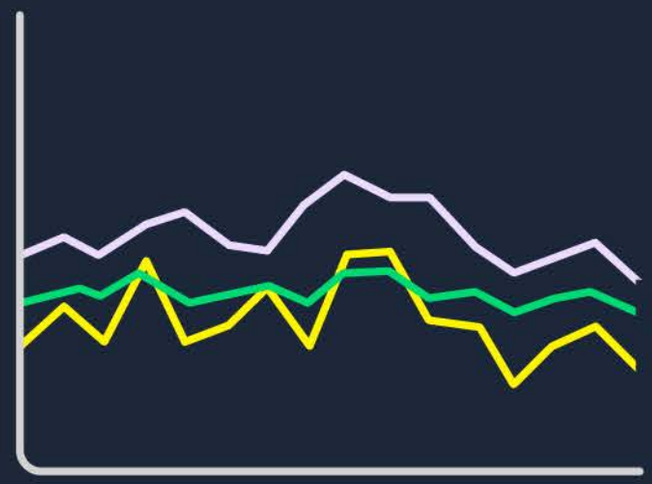
CI/CD Metrics for Platform Engineers



circleci

Software engineering is the driving force behind every enterprise's innovations and the heart of today's competitive business landscape. Unlike years past, when software delivery might happen once or twice a month, modern software delivery is an ongoing process. The competition is high, and the pressure is on for every company to deliver high-quality products quickly and effectively.

Platform engineers play **a crucial role** in enabling their organization to deliver quality software quickly, safely, and at scale. But shipping complex code with less time between releases is challenging and only happens when engineers are paying close attention to metrics.

Platform engineers agree that metrics matter, but it can be difficult to know which metrics are the most critical in helping your team improve its ability to build, test, and deploy quality code consistently.

To answer this question, the team reached out to engineering leaders to find out which metrics they actually monitor daily to guide and optimize their team's work.

This guide will break down the most meaningful engineering metrics and how you can use them to improve your team's performance.

The most meaningful metrics for platform teams

After speaking to a broad range of engineering industry leaders, and poring over **nearly 15 million data points on our own platform**, we found that the most insightful and relevant metrics fall into three main categories:

- Velocity metrics
- Morale metrics
- Business metrics

Velocity metrics

Velocity metrics measure the speed and efficiency with which software moves through a delivery pipeline. Watching these metrics helps platform engineers identify slowdowns and find ways to optimize overall performance.

VELOCITY METRICS INCLUDE:

- Duration
- Throughput
- Deployment frequency
- Cycle time or duration
- Change lead time
- Sprint velocity
- Velocity over time
- Sprint commit to achieved ratio

> Looking into key velocity metrics helps our engineering teams justify trends we see. In other words, these metrics are making them think about their own performance and learn what behaviors have an impact on metrics."
>
> — **SERGIY TUPCHIY,** Former Engineering Manager at Contentful

Duration

DURATION MEASURES THE LENGTH OF TIME IT TAKES FOR A WORKFLOW TO RUN.

Duration, similar to cycle time, measures how long it takes for a workflow to go from beginning to a complete status. Essentially, it tells you how long a single build takes to get a pass or fail.

Platform engineers measure duration for a couple of reasons. The first is that decreasing duration helps create a fast feedback cycle. When we track duration correctly, we decrease the chances of losing our developers' attention as they wait to see if a workflow passes or fails. For a fast feedback loop, we need to get the signal back as fast as reasonably possible. Testing takes time but the better we test, the better our test output. The better our test output, the faster we can fix failures because we have the information we need without needing to debug.

Tracking cycle time can help with allocation of the right team members in the right places. Moses Mendoza, former Head of Engineering at data processing and review platform Zapproved explains, "Cycle time helps our team when we are arguing about things. For example, if I'm trying to prove that a category is understaffed or if one team or another doesn't have enough support, cycle time will show where the issues sit in a product review for longer than they sit in development. This tells us if we need more staff members reviewing products so they can be pushed to development. It's a nice data point to reference at a systemic level."

