Operationalizing DORA metrics:

From theory to practice
CTO Summit SF 2022

Emily Nakashima

VP Engineering @ Honeycomb.io

@eanakashima

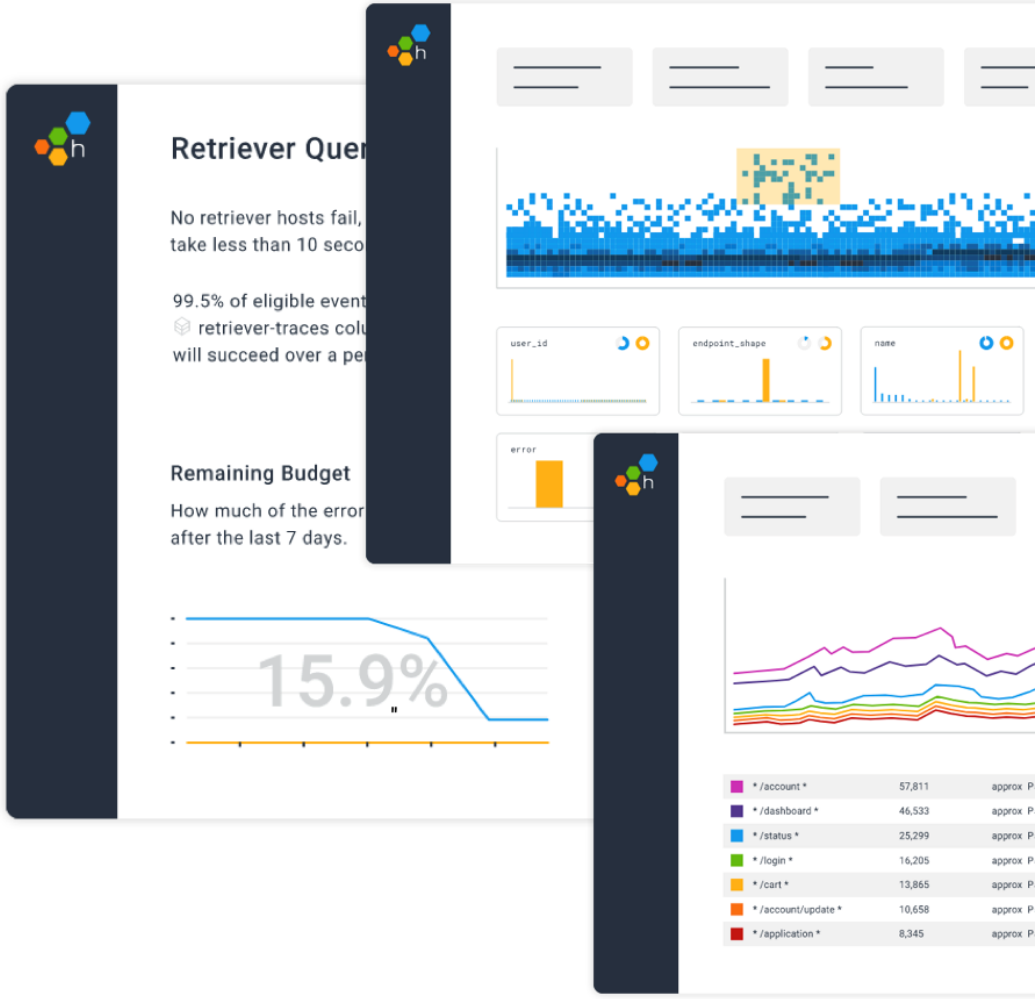
@eanakashima@hachyderm.io

About where I work

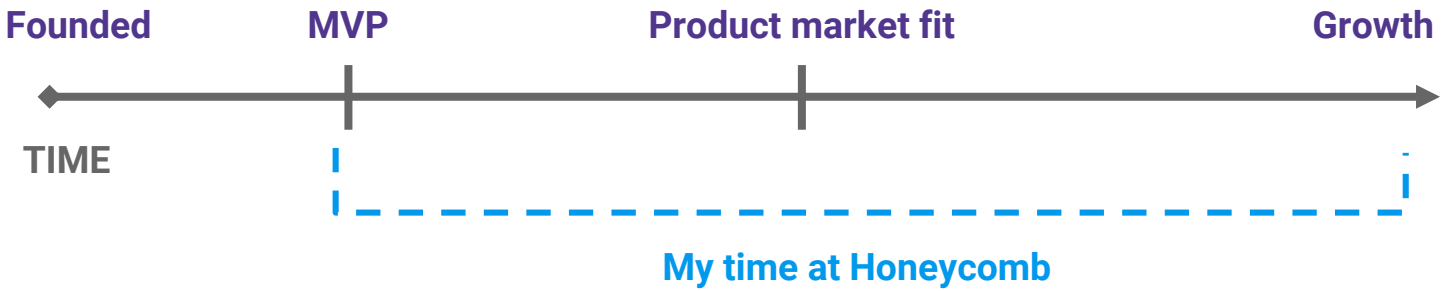
Honeycomb is a realtime SaaS observability product for software systems.

TL;DR: for this talk:

- Startup (150 people)
- Highly technical product
- Distributed team



About where I work



* Not to scale

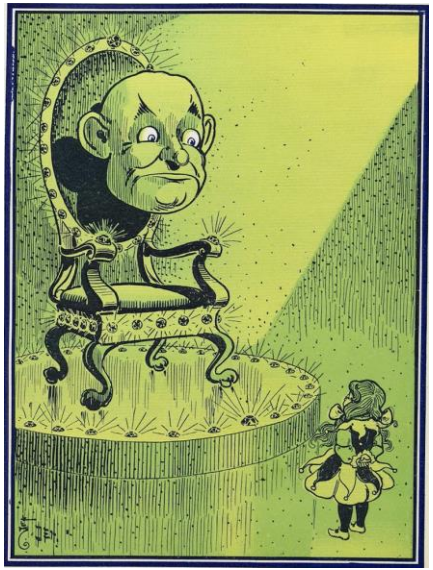
Three reasons to look at quantifying work now

- Switch to distributed work
- Company hit new size threshold (150 people)
- Macroeconomic changes
  - VC-backed companies now worry more about efficiency

Illustrations are by W. W. Denslow from the Wonderful Wizard of Oz (1900) and are in the public domain

The problem

