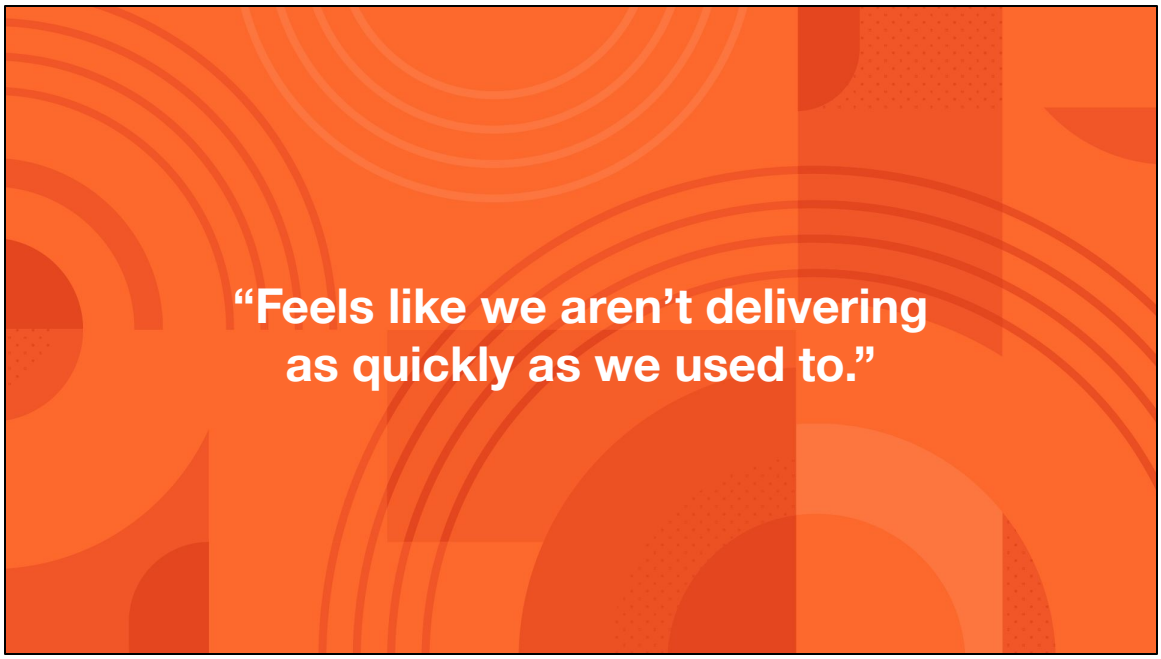
Accelerating Delivery with Metrics-Driven Insights

Virtual CTO Summit
June 2, 2020

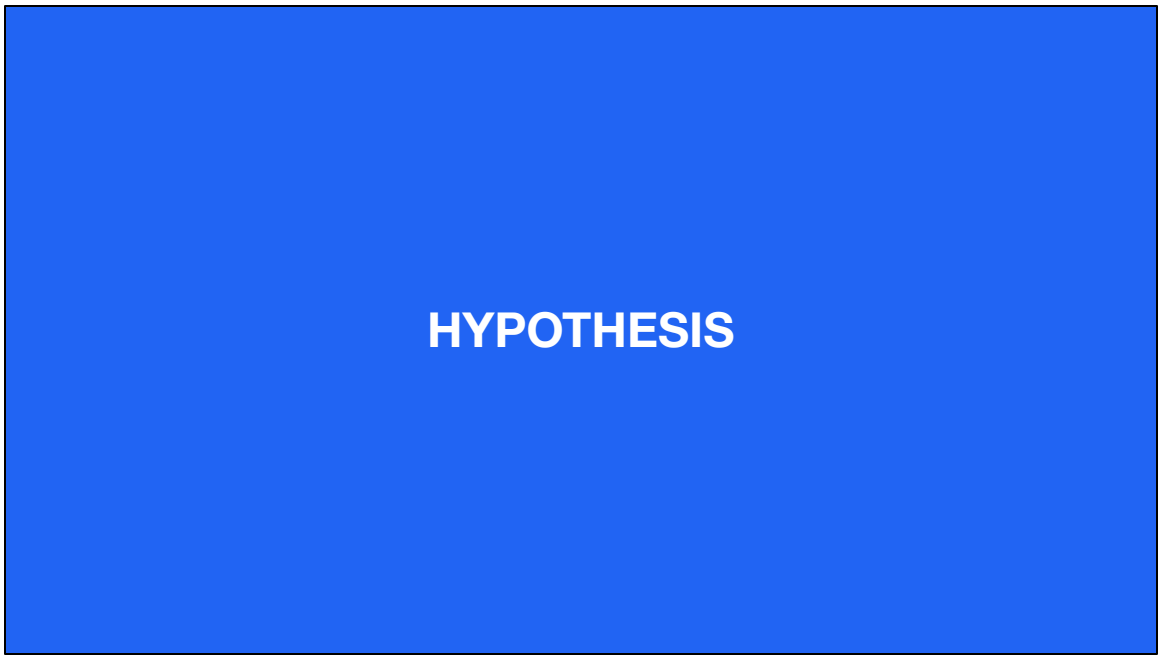
Jack Humphrey, VP Engineering, Indeed

jackhumphrey.me
linkedin.com/in/leejack
twitter.com/**@youknowjack**

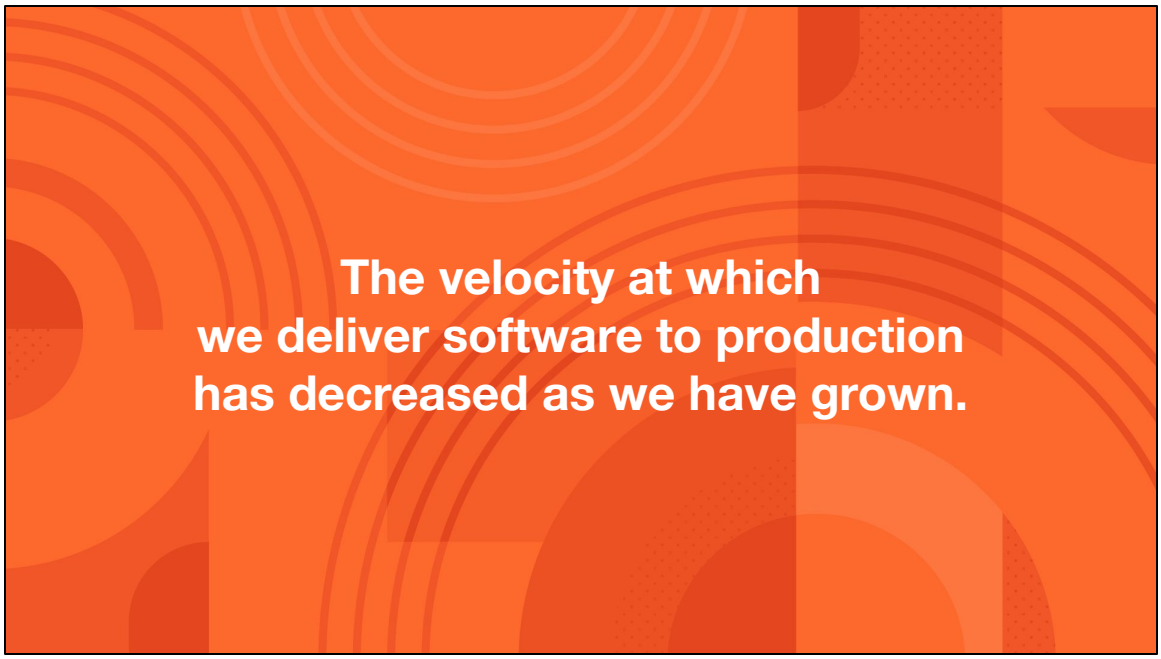
Intro



This story starts with a sentiment we were feeling in late 2018.



Really, that sentiment is a form of hypothesis.


The velocity at which
we deliver software to production
has decreased as we have grown.

Restated.

How can we increase velocity and get more done each week?

So from there, we could have asked this question.

~~How can we increase velocity and get more done each week?~~

But that’s not the right next question.

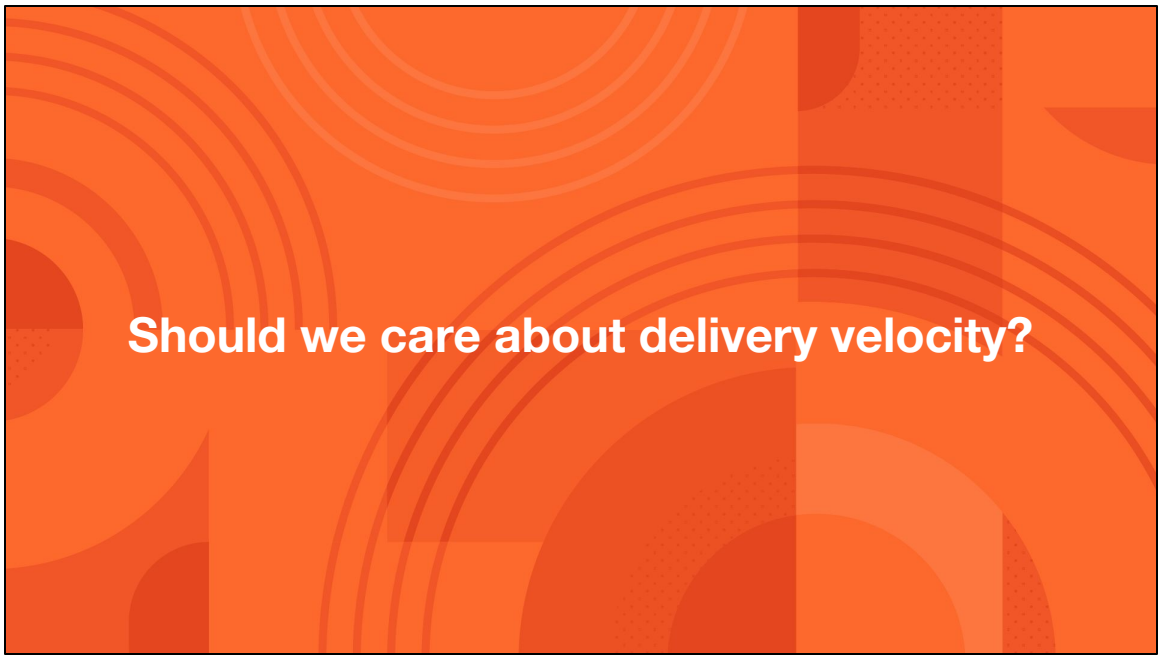
1. Should we care about delivery velocity?