Throughput

THROUGHPUT MEASURES THE AVERAGE NUMBER OF WORKFLOW RUNS PER DAY.

Throughput is a direct measurement of commit frequency, showing how often workflows were started. No matter the number, measuring throughput helps you establish a baseline of commit activity. When you know this average number, it makes it possible to forecast engineering productivity, monitor fluctuations in this baseline number, and understand upsets and when work didn't get done.

While throughput will show you how much work is going through the system, it's critical to note that it measures all updates, including small and inconsequential ones.

Understanding how many workflows run per day establishes your baseline, but measuring throughput alone won't always tell you how much value is delivered or whether a deployment occurred. You can push small, inconsequential changes and still see increases in throughput. Throughput tells a story of quantity, not quality.

Mendoza says, "Throughput helps us identify and understand speed, but the throughput of a system is also bound by its primary constraint. In other words, throughput will show you what the slowest issue is in a chain of events, but it won't show you how to fix it to speed up your work."

Due to the potential for throughput to tell a misleading or incomplete story, it's important for platform engineers to customize throughput measurement to their teams.

At CircleCI, for example, many development teams practice pair programming, which means throughput can't be tied to an individual's productivity in any way. Additionally, work in progress limits (i.e., no more than 6 tickets open at any one time) can also end up creating an artificial ceiling on throughput.

While throughput is a valuable metric that helps you track output, there is no one-size-fits-all way to measure it. Measuring throughput accurately requires you to evaluate the structure of your development teams and how they work..

Deployment frequency

DEPLOYMENT FREQUENCY MEASURES HOW LONG IT TAKES TO DEPLOY TO PRODUCTION (OR ANOTHER SIGNIFICANT MILESTONE IN YOUR CD PIPELINE)

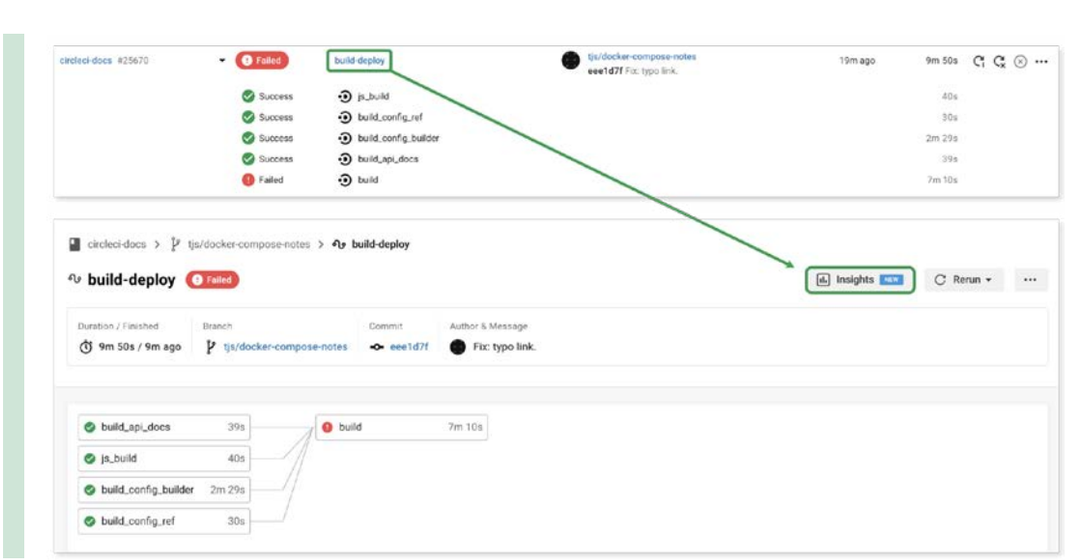
In research from our **2023 State of Software Delivery report**, we found that it was critical for delivery teams to keep their software deploy-ready, so that the organization can ship updates to users as often as the business requires. To capture a baseline for this demand, and the team's ability to meet it, many platform engineers track deployment frequency.

Deployment frequency measures how many units of work were actually released into production, not just how many ran through the system. Like throughput, deployment frequency is not a perfect corollary to value creation. Deploying many updates per day doesn't always translate to meaningful, high-quality work, but tracking deployment frequency can help your team measure speed and progress over time.