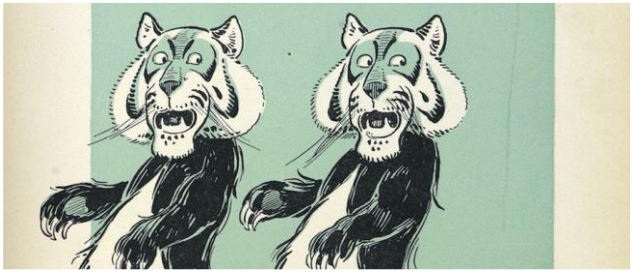
> **You say your team is delivering software *fast* — but I think they're slow. Prove me wrong.**
>
> - random stakeholder who's ruining your day



Two answers

OKRs (Objectives & Key Results)

- Ambitious goals that describe how we want to **grow & change** as a business.
- Answers: are we making good progress on our highest priorities?
- Subject to change every quarter.
- Focus of intense work/effort.



KPIs (Key Performance Indicators)

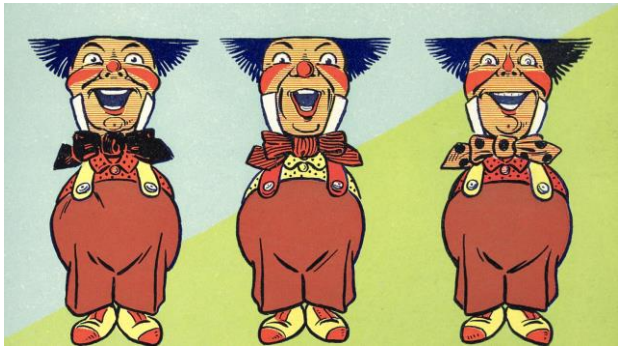
- Metrics that assess the **effectiveness & health** of ongoing business operations.
- Answers: are we working effectively?
- Longitudinal; tracked over the long haul to watch for both slow & acute changes.
- Generally bad to try to optimize…
- … but can be a basis for OKRs when something isn’t working.