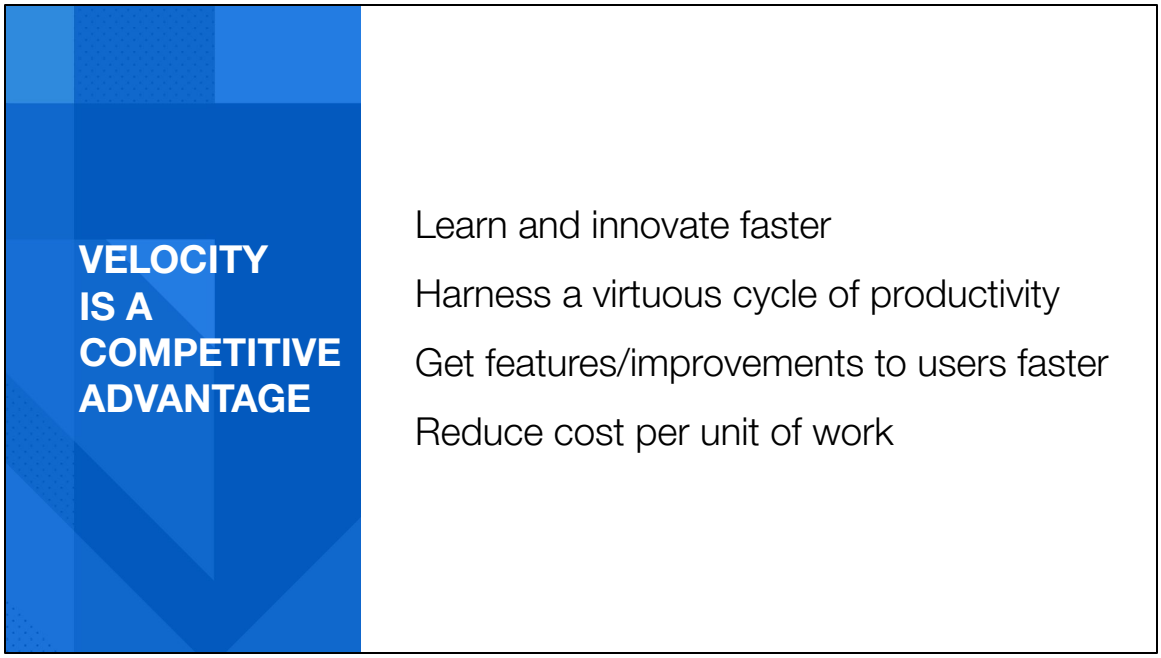
2. How do we define velocity?

3. How do we measure velocity?

4. How do we increase velocity?

Instead, I think these are the right kinds of questions to ask.


Should we care about delivery velocity?



- Delivering faster means learning and innovating faster. And this really matters to us at Indeed! We want to explore more ideas and deliver more value, as fast as we can.
- Developers on your team will feel more excited to get things done, and that will virtuously lead to higher energy and productivity.
- You'll have a competitive advantage by getting new features and improvements to your users faster.
- You may actually reduce your cost per unit of work, if that matters a lot to your bottom line.

Whether or not you want to do **more with less** or, like us, do **even more with more**, we advocate adopting the Kaizen philosophy of continuous, incremental improvement.

I encourage you to read more about Kaizen (if you haven’t already), but the emphasis is on “good change” (the literal translation), like streamlining process, or eliminating waste, rework, and excessive communication.

In order to implement a Kaizen approach, you have to develop measurements, and you have to be open to learning and continuously evolving over time.


How do we define velocity?

...and can we measure it?

There is no single best metric for development velocity.

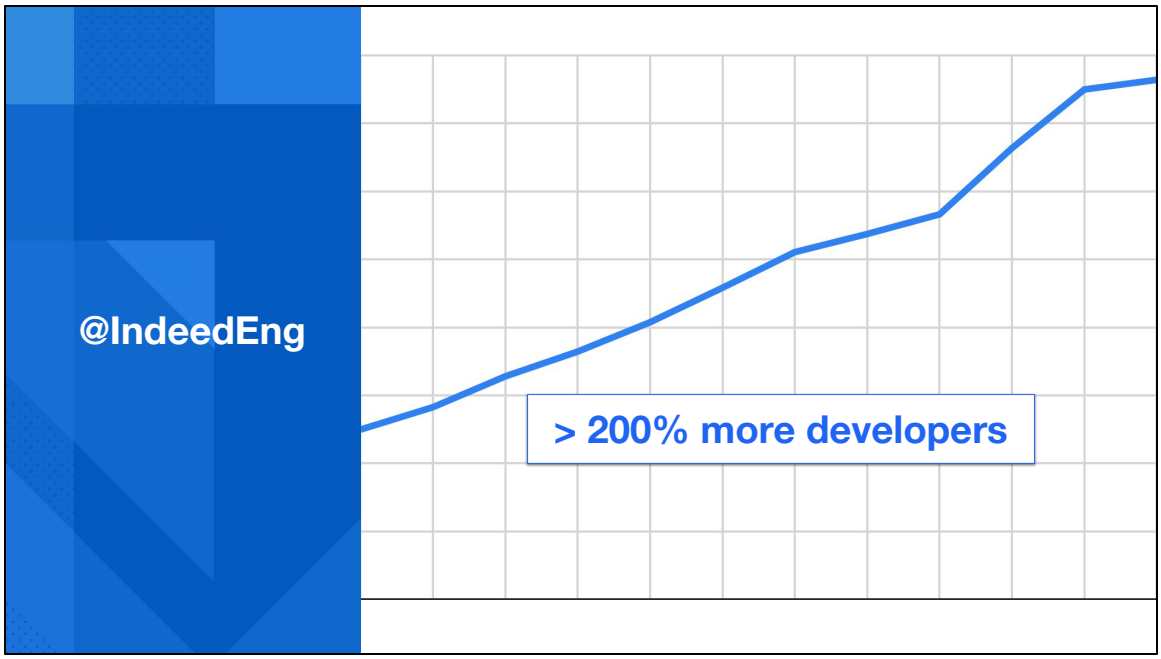
- There are different ways to measure software delivery and velocity.
- There is no silver bullet for measuring velocity.
- You should start with what matters for your organization and work backwards on reliably measuring that metric for continuous improvement.

THROUGHPUT

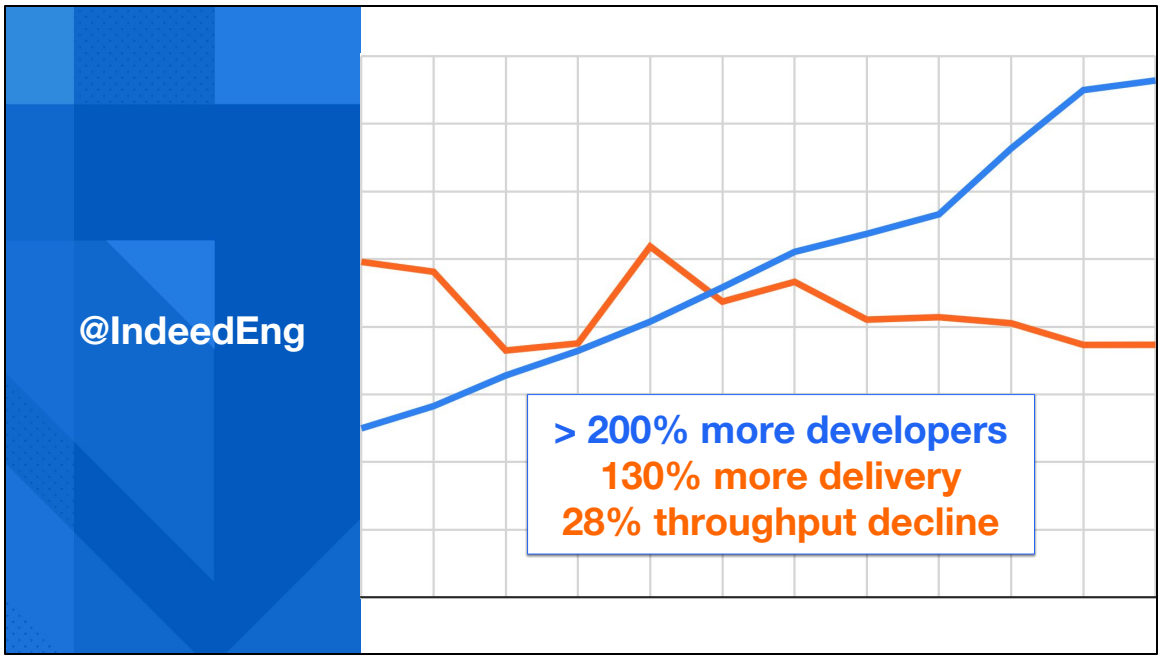
Total number of changes are deployed to production over a given period of time

Total number of experiments completed over a given period of time

- Throughput is a great way to measure how many features are complete and can be delivered to customers in a given time interval
- Higher throughput would mean higher velocity
- Throughput will start to decrease if there are bottlenecks or increased unnecessary communications in the team



