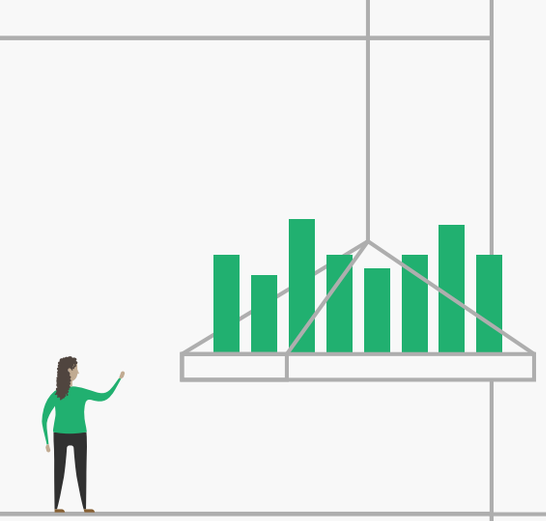
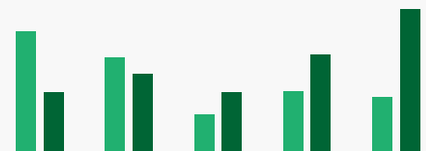
How to Build a Team that Demands Metrics

Rob Zuber, CircleCI, CTO





circleci

If you're an engineering manager, you know metrics are important. But how do you know which metrics are the most important? Which metrics should your team be tracking? And how do you get your engineers to care about metrics and use them to improve their work?

The fact is, managers care about metrics because they have to. Developers, on the other hand, are often suspicious that metrics are only being used to evaluate their performance, and that they're not always an accurate representation of their work.

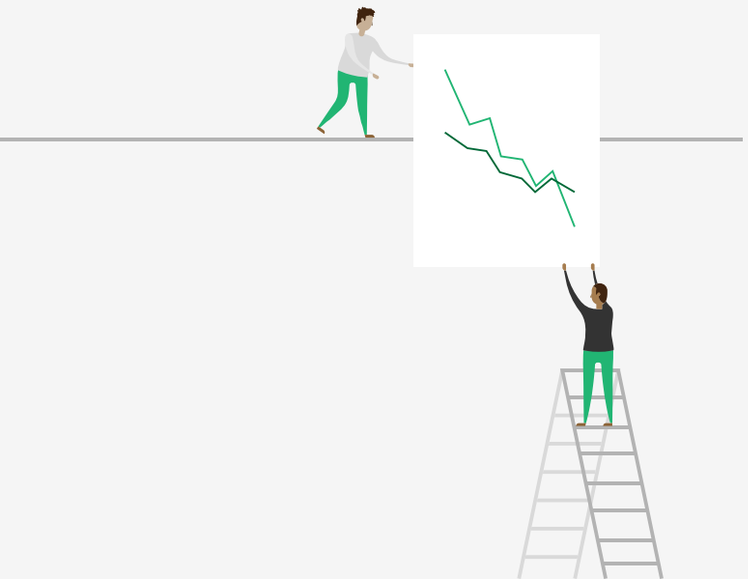
When we talk about metrics in software delivery, a lot of developers think of execution metrics — things like throughput, delivery, and number of deploys. But in reality, those types of metrics don't motivate anyone — at least not without paying attention to something else first: how those metrics and the work engineers do every day are connected to the mission of the company they work for and the customers they serve.

I've worked in software for 23 years. I'm a three-time founder and four-time CTO, and for the past 7 years, I've led the engineering team at CircleCI. I'm passionate about metrics because they reliably tell me if I'm succeeding. They also help me understand when I'm having an impact and how to stay focused on what matters in my work.

When engineers understand how their day-to-day work directly impacts the company's goals and vision, they're able to understand their impact too, and they want to do better. When they want to do better, they want metrics to help them get there.

In this ebook, I'm going to tell you why, and how to get your engineering team to demand metrics.

Clarify your goals



If you want your developers to not only track their metrics but be excited to use them to improve their work, there are a few things you need to do first:

- Get clarity on the goals you're trying to accomplish as a team and as an organization
- Figure out how you're executing those goals
- Find out what's getting in the way

Managers and executives should be focused on business outcomes. At CircleCI, we use the Objectives and Key Results (OKRs) framework to communicate business goals across the org. Are you trying to drive costs down, increase engagement, or trying to increase average revenue per customer? This is the business-level view.

Business outcomes should support the vision and strategy of the company, and your team should understand them without needing to track them individually.

Not every engineer needs insight into operational metrics, like whether you're meeting all your deliverables on time. Rather, you need them to understand what the company is trying to achieve. At the end of the day, what is the outcome you're trying to deliver to customers and for the business overall?