Engineering KPIs: the big bikeshed

What do we want to track? Lots of ideas:

- Deployment frequency
- Incident frequency
- Incident duration
- Pull Request metrics
  - Rate of common events
  - Duration from PR open to merge
- Story points
- On-call tasks
- Employee NPS
- On-call Sadness™
- [30 more]

Lesson #1:
Standardized metrics reduce bikeshedding (thanks DORA!)



What are DORA Metrics?

Wait, you said you were Elite what?

What are the DORA Metrics?

- Developed by an independent group of DevOps pioneers (Nicole Forsgren, Gene Kim, Jez Humble)
- Based on survey data of thousands, published in the “Accelerate State of DevOps” report
- Focused on factors that “drive software delivery as well as operational & organizational performance”
- DORA is now part of Google

Source: https://www.devops-research.com/research.html

What are the DORA Metrics?

1. **Deployment Frequency**
   - How often do you deploy to users?
2. **Change Lead Time**
   - How long until a code change gets to prod?
3. **Change Failure Rate**
   - What changes break prod/require rollback?
4. **Mean Time to Recovery**
   - How long to restore service in an incident?
5. **Reliability (new)**
   - Do you meet your reliability targets?

Source: https://cloud.google.com/blog/products/devops-sre/using-the-four-keys-to-measure-your-devops-performance

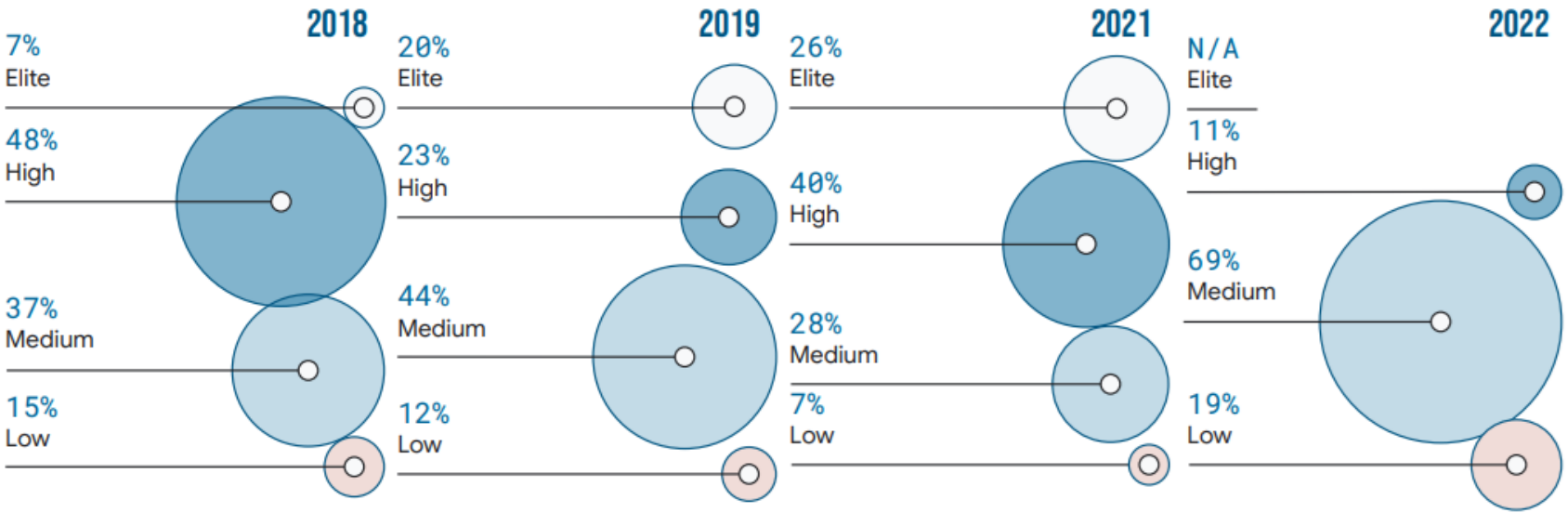
DORA tiers

1. Elite
2. High
3. Medium
4. Low

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%

Source: https://cloud.google.com/blog/products/devops-sre/announcing-dora-2021-accelerate-state-of-devops-report

DORA tiers



Source: https://cloud.google.com/blog/products/devops-sre/announcing-dora-2021-accelerate-state-of-devops-report