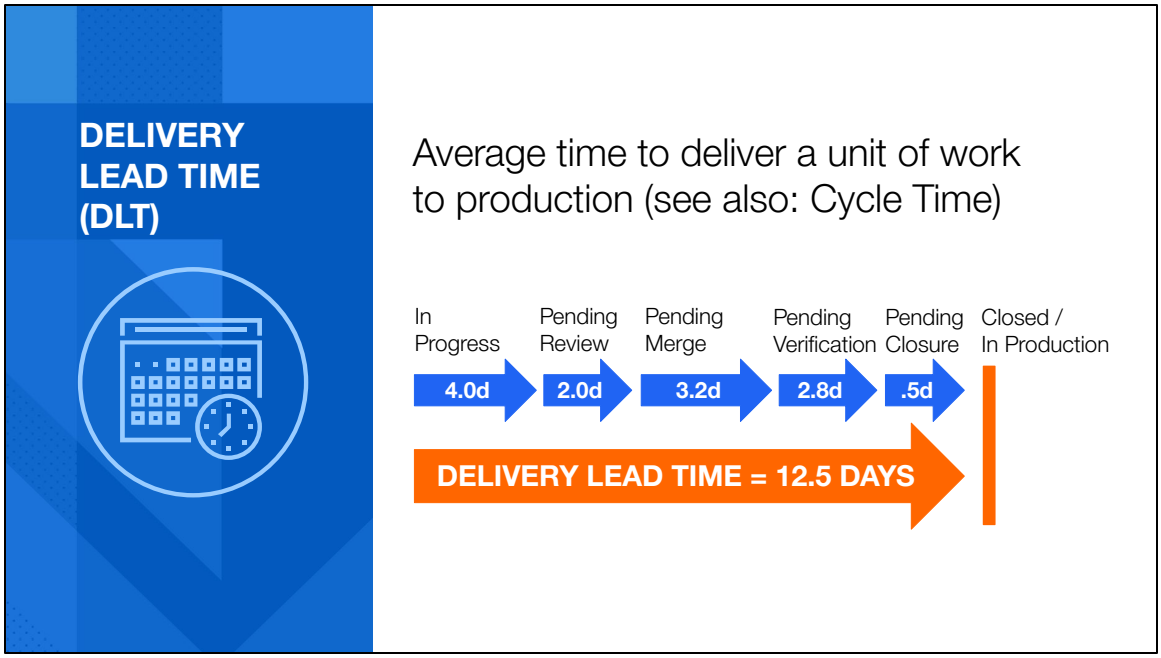
Indeed Engineering grew a lot in the years leading up to 2019, adding over 200% more developers who could work on new features and improvements to our products.



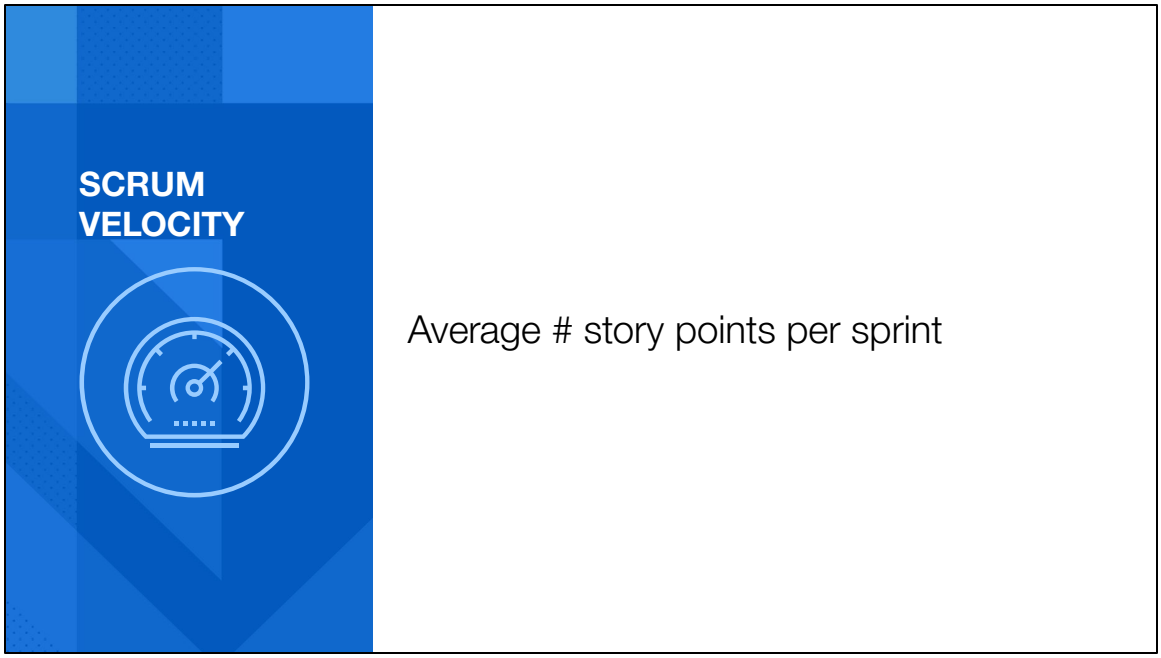
However, over that time, throughput per developer (the ORANGE line) declined as we grew. This is a well-understood phenomenon, when communication and coordination overhead increase, and ad-hoc decision-making starts to slow delivery.

Over those few years, we added >200% more developers, but we only saw a 130% increase in the number of new features and improvements implemented. That's a 28% decline in throughput per developer.

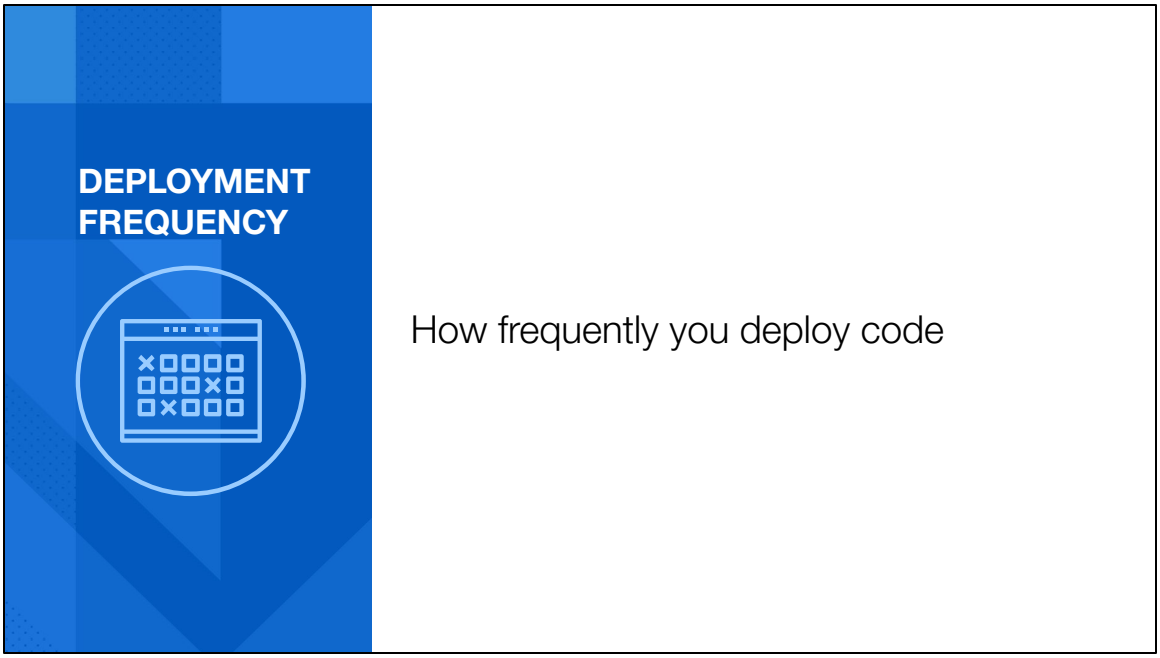
You might say this is not so bad, but it's not the kind of return we'd like to get on our investment in growing our organization. So we turned our attention over the last year to driving greater velocity.



- Delivery Lead Time aka DLT is a way to measure the time it takes to deliver a unit of work to customer
- Software Development and Delivery can be comprised of different states like, In Development, In Code Review, In Merge, In Verification etc for different teams
- DLT is sum total of all the time it takes to spend in a particular state and the wait time between those states
- In this scenario, total DLT is 12.5 days
- Teams that are driven to provide faster value to customers can focus on reducing the average DLT for a unit of work
- DLT can be reduced by eliminating or reducing the wait times between the states or by simplifying the process through automation and CI/CD best practices



- Some teams use story points to measure velocity. This is also called as scrum velocity.
- Typically, this is measured after the sprint is complete
- Efficient teams will have a predictable scrum velocity based on the capacity of team



- Deployment Frequency is way to measure how frequently the team deploys to production
- **This is not the same as throughput, which is about how many changes you deliver in a period of time. This is just the cadence at which that delivery is happening.**
- Having a high deployment frequency means that we are d**elivering smaller sets of changes in each deployment**. This will give more opportunities for learning and faster feedback loop.
- This will also reduce the risk as we are delivering more frequently with smaller sizes.

WORK IN PROGRESS (WIP)

Amount of work partially finished and awaiting completion

WORK IN PROGRESS (WIP)

Limiting WIP boosts collaboration and increases throughput

Capacity: 95%
Avg Speed: 19 mph

Throughput:
70 cars/segment/hour

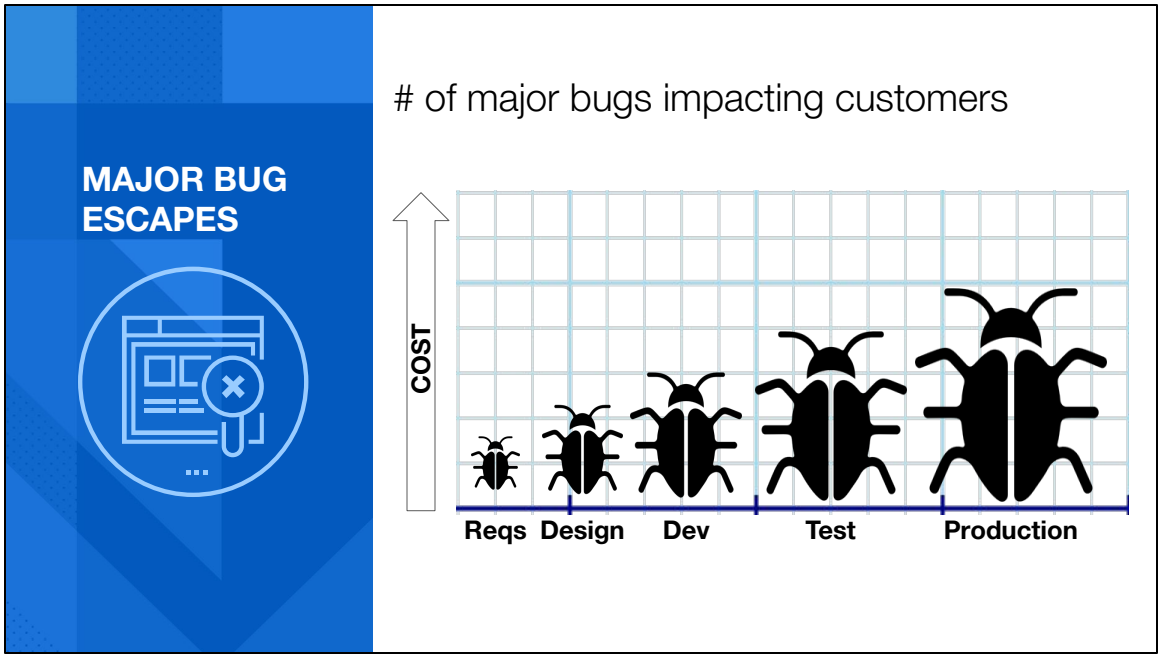
Capacity: 60%
Avg Speed: 60 mph

Throughput:
80 cars/segment/hour

Inspiration: https://go.indeed.com/NWZNA8

- For higher velocity/throughput, WIP should be lower,
- More WIP will result in greater overhead on context switching, higher DLT and lesser throughput
- WIP Limits encourage you not to maximize utilization of the individuals in your team, just like it's not very effective to maximize utilization of a highway

BTW
Shouldn’t we also worry about quality?