For example, if you can see through tracking deployment frequency that code deployment is happening less frequently, you can look for ways to fine-tune your workflow, adjust tools, eliminate unnecessary steps, and, ultimately, help track your team's efficiency and growth.

Juan Pablo Buriticá, former VP of Engineering at Splice.com, explains that **measuring deployment frequency** can give you a baseline and help you monitor how long it takes you to deploy over time. Buriticá says that even if a baseline is bad, it acts as a realistic frame of reference and can help you move forward.

The Insights dashboard in CircleCI allows you to monitor metrics related to your workflows, like deployment frequency, in order to do just that. After establishing baselines using a performance dashboard such as CircleCI Insights, you'll be able to track when you've made changes that have improved your numbers.



Change lead time

CHANGE LEAD TIME MEASURES HOW LONG IT TAKES TO DELIVER AN IDEA OR HOW LONG IT TAKES TO DELIVER CODE TO PRODUCTION FROM START TO FINISH.

Alex Bilmes, former VP of Growth at Puppet, explains two different ways to measure change lead time. Bilmes says, “One way to measure change lead time is to look at how long it takes to get an idea out and for the idea to go full cycle. The other way to measure lead time is to look at deployment lead time, which measures how long it takes to get to production after a developer has pushed the change to production.”

Measuring lead time is helpful because it identifies slow-downs in your processes. As you monitor lead time, you can streamline and optimize both the software delivery part of the lifecycle and how long it takes to get an idea to production.

In terms of change lead time, Bilmes recommends breaking up lead time into stages to get more granular, monitor over time, and look at the whole life cycle of an idea. The process of taking a few metrics and tracking them effectively helps you learn where things are slowing and how you can become more productive.

Bilmes also recommends that engineers track change lead time over a period of time to monitor team efficiency. “You want your time between an idea, turning that idea into code, and then moving that code into production to be as short as possible,” Bilmes says.

Continuous delivery suggests that validated code is automatically shipped to production and made available for users. However, changes are not necessarily visible to end users until traffic is turned on for those features. Teams at CircleCI often measure change lead time from when work begins on a feature to when the feature flag gets turned off. This helps developers and platform teams think more holistically about the software feedback loop as opposed to focusing on when code is merged and tickets are closed.

To get the most benefit from change lead time measurements, it’s important to consider the entire delivery cycle. Focusing on ticket completion means you’re tracking the delivery of small pieces of work, which doesn’t necessarily tell the whole story of a funnel or flow. A more customer-oriented perspective on throughput and change lead time will capture the time required to make a full feature available to end users.

Sprint velocity

SPRINT VELOCITY IS A MEASURE OF THE AMOUNT OF WORK A TEAM CAN TACKLE DURING A SINGLE SPRINT. IT IS USED FOR PLANNING AND MEASURING TEAM PERFORMANCE.

Sprint velocity is an excellent metric to use for planning sprints. “Let’s say we’re mid-term planning,” Mendoza says. “If velocity went down in the first two sprints, then we’re going to start looking at the next four sprints. If our velocity trend line is not on track to achieve our ultimate target, we know we need to let the business know we are going to be late. Whereas, if the velocity trendline is consistent or going up, then you know you’re on track.”

A higher velocity isn’t always an indicator that your team is doing well. Measuring velocity just shows whether or not you’re on target for hitting promised milestones.

Tom Forlini, CTO at Livestorm, dives deeper when measuring velocity. His team breaks velocity into three smaller metrics, including:

- Number of issues and story points
- Number of issues done vs. what was planned
- The percentage of issues distributed depending on their type

Forlini explains, “The number of issues and story points metric is critical during our sprint planning. At Livestorm, our engineers work on two-week sprints and have 50 story points per sprint. We track the number of issues done vs. what was planned because it gives us a good indication of the sprint planning quality between Product and Tech. Finally, we look at the percentage of issues [by] type (e.g., bug, improvements, new features, refactoring, etc.). For example, when a sprint contains only new feature issues, we know from the start that it might be quite a challenging sprint to tackle. Ideally, you should try to balance the type of issues by sprint as much as possible.”