2022 DORA tiers

1. Starting
2. Flowing
3. Slowing
4. Retiring

Cluster	Stability		Operational Performance	Throughput		% respondents
	Time to restore service	Change failure rate	Reliability	Lead time	Deployment frequency	
Starting	Between one day and one week	31%-45%	Sometimes meet expectations	Between one week and one month	Between once per week and once per month	28%
Flowing	Less than one hour	0%-15%	Usually meet expectations	Less than one day	On demand (multiple deploys per day)	17%
Slowing	Less than one day	0%-15%	Usually meet expectations	Between one week and one month	Between once per week and once per month	34%
Retiring	Between one month and six months	46%-60%	Usually meet expectations	Between one month and six months	Between once per month and once every 6 months	21%

Source: https://services.google.com/fh/files/misc/2022_state_of_devops_report.pdf

Back to the problem...

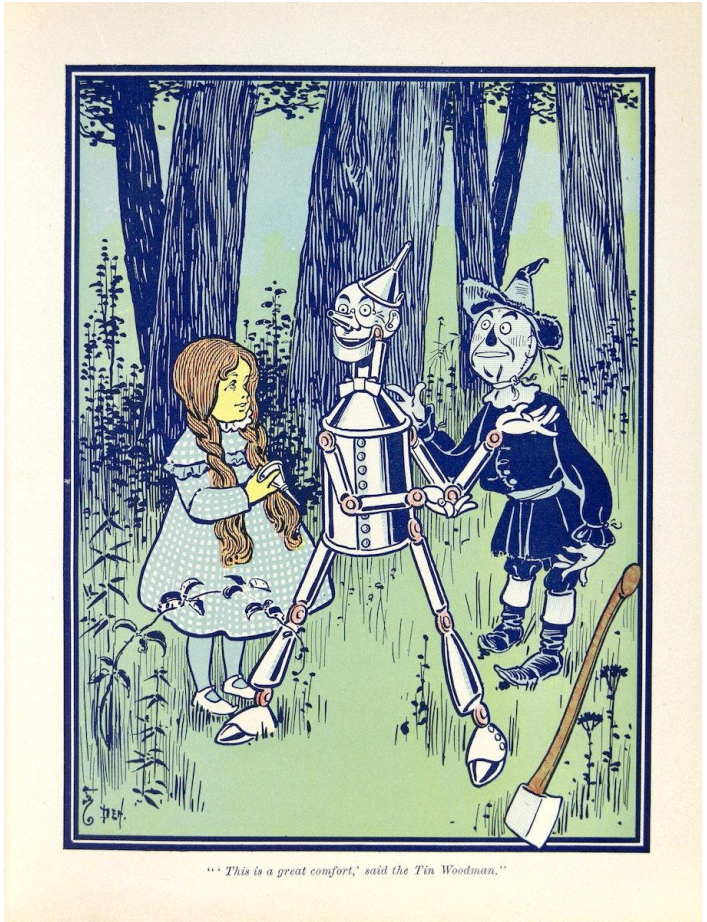
Why DORA Metrics?

They are correlated with the experience of the *two* groups I care most about:

- Customers
- Employees

Caution: measuring work evokes *feelings*

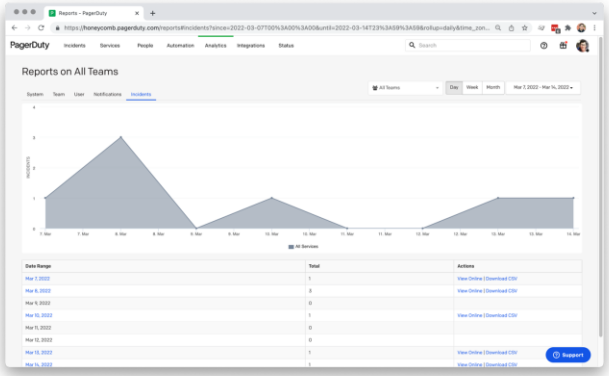
Lesson #2:
Lots of people are uncomfortable with being measured. Focus on the “whys” behind the metrics — and how they’ll be used.