- The later in the delivery process you find bugs that affect your customers, the costlier they are for you to resolve.
- And that cost impacts the velocity of your team.
- So one metric you can look at is what we sometimes call “Major Bug Escapes” -- the number of major bugs that get out into production.

CHANGE FAIL PERCENTAGE

What % of deploys result in failure?

Another interesting measurement of quality is your change fail percentage.
This is the percentage of your deploys that fail. In the great book Accelerate (by Forsgren, Humble, and Kim), they define failed deploys as deploys that **“result in degraded service or subsequently require remediation"**

It’s great if you can deploy smaller releases more frequently, but not if too many of those deploys require remediation.

MEAN TIME TO RESTORE (MTTR)

How long does it takes to restore service after failure?

Another quality delivery metric they discuss in Accelerate is Mean Time to Restore, or MTTR.

When you have a failure, how long does it take you to restore service? Even if you have a low Change Fail Percentage, if it takes you days to restore service, the impact is still huge for your users. By the same token, if you have a higher Change Fail Percentage, but a really low MTTR, your users might never even notice those failures!

How did we choose and develop our metrics?

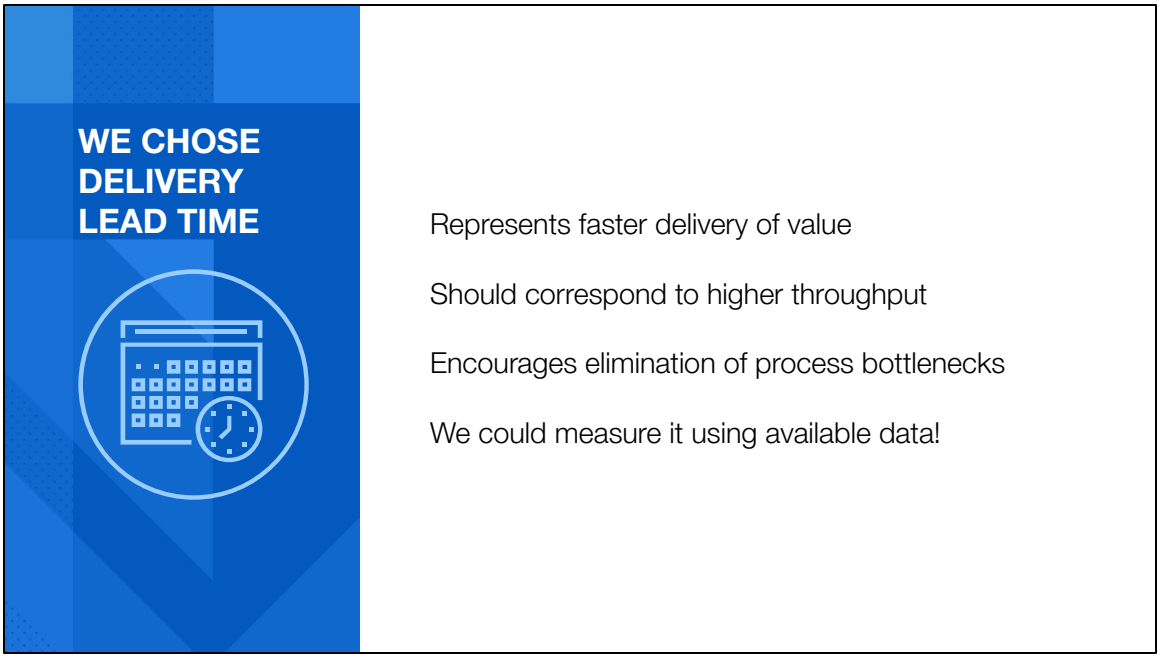
We explored our Jira data.

To hone in on measuring Delivery Lead Time, we explored our Jira data. Jira is where we track work, so we knew the data is in there, we just had to figure out how to get it out.

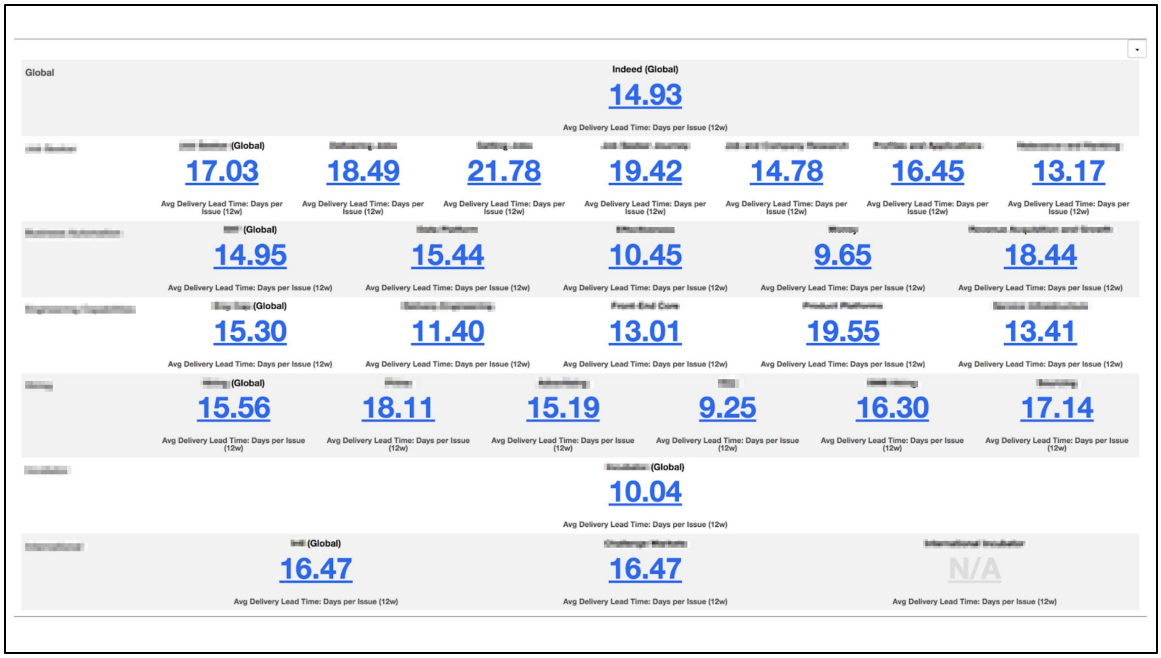
Imhotep analysis of Jira issue state transitions

github.com/indeedeng/imhotep

Luckily, we were able to do this using Imhotep, our open source data analytics platform. We had built a dataset of all actions (including state transitions) on Jira issues over time. Using this, we could look at time spent in different stages of delivery.



- We picked DLT as our core metric to measure and improve
- Improving DLT can result in improving other types of velocity metrics
- Few reasons behind why we have picked DLT amongst others
    - DLT will enable our teams to deliver faster value to our customers.
    - If DLT is reduced, developers will have more time to work on other features or improvements and will provide them with opportunities to increase their throughput and productivity
    - To reduce DLT, teams should look at bottlenecks in the delivery process and will have to either eliminate or reduce the time spent on these bottlenecks
    - Last but not least, using our Imhotep data set we could measure the DLT for all the units of work performed by our developers



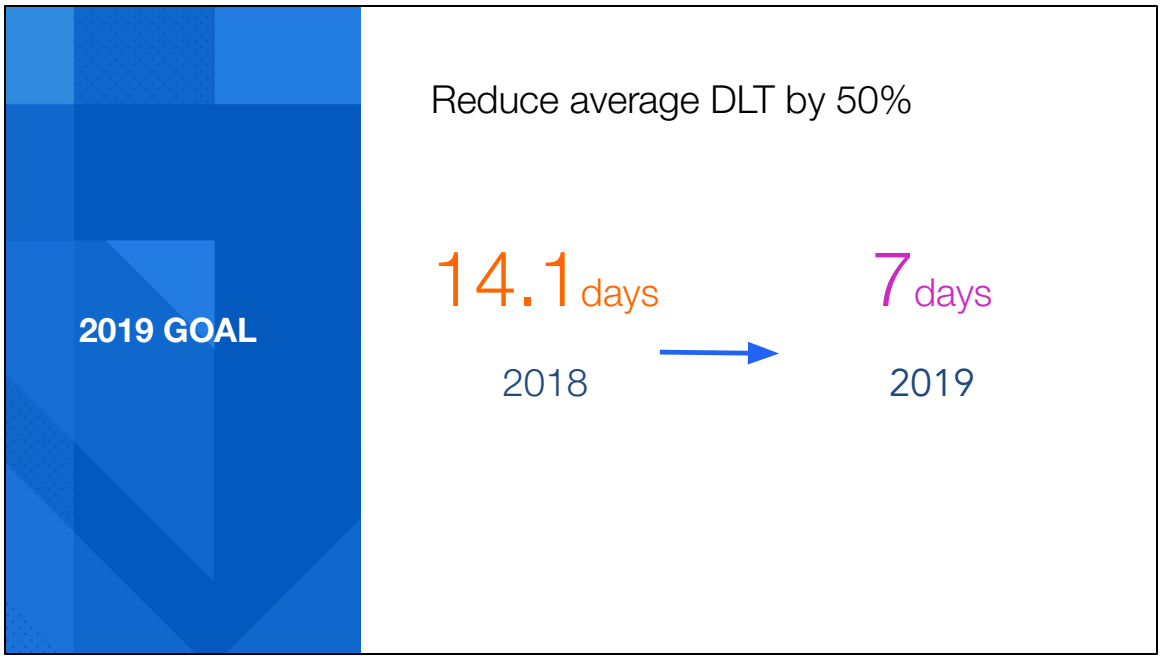
This is a dashboard that we developed, and this screenshot is from early 2019. The number at the top is our average (mean) DLT across Indeed. Almost 15 days.