If your engineers have clarity on the company's mission and vision, they understand how their work rolls up into the bigger picture. Your team will be motivated to get better and better at delivering because they can see the direct impact.

And when you have that motivation to get better, you have a desire to understand how your team is operating — that's where the demand comes in.

Individual contributors will start wondering how they're supposed to get better if they don't have access to team metrics. They're going to ask for access to more information so they can diagnose their own problems.

Are we spending too much time on technical investment? Does it feel like we have too many incidents, issues, and escalations from customers? Are we far off on our estimates? Does it feel like we just can't get software deployed? Your team will want the answers to these questions so they can determine which metrics to track at the team level.

The ultimate goal of tracking metrics is assessing how your team can be better. Agile methodologies, which we practice at CircleCI, are all about continuous improvement — particularly the rituals around retrospectives. No matter where you are on the spectrum of hitting your goals, you should be continually reflecting on your progress and asking yourself and your team how you can be better. You should be striving to learn, grow, and move faster because there's always room for improvement.

Don’t make these mistakes

A lot of engineering organizations try to organize whole departments around the same execution metrics. I often hear about engineering managers telling their developers to fix their execution metrics. But from a developer’s perspective, execution metrics aren’t the real problem. Rather, they should be used as a tool to diagnose the real problem.

Velocity, or throughput, is a great tool to expose other issues, but consistent velocity is not a guarantee of delivering business value. In fact, targeting consistent velocity is likely to result in tradeoffs you didn’t anticipate, like taking shortcuts to get things out the door.

Treating the metric itself as the goal usually means your goals are not aligned with the company’s mission and vision. Organizing around business value is what really helps your team understand the purpose of their work. If you’re just rallying around how many times you can deploy per day, then sure, you can be an amazing software-deploying machine — but you might be deploying changes all day that have nothing to do with the actual goals of the business.

At large organizations, the challenges are going to vary from team to team. The team working on billing might have very different challenges than the team that owns job orchestration. In some cases, their main problem is stakeholder alignment, in some cases, it's that engineers can't carve out the time to invest in resolving technical debt — there could be a million different types of roadblocks.

When everyone has clarity oriented around the business goals and anyone can ask for metrics to help them understand how they're doing as a team, you have better insight overall. It's about giving teams a high-level objective and letting them determine what metrics they need to push, instead of giving everybody the same metrics and telling them to get better.

Engineers can always move faster but if they don't have the autonomy to make their own decisions, then they're just part of the machine. And you don't ever want your developers to feel like they're just a cog in the machine. As a knowledge worker, it's not particularly motivating to feel like your knowledge is of no value. As engineers, and as humans, problem-solving is what motivates us, along with knowing that our skills are contributing to a larger, shared goal.

A team that demands metrics: putting it into practice

There are a few different paths that are essential to getting your team to understand and appreciate business-value metrics. At the company level, executives will introduce those high-level goals at all-hands meetings — where the company is trying to go this year, how you differentiate in the market — the things that really matter to a business.

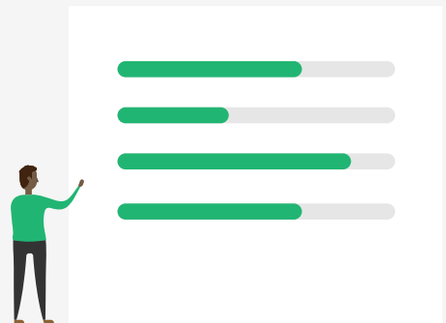
One level down, like what I do at CircleCI's engineering org meetings, is where you communicate how teams should think about those goals from a product and engineering perspective, while still thinking about them within that overall business context.

It's also important that this is understood down through the management chain. If I'm an engineering manager and individual engineers ask me for more depth on those business goals, I should be able to tell them what we're doing as a business with local-level reinforcement, and how it all ties together.

When you have that knowledge as a team, you know what you're capable of executing because you understand what you're trying to achieve for the company and the customer. At the same time, you have the autonomy you need to deliver in the way that works best for you.

That's when engineers start asking for more metrics because they want to understand how to get there faster.

Focus on goals at the team level



For individual contributors, execution metrics can sometimes feel like Big Brother. They start wondering how they're being measured for their next performance review and how fast they can deliver while feeling like their boss or their organization doesn't understand how many unknowns there are in the world of software development.

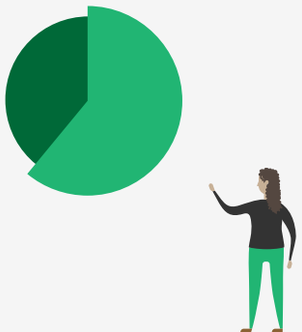
This results in a bunch of debates and arguments that are not useful or valuable. Instead, it's more important to focus on team-level goals.