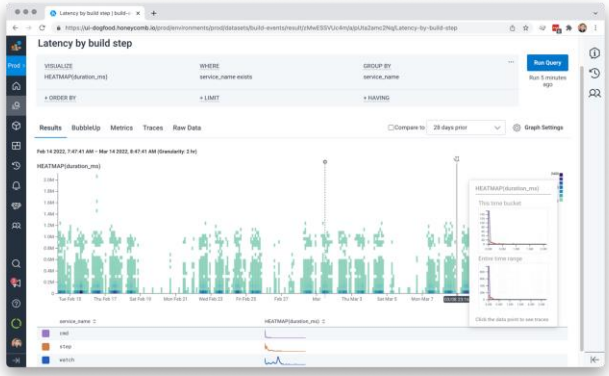


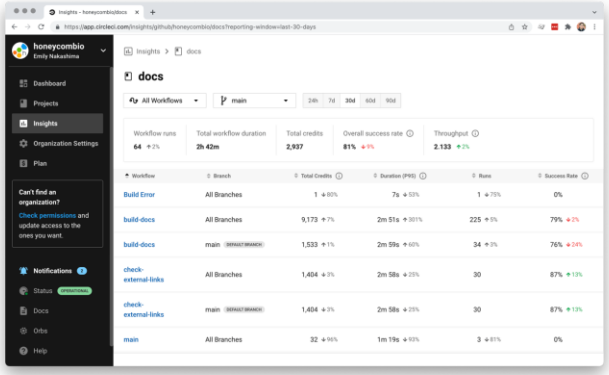
Getting started

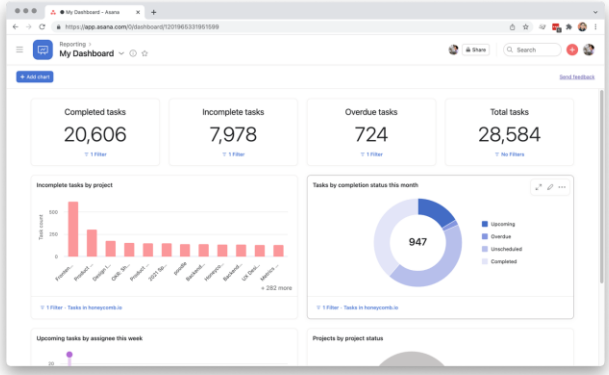
How we started tracking these metrics

Step 1: validate metrics w/ least possible effort









Relevant graphs in our existing tools, clockwise from top left:

1. Number of incidents (part of metric #4)
2. Build time from pipeline (part of metric #2)
3. Build time from CI provider (less useful)
4. Ticket stats (part of metric #2 but useless)

Step 1: validate metrics w/ least possible effort

How to measure each metric in a way that balances ease against usefulness?

1. **Deployment Frequency**
   - Instrument deploy events and send to observability stack
2. **Change Lead Time**
   - Start as multiple measures:
     - Ticket cycle time
     - Build time
     - PR cycle time
3. **Change Failure Rate**
   - Instrument rollback events and send to observability stack
4. **Mean Time to Recovery**
   - Use PagerDuty webhooks to send incident start & stop to observability stack
5. **Reliability**
   - Use existing SLOs in Honeycomb

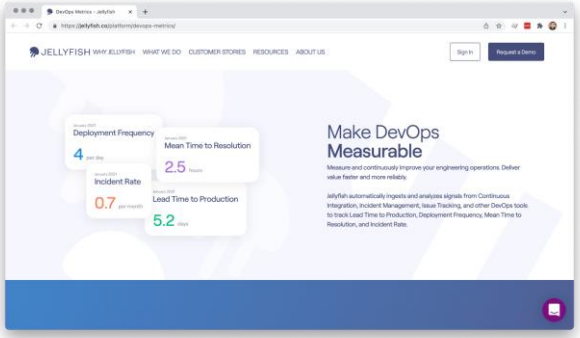
Step 2: make the big project plan

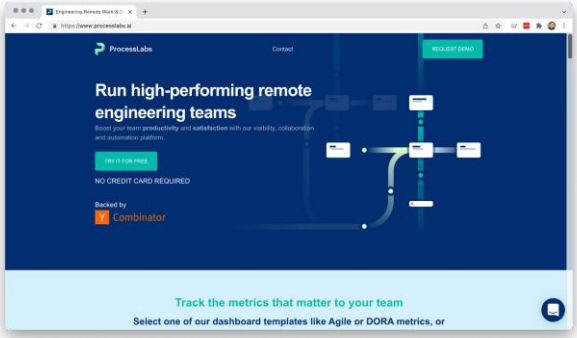
"I'll just write a little software to capture these metrics myself…"