We were able to further break it down by organizations within Indeed (the rows) and groups of teams within those orgs (across). As you can see, there was a lot of variation by team. Some were at nearly 22 days on average, and others as low as 9 or 10 days. This was a critical observation, and we looked closely at teams at all points on the spectrum to see what they were doing and how well this measurement worked for them.

INDEED DLT IN EARLY 2019

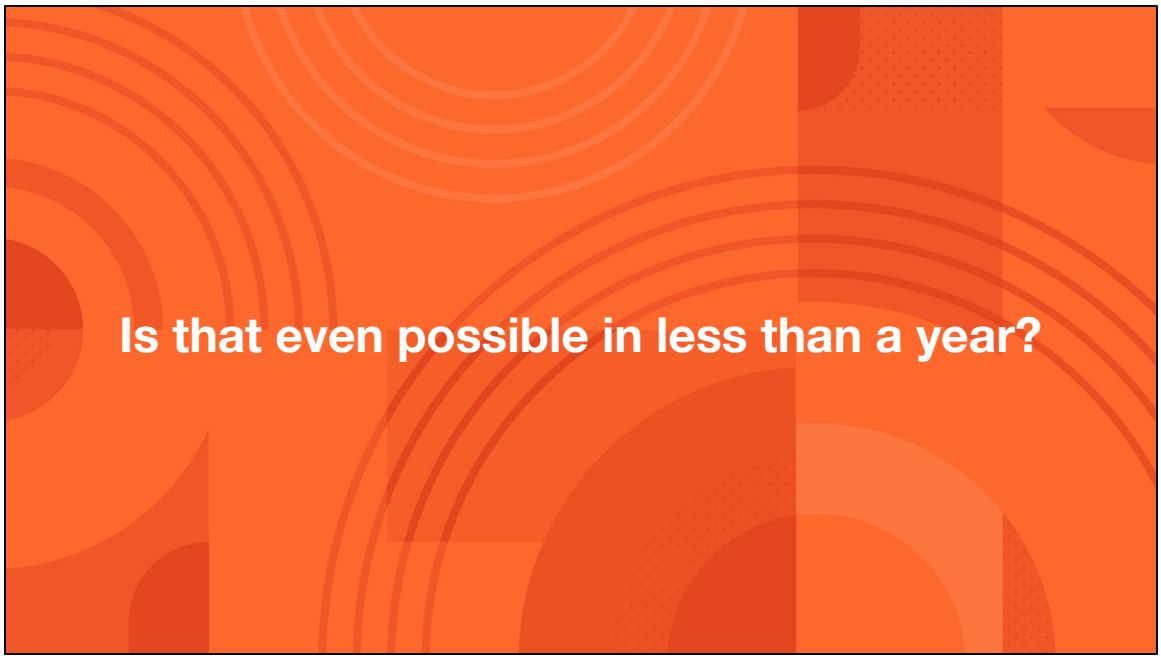
Average	14.9 days
50th Percentile	8.2 days
65th Percentile	14 days
75th Percentile	21 days

So again, were started the year around with a mean DLT of 15 days. But notice that the median, or 50th percentile, was 8.2 days, and 25% of all Jira issues took more than 21 days to resolve. We had a lot of variation in delivery lead time.

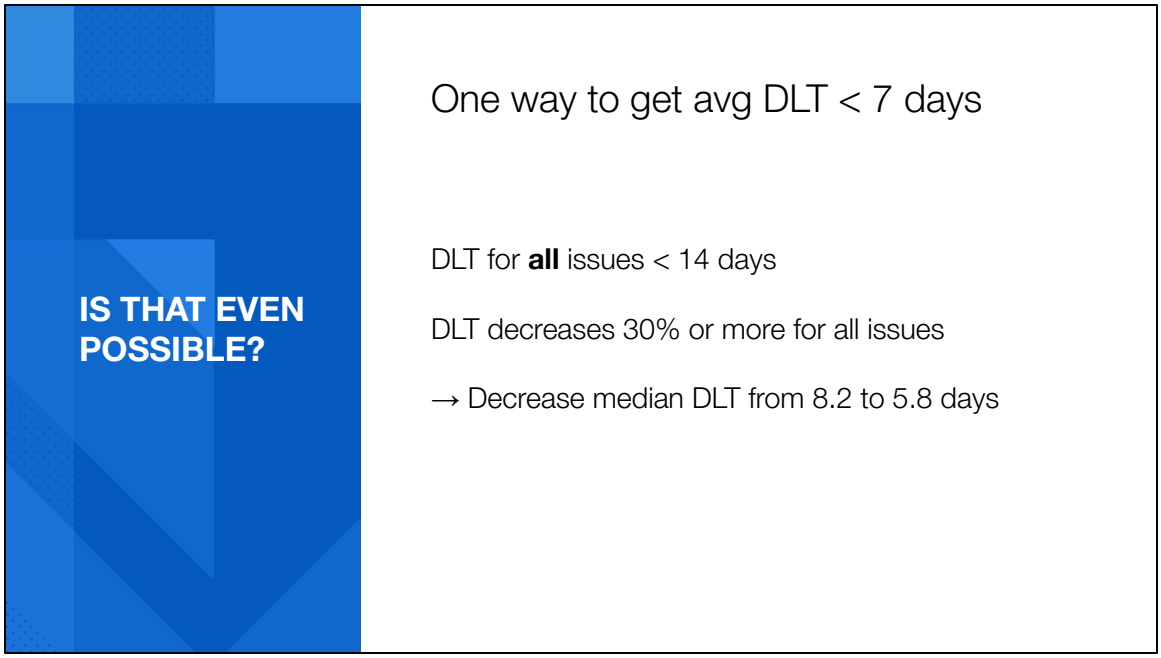


Before we even got into 2019 and got that readout, we had a preliminary measure of 14 days in December 2018, and we decided to embark on a program through 2019 to reverse the slow-down in delivery.

We set a goal of 7 days on average by the end of the year. This appealed to us intuitively, because we felt that most units of work should be deliverable in about a calendar week.

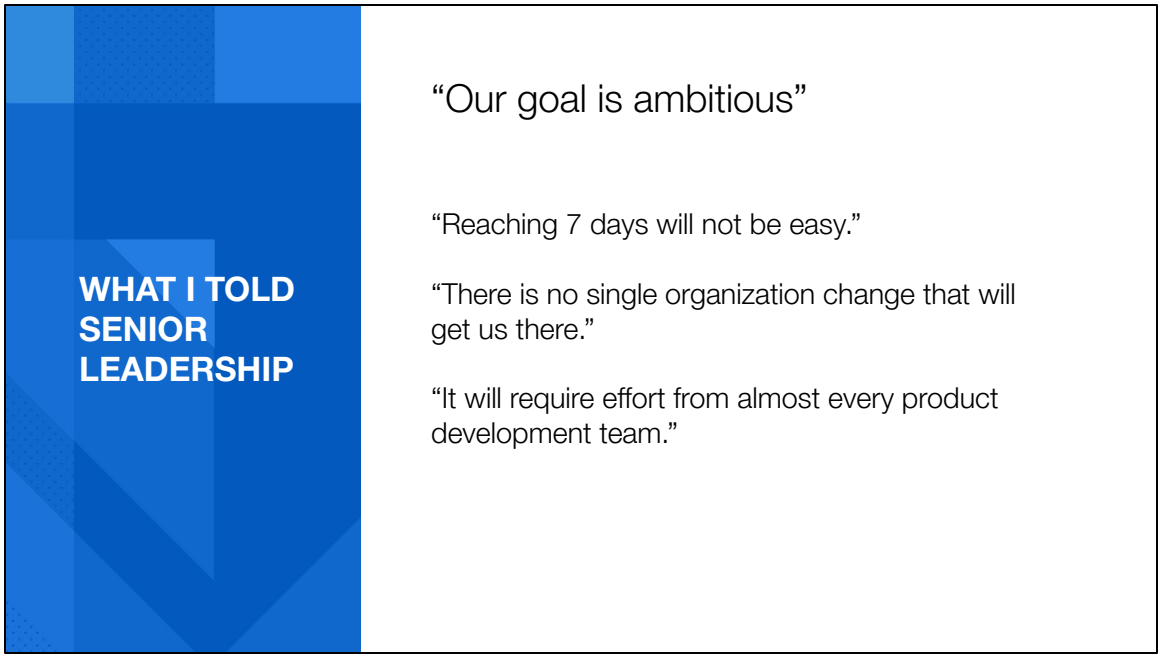


But it was important to me to convince myself it was an achievable goal.