Teams should have the freedom to adapt when they see things getting in their way. In Kanban or flow-based development, we only allow a certain amount of work to be in progress at any given time. If you hit that limit, members of the team who don't have something to work on directly will go help someone else. Clearing the blockages is better for the team than for every person to be working on something different.

Thinking about metrics at a team level also creates a team bond. You want to know how you're doing as a team, rather than pointing the finger at individuals. Maybe someone is new and doesn't understand how to do a particular thing yet. But you're all trying to get the same thing done so you're going to help that person because it makes the whole team better. Team-level metrics are also motivational as an individual. You want to keep your numbers up because others are relying on you.

Understanding what you're driving towards and having real ownership over that gives you the opportunity to optimize yourselves as a team to get it done.

Intrinsic vs extrinsic rewards

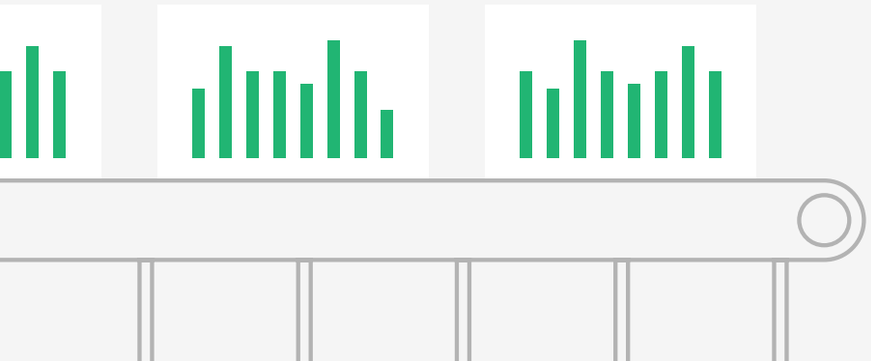


One of the things about business outcomes that can be challenging is extrinsic motivation. You want to set stretch targets because people faced with stretch targets tend to push themselves to achieve the challenge.

For example, if your goal is to reduce your mean time to recovery from 55 minutes to 50 minutes and you can easily get to 50 minutes, you're going to hit that number and be pretty satisfied. But if you set the goal at 45 minutes and the best you can do is 47 minutes, getting to 47 is still way better than getting to 50. Even if you didn't hit that stretch goal of 45, you've still achieved more by aiming for it.

You need the safety to be able to push yourself to those stretch goals and that safety comes in part from the right approaches to motivation — the flexibility to fail.

If you’re not hitting your goals, you should be asking for the data



Real failure is not learning from failure. It should be okay to not hit your numbers. But to not hit your numbers and then do nothing about it is the wrong approach. You need to figure out what you didn’t understand, what got in your way, and why.

> ***The real failure is not learning from failure.***

If you’re starting to see a pattern of incidents, you’re going to need to build in enough slack to diagnose and fix a deeper issue, while still having the bandwidth to handle more issues as you get that resolved.

Estimation is hard. Trying to plan work is hard. But if you can study how you’re doing the work, there are different models you can use to interpret historical data. If you know how much you were able to deliver last quarter, you can focus on getting something similar done this quarter, instead of assuming this will be the one quarter where there are no interruptions, your initiatives will all be perfectly defined, and nobody will get sick.

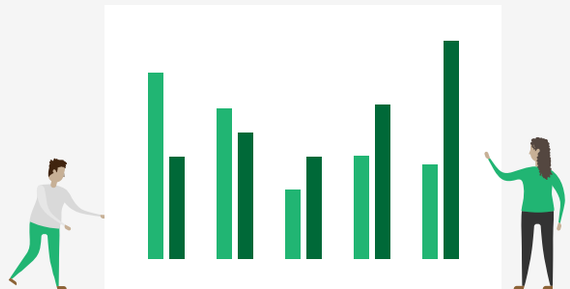
If you're setting business goals and not achieving them, you should be motivated to figure out how to set better business goals. Get up every morning, look at the numbers, and if you're not there, ask yourself and your team how you're going to get there.

Some of that comes from experience. If you know what you were able to do for the last seven quarters and how much you've increased your output, don't create a goal for your team that is completely outside of that. Look at the data and understand what your biggest impediments are. If you fix that thing, you'll very likely hit your numbers.

Building software is a constant state of design. It's not as numerically driven as everybody would like it to be. You're assembling a bunch of pieces off the shelf but you're also designing new ones as you go. The way you plan for that is by remembering that while you're creating new things all the time, you're not redesigning the entire system from the ground up.