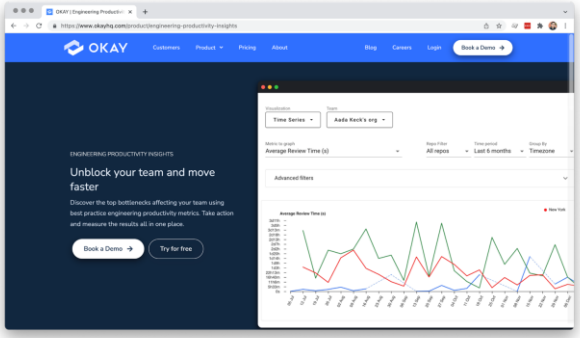
Step 3: scrap the plan for roadmap work

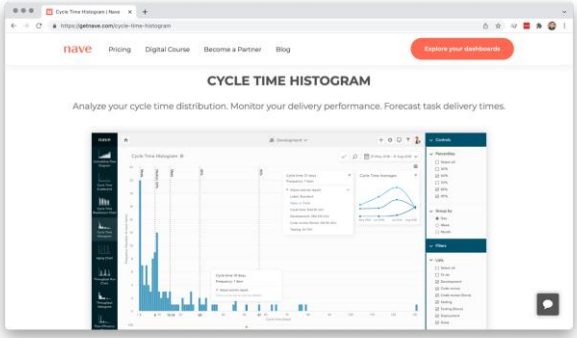
“Well, that was a mistake”

Step 4: evaluate vendor products







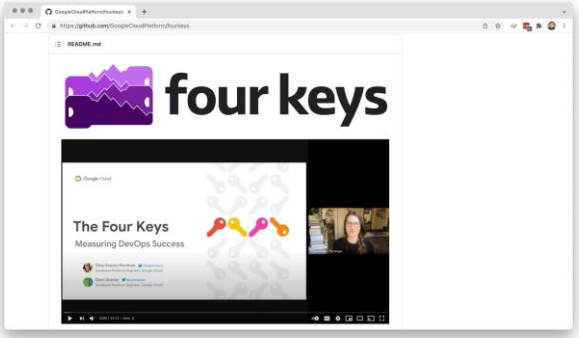


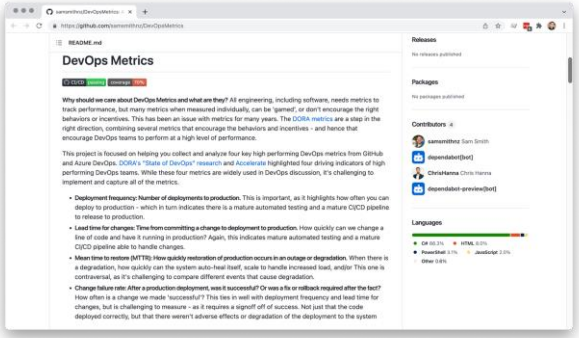
Vendor products I evaluated in this space, clockwise from top left:

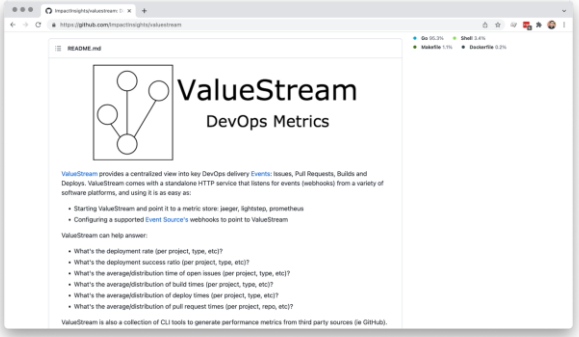
1. Jellyfish
2. ProcessLabs
3. Okay
4. Nave (for Asana cycle time; not DORA-specific)

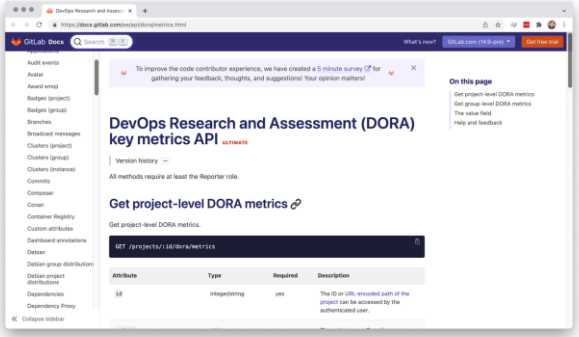
Plus many more: Code Climate, LinearB, etc.