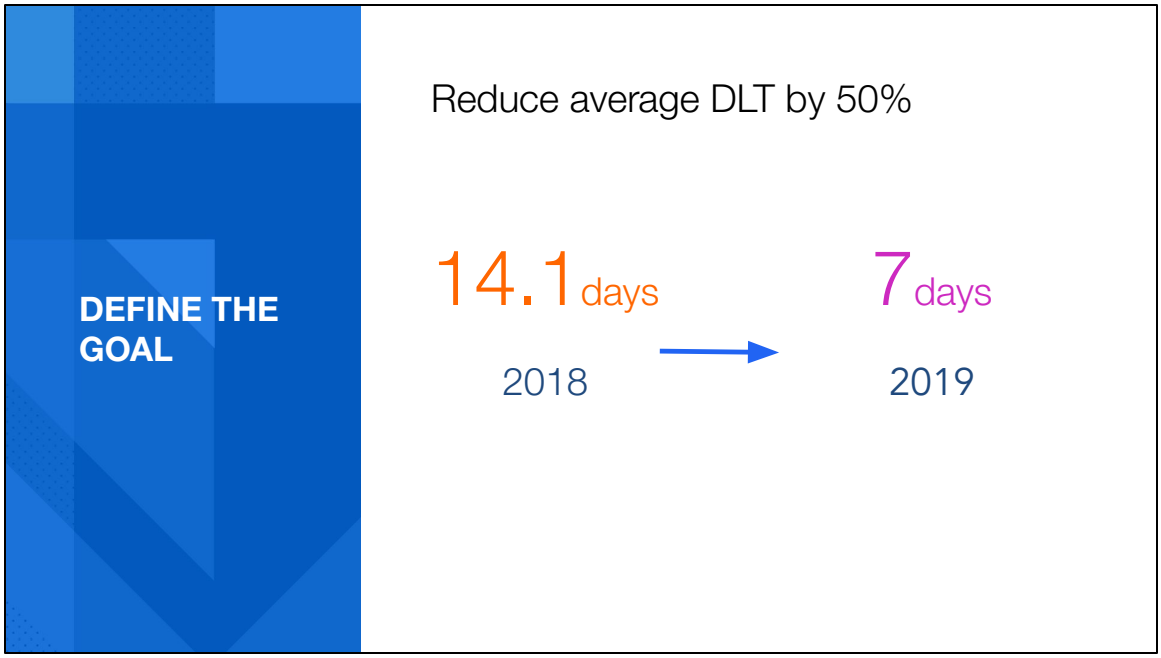
I looked at scenarios based on 2018 data, and I saw that one way to hit our goal for getting the average to 7 days would be to eradicate the really long DLTs, and get all issues' DLTs under 14 days. One top of that, we would need to see DLT decreases of at least 30% across the board.

If we managed all that, our median would drop from 8.2 days to 5.8 days.

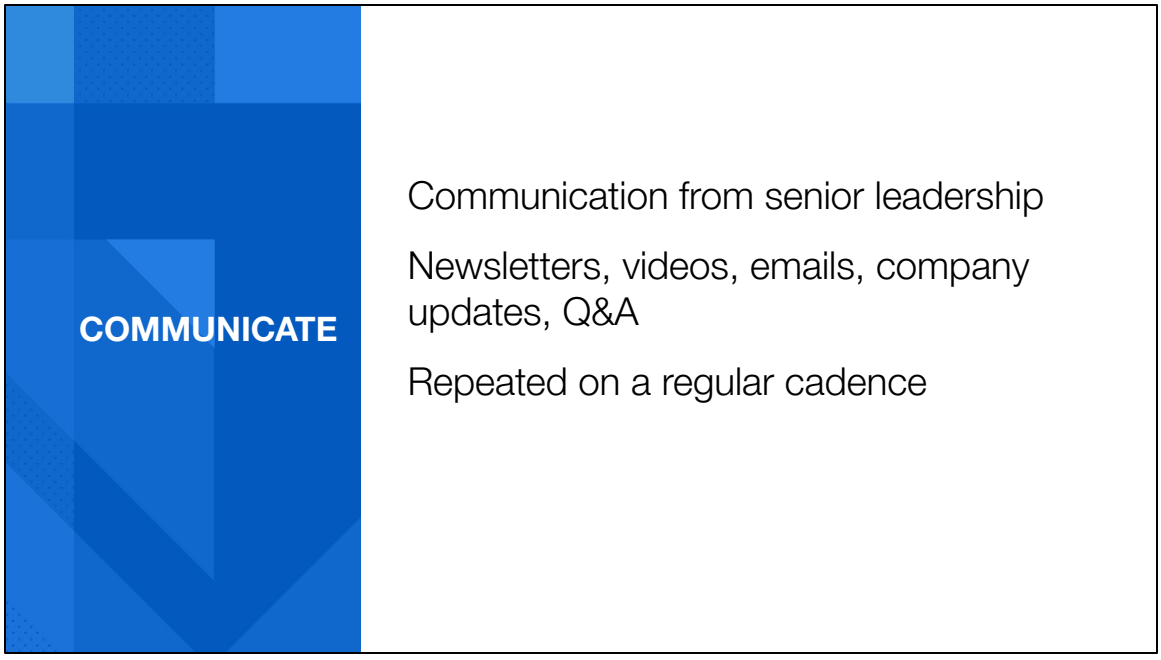


But I made these points very clear to senior leadership.

What did we actually do?

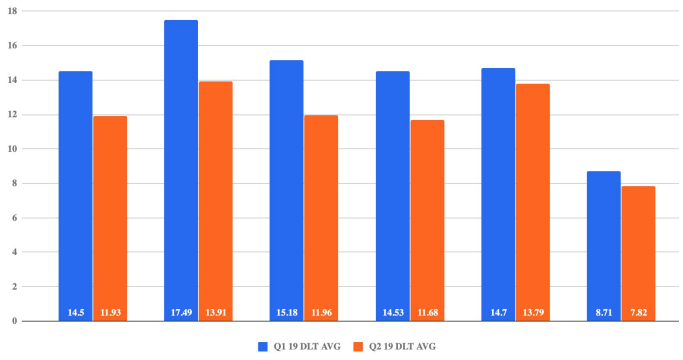


As I noted, we set a goal.


COMMUNICATE
Communication from senior leadership
Newsletters, videos, emails, company updates, Q&A
Repeated on a regular cadence

ALIGN TEAM GOALS

Every group/team set a DLT goal.



Delivery Lead Time by Group, Q1 Vs Q2

This was crucial, every group set a DLT improvement goal. And we tracked and reported to leadership the progress. Here we see that in Q2, every group was making progress to reduce their DLT.

DELIVER INSIGHTS

We built tools to explore DLT.

We built tools to help teams better understand their Delivery Lead Time. This is a screenshot from Peregrine, an interactive dashboard we built for this purpose. In this view, a team can examine how all the different stages of delivery contribute to DLT over time. This allowed teams to focus in on where the biggest opportunities for improvement were.

PROVIDE TRAINING AND SUPPORT

We offered velocity workshops.

Senior eng leaders from around Indeed led workshops to share techniques for improving velocity. This helped people take the insights from Peregrine and attempt different process changes to address them.

ALIGN INDIVIDUAL INCENTIVES

We encouraged leaders to value individual contributions to velocity improvement.

We modified performance/career rubrics to clarify this value.

MEASURE AND REPORT ON PROGRESS

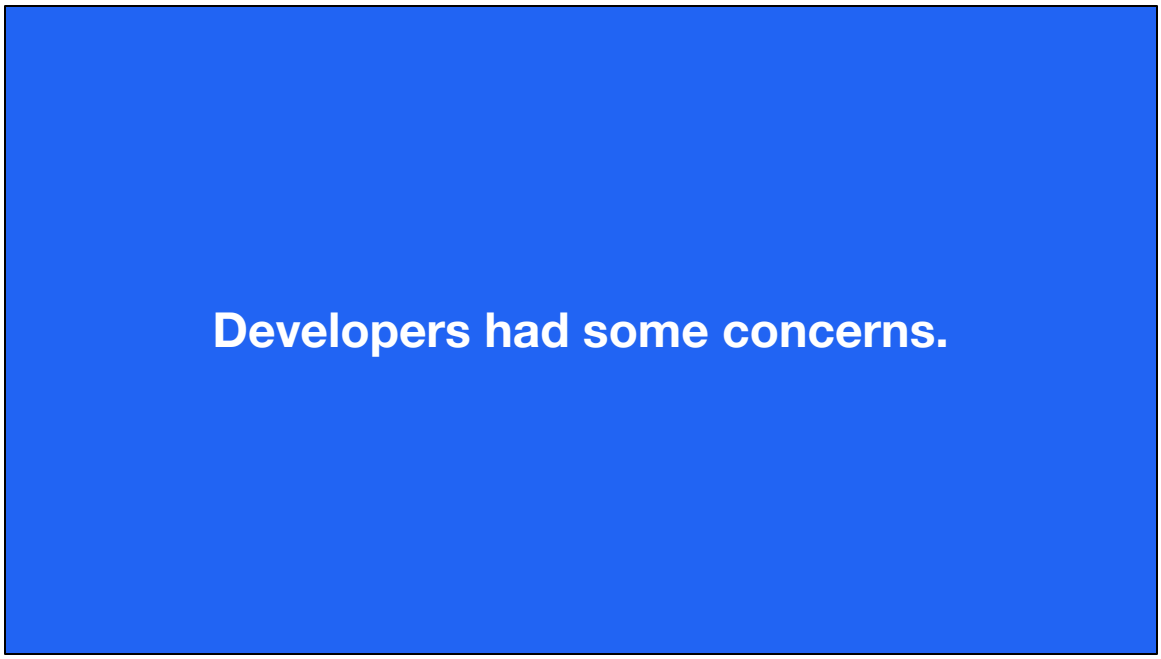
Dashboards

Slack notifications

Content widgets

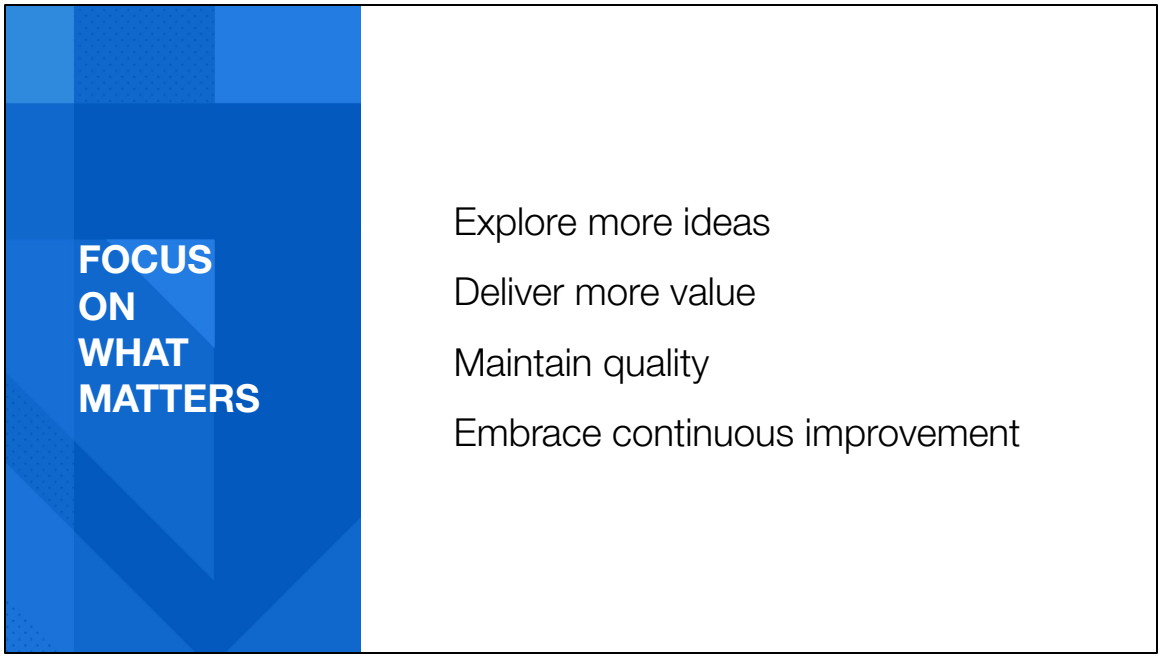
Leadership reviews

Company updates



Along the way, I heard concerns from developers. Here are three.