If you're not hitting your business goals but you're not looking at your historical data, then you're missing your opportunity to improve.

How to get your team truly passionate about metrics



I think the problems we work on at CircleCI are extremely interesting and I always want to learn more about those problems. We're a company that builds tools to make the lives of software engineers better, and as a developer and a CTO, it's pretty easy to get connected to that mission.

Feeling that connection to the mission of your company might not look like that for everyone. But the bigger problem is that we so often get dragged down into the mundane tasks and challenges of our day-to-day work. And when you need to reverse-engineer a former teammate's code, you're less likely to think about all the awesome software engineers who are able to deliver great products because of the tool you're building.

One of the startups I founded early in my career was an online marketplace for women's clothing and accessories. I was not the customer but I found the work academically interesting.

The first time I started using continuous deployment (CD) at that previous company, I was truly amazed at how quickly it allowed us to put new functionality into the hands of our customers. We could instantly find out if they liked it, and if they didn't, we would build another version tomorrow.

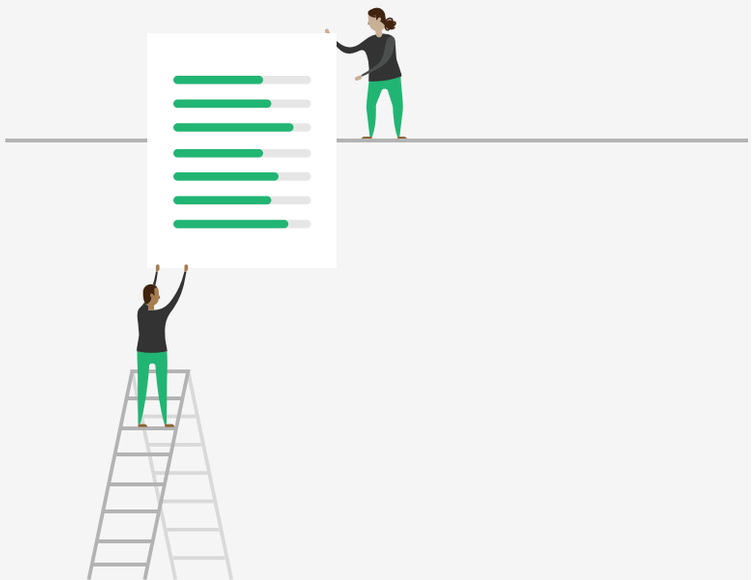
As an engineer, CD is a great way to feel more connected to your customers because you get to put something you built in front of them the same day you complete it. You also become interested in a different form of metrics: whether users are adopting the features and products you build.

> ***The connection between your work and the ultimate value to the user is much clearer when the feedback cycle is so short.***

Finding a way to connect to the value you're delivering to your users through the work that's being done by you and your team is incredibly important. Thinking about your impact helps you understand that your work isn't just a big checklist. Rather, the sooner you get these things into the hands of your customers, the sooner their lives will be better for it.

With CD, you're more connected to what you're really doing for end-users, as opposed to building something so long ago that you don't even remember it by the time it goes out to a customer. CD allows you to get that feedback from customers directly and really see the impact you have on them. That gets people excited.

There’s no shortcut



You can try to drive an entire organization by execution metrics but it simply won’t be fulfilling for anyone.

If your team is aligned around what they’re trying to achieve, they will recognize the value of execution metrics. And they will come looking for them.

There’s no shortcut.

One of the biggest challenges as a leader is that you always have a lot of context around the decisions being made and the goals being set. You may feel like it’s easier to just process that information and convey the results to your team, rather than bringing them along on the journey. But the likelihood of alignment through that approach is extremely low.

If I tell my team to make sure they hit their numbers because “I say so” and they should “just trust me,” that’s not going to work. Similarly, if you show up and say they need to have 27% allocation to bug fixes and 32% allocation to new features because that’s your diagnosis of the problem from looking at the dashboard, you’ll get people allocating their time that way, but no actual benefit to the business.

As a leader, you shouldn't read our 2020 State of Software Delivery report and go tell all your teams to make sure their builds are 3.5 minutes long and they deliver 7.3 workflows per day. Those are all tools to diagnose where your problem might be, as opposed to the secret sauce of a great organization.

You need clarity on the team goals, people need to understand the business metrics, and be motivated to deliver because they're excited about what they're doing. Don't assume if you hit specific metrics, your numbers will start to look a certain way.

If you start with the reason and you end with the numbers, you know you're on the right track. You'll know you're getting there because you'll see the numbers moving, your team will be excited by that, and they'll be motivated to do even better.