Step 4: evaluate OSS





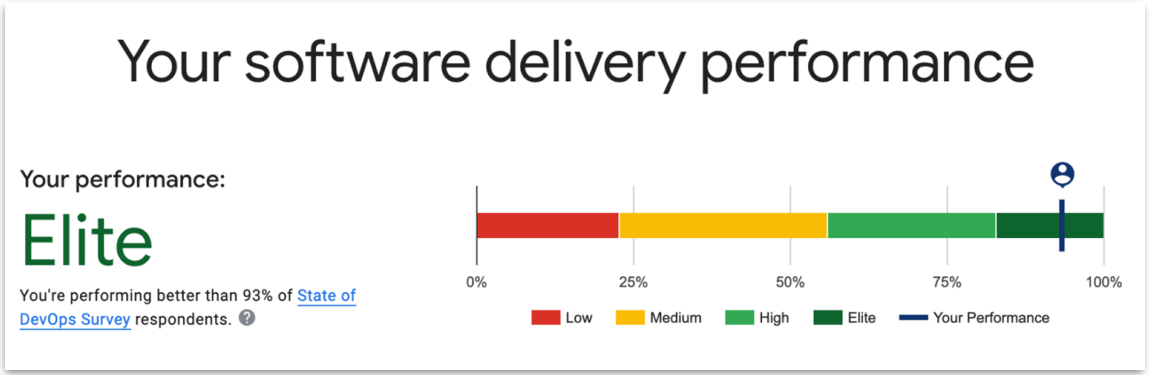




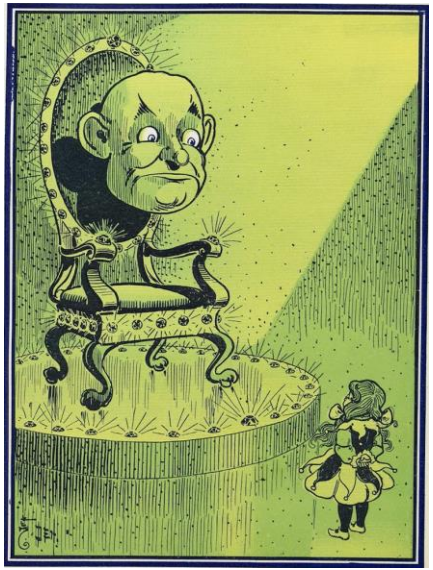
Relevant OSS tools, clockwise from top left:

1. Four Keys project from Google/DORA
2. DevOps Metrics project on GitHub
3. ValueStream project on GitHub
4. DORA metrics support in GitLab

Success! We're a 2021 "Elite" team



> **We’re an elite team — and here’s the data to prove it.**
>
> - me to our stakeholder friend



How’d it go? What’s next?

Lesson #3:
Department-wide is good for KPIs, but to actually diagnose, you need to break down by team, service, and individual.

Lesson #4:
Eliminating recency bias is valuable.
Software delivery can be cyclical.

Lesson 5:
Amazing software delivery and operations != value to customers.

Sometimes perception problems are communication problems.

Summary: where did we end up?

The good, the bad, the ugly

- **Good: actionable data**
  - Successfully improved stakeholder management.
  - Builds are a frog boil: 7 minutes average -> 18 minutes average over 3 years.
  - Spotted some “hidden work:” required team coordination with no channel.
- **Bad: imperfect visibility** into some metrics due to tool constraints.
  - Cycle time is hard to track well. Either do lots of work or accept imperfect numbers.
- **Ugly: data is not centralized**
  - I have to report out on it manually, and it’s push data, not pull data.
  - Not as easy for engineers to access the data as I would like.

What's next?

- Have engineers pick additional team KPIs
  - Show team pain in addition to system health
- Make data more available to every engineer
- Combine with product KPIs to capture value delivery
- Refine team ritual for working with all these KPIs

… and keep iterating, forever!

Thank you!