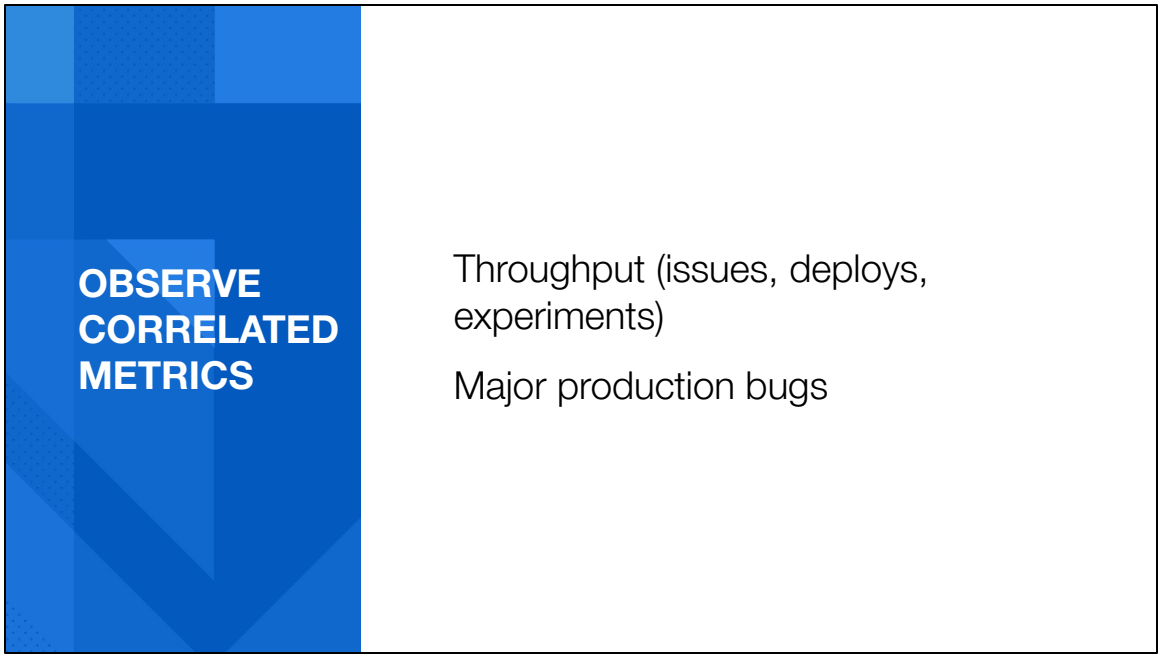
People have heard me talk about Goodhart’s Law, which says that when a measurement becomes a target, it ceases to be a good measurement. This is something you have to actively work to counteract when you have a goal based on your metric. And we did that be constantly reminding people that hitting the number is not what matters.



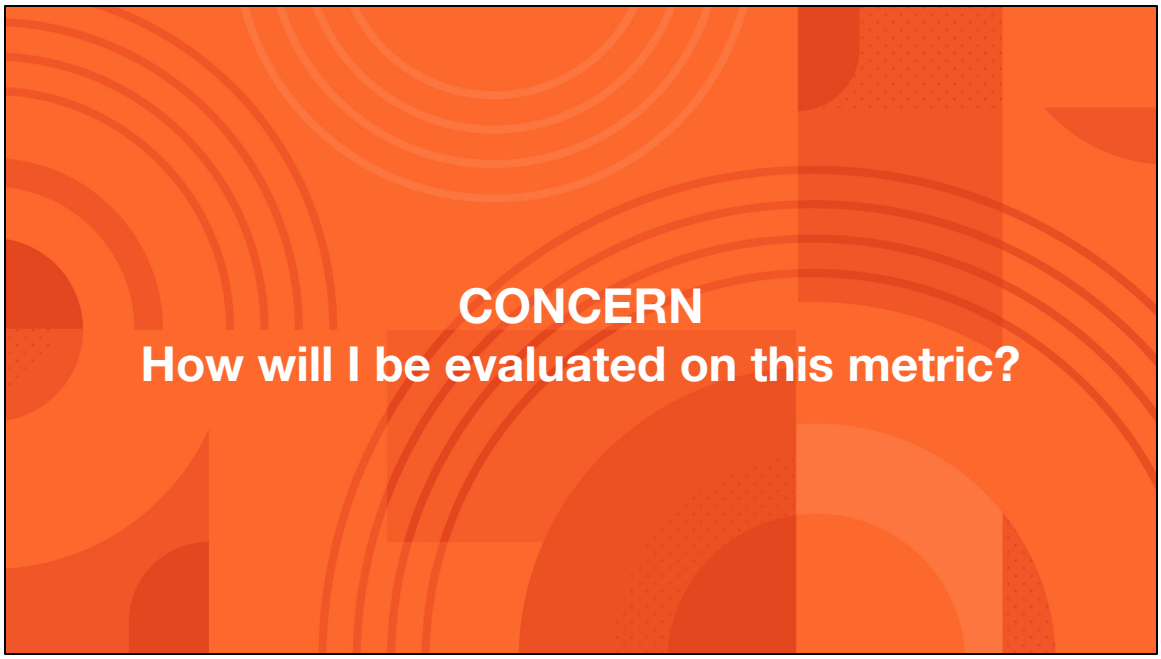
We encouraged your teams to focus on what matters.

What's important is not hitting the number, but getting the benefit:
Exploring more ideas, delivering more value, and maintaining quality while we do it.

A continuously, iteratively improving, in ways that are meaningful to the people doing the work.



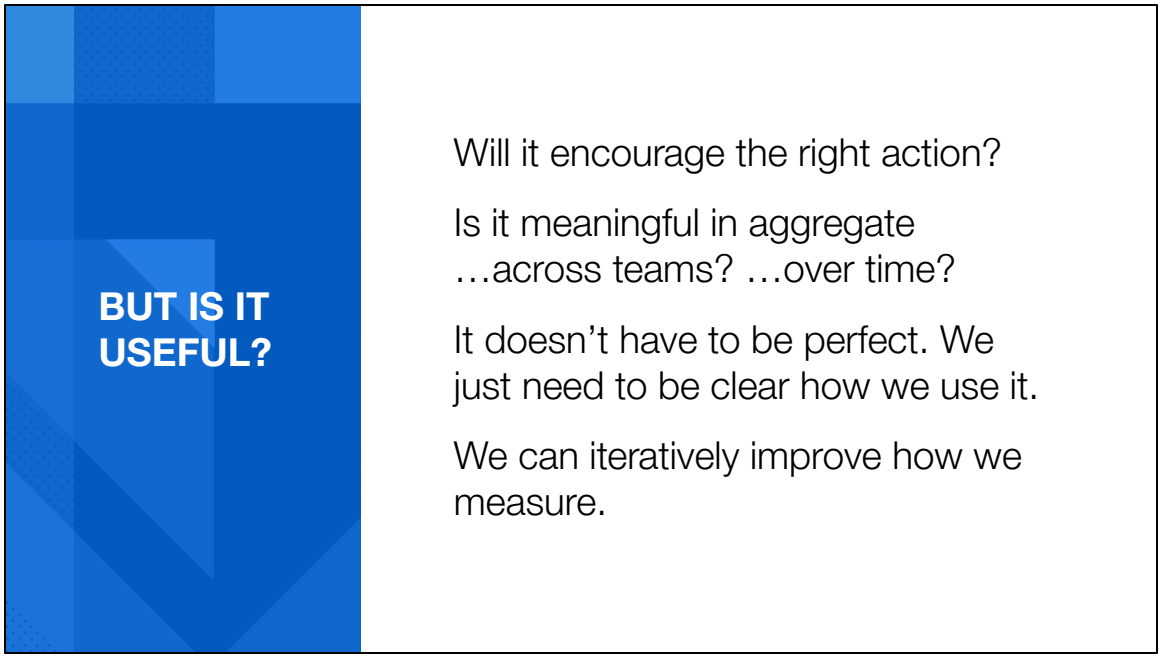
Monitor related metrics and dig into the data. We expected to should see increases in deploy throughput and experiment throughput as DLT decreases. And watching a metric like major bugs reaching production will let us know if we're sacrificing quality in the name of going faster.



The answer to this concern was easy: you won't be evaluated based on your individual DLT.

Instead, we focused on what teams could accomplish together.

Lots of people found issues with the way we measured DLT. It certainly wasn't perfect.



