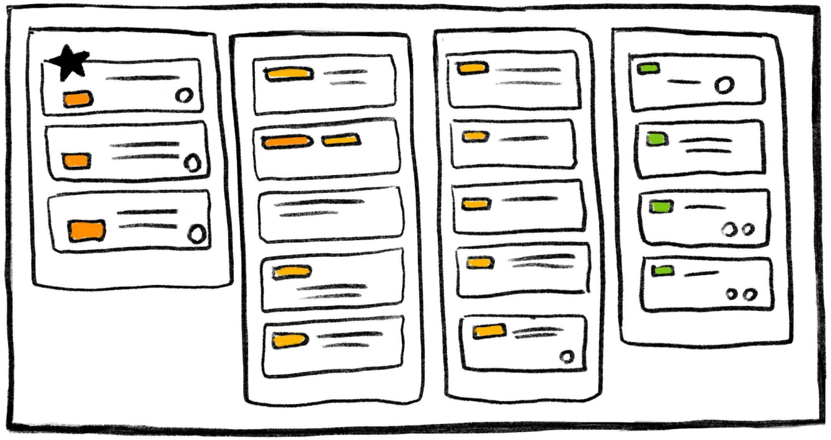
Quantify risks by frequency & impact.



Which risks are most significant?

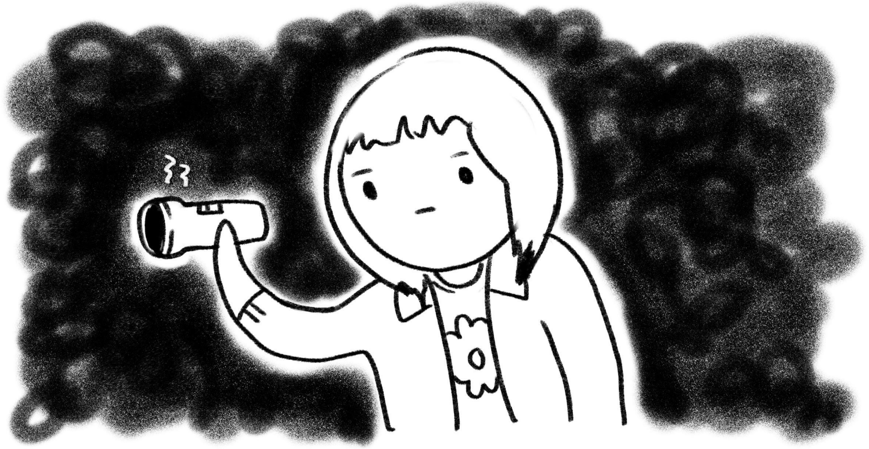


Address risks that threaten the SLO.



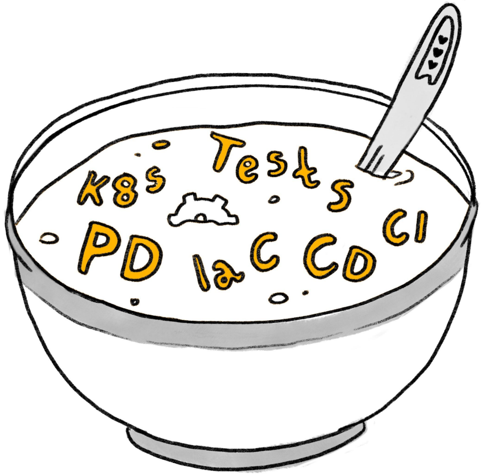
Make the business case to fix them.

And prioritize completing the work.

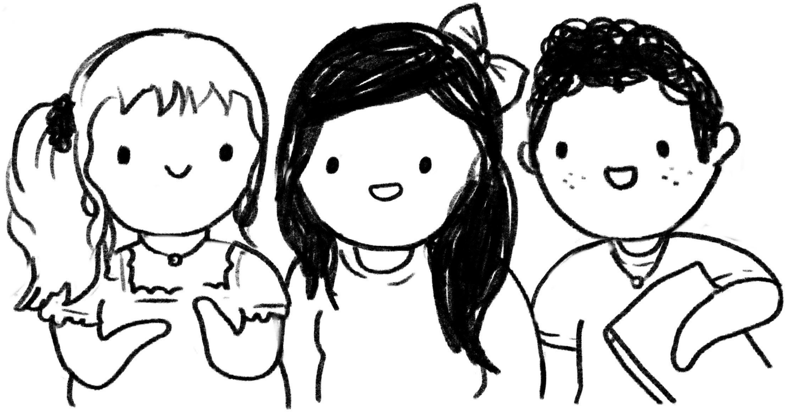


Lack of observability is systemic risk.

So is lack of collaboration.



Season the alphabet soup with ProdEx



Production Excellence brings teams closer together.

Measure. Debug. Collaborate. Fix.

lizthegrey.com; @lizthegrey

honeycomb.io