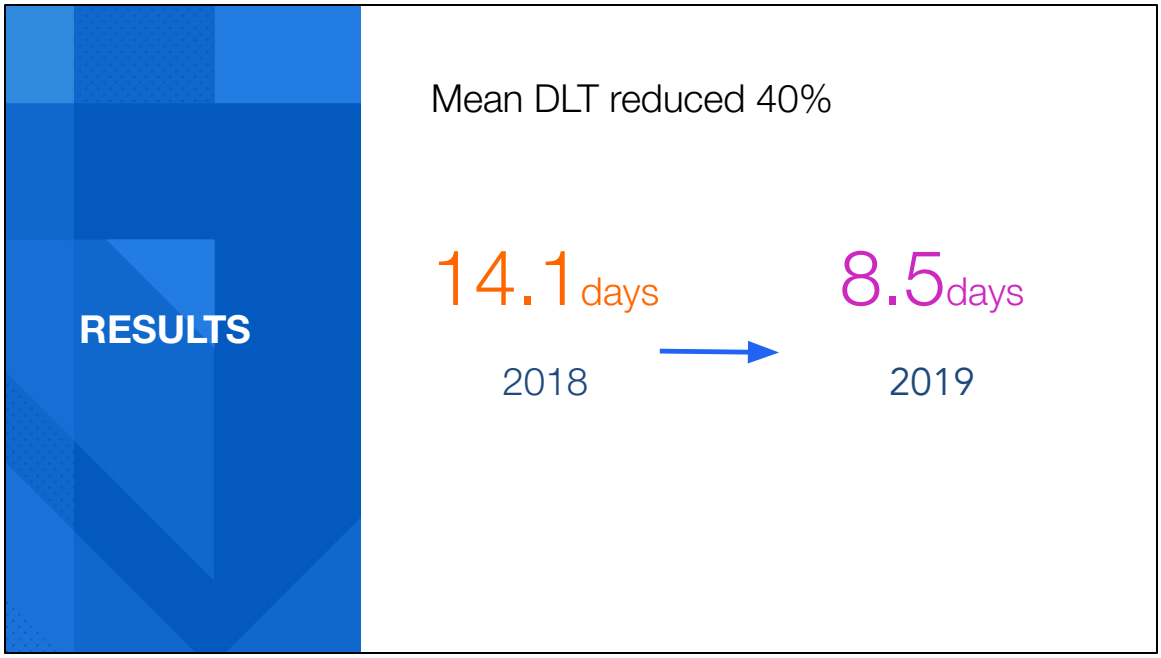
But people were usually convinced that despite not being perfect, it was still generally useful.

You may be wondering if we hit the goal.

Time to answer the burning question, did we hit the goal?

We didn’t reduce company-wide DLT to 7 days on average. But...

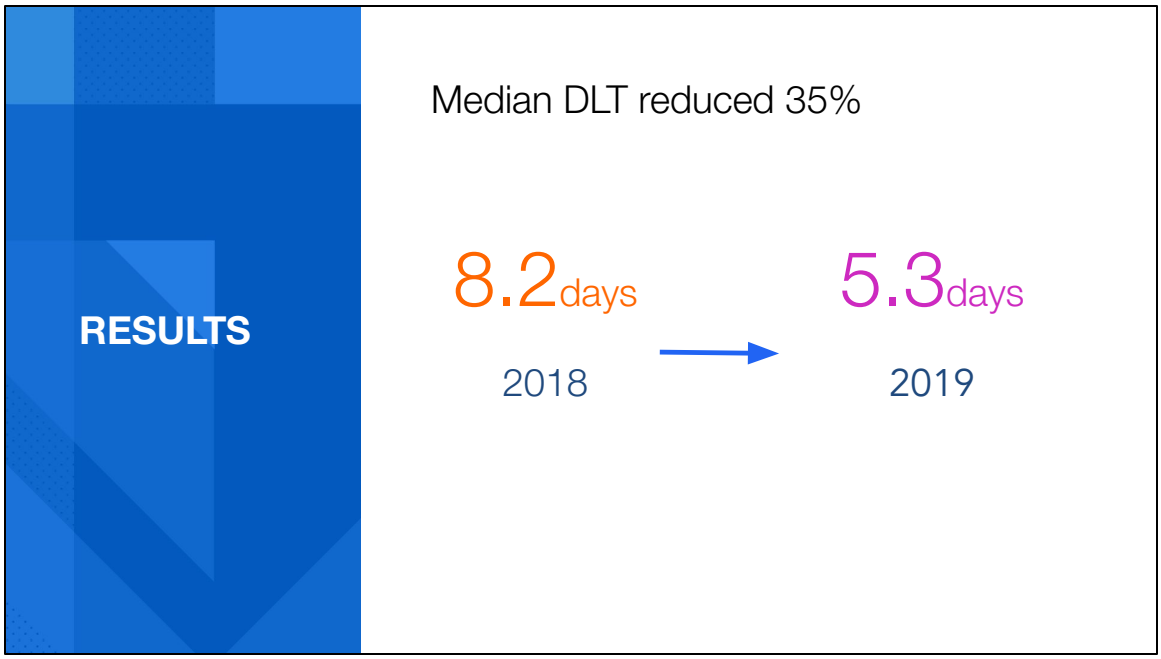
Nope.



We did not, but we were very happy with the reduction we accomplished.

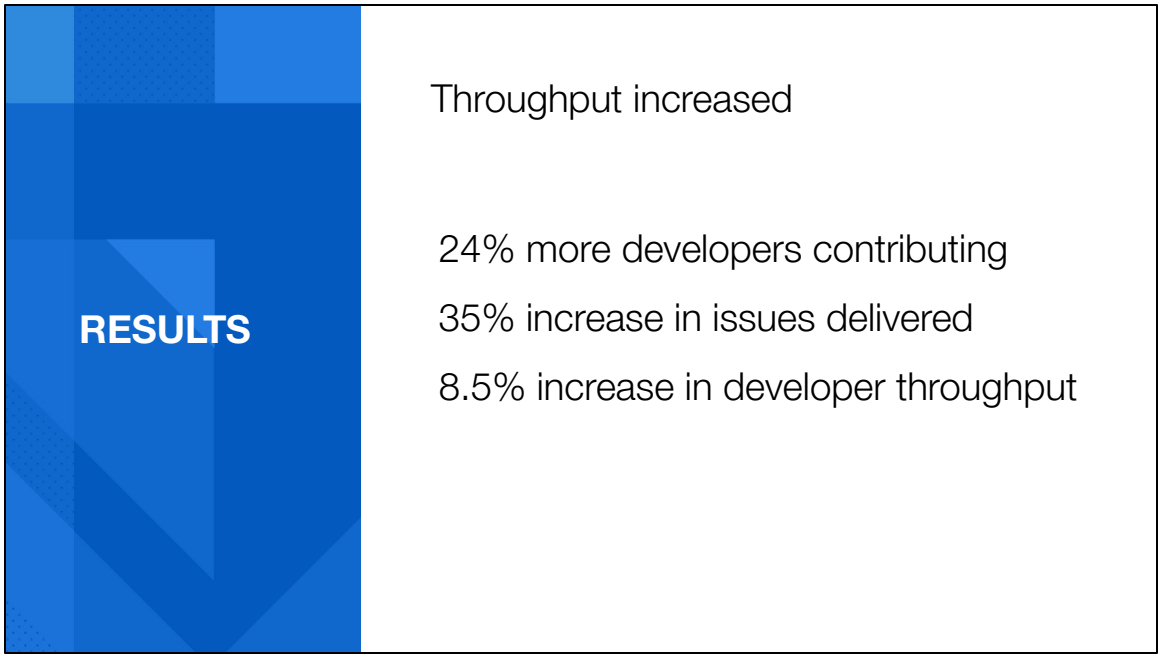
12 weeks ending 2019-12-15:

- DLT 8.5 days (from 14.1; 40% decrease)
- 35% increase in # Jira issues
- 24% increase in assignees, 8.5% increase in issues/assignee

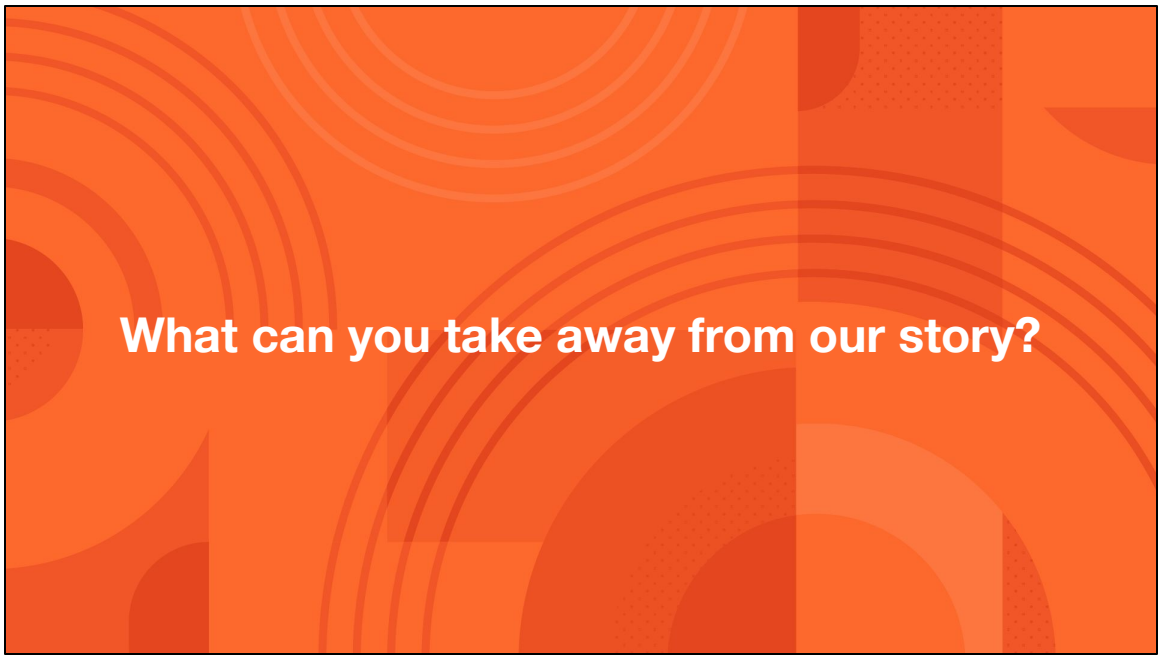


I was especially happy with the median reduction.
12 weeks ending 2019-12-15:

- DLT 8.5 days (from 14.1; 40% decrease)
- 35% increase in # Jira issues
- 24% increase in assignees, 8.5% increase in issues/assignee



And I'm very happy to report that we reversed the trend, and started increasing our throughput.


What can you take away from our story?

PUT VELOCITY IN CONTEXT

What does it mean for your domain and your teams?

How do your teams feel about it?

How will you measure it?

Do you need to use metrics to drive improvement?

PRACTICE KAIZEN

Focus on continuous improvement

Develop metrics & iterate on them

Monitor for undesirable outcomes

Celebrate all the wins

Thanks!

indeed® engineering

go.indeed.com/indeed-eng-blog

blog, talks, open source, and great ideas

Special thank you to my colleague Siva Dosapati, who helped create a previous, longer-form version of this talk that we delivered at a conference last year.