For Forlini, it’s essential to track these three metrics to measure velocity. These metrics help his team define a short-term roadmap, and help them plan sprints in advance. Not to mention, these three metrics work together to identify flaws in the team’s processes and the technology platform that enables them to do their best work.. For example, if an engineer finished the sprint in advance or too late, it might indicate that the alignment between the engineering team and the product team should be improved. Missed targets might also be a symptom of unnecessary friction in the development process that the platform team can address through automation and shared tooling.

Velocity over time and sprint commit to achieved ratio

VELOCITY OVER TIME MEASURES THE WORK A TEAM COMPLETES DURING A SOFTWARE SPRINT OVER A SPECIFIC PERIOD OF TIME.

SPRINT COMMIT TO ACHIEVED MEASURES HOW MUCH A TEAM COMMITS TO ACCOMPLISHING VS HOW MUCH IT ACTUALLY ACCOMPLISHED.

Velocity over time and the sprint commit to achieved ratio are excellent metrics to measure in relationship to one another.

Mendoza explains how his team used these two metrics to track progress and make quarterly plans.

"We measure velocity over time—not across teams, but velocity over time within an individual team. As we measure this metric, we're measuring both their ability to plan a sprint and then execute against it, as well as the general trend line of how much work they're accomplishing based on their story point estimates."

He continues, "We used the sprint commit to achieved [metric] to do planning, not dissimilarly from other teams that do sprint planning. Other teams use sprint commit to achieved to help inform how much they're going to do and to react to the previous sprint's information. We glean useful information if there was a huge gap in what was committed versus what was achieved. With these metrics, the scrum master retrospects on what happened and why."

Velocity over time combined with sprint commit to achieved allows teams to do rough quarterly feature planning.

Morale metrics

Morale metrics track how developers feel about the quality of their work and whether or not they are happy at their job. Platform engineers pay close attention to developer sentiment, as it indicates how effectively platform tools and services address common pain points and empower developers to do meaningful, impactful work.

MORALE METRICS INCLUDE:

- Code quality confidence
- Employee morale
- Transparent solutions over heroic individual action

> “We try to use metrics to identify issues when we fail [as a team]. That’s why breaking them down on auxiliary components helps us spot actual behaviors which led to those mistakes and [help us] try to fix those in the future. We also review them on a regular basis within teams and with senior management to react in an agile manner.”
>
> — **SERGIY TUPCHIY,** Former Engineering Manager at Contentful

Code quality confidence

CODE QUALITY CONFIDENCE MEASURES HOW GOOD OR BAD DEVELOPERS PERCEIVE THE CODE TO BE.

Code quality confidence is a telling metric. It's a leading predictor of escaped bugs and a trailing indicator of employee burnout, as well as a sign that a sprint was potentially overloaded.

Mendoza explains how his team at Zapproved used this metric. "We reviewed every sprint by the team in conjunction with their manager and their scrum master. As we measured code quality confidence over two or three sprints, and if we saw code quality tanking, it meant something was wrong in terms of the teams planning their individual investment with the work," says Mendoza.

Falling code quality confidence is also a sign that your team may be on the way out the door. "We have a belief that the best engineers are the first ones to leave when they don't believe they're making good stuff," Mendoza continues. "Your true craftspeople want to hone their craft. If they think they're sputtering out, they're not going to stick around."

The CircleCI delivery team also measures work by confidence. While test coverage and code coverage are essential metrics that support confidence, there is a larger story that platform teams can capture about developers' overall confidence to produce quality work.

When it comes to measuring code quality confidence, it's important not to over-rely on code coverage metrics. Having 80% or 90% code coverage only to find out the code is broken after it ships can undermine developers' confidence in their code and development platform.

Essentially, test coverage is a partial proxy for code confidence. If you know 95% of your code is fully tested, versus 20%, then you're going to feel pretty confident that if your tests pass, your code is legitimate.

At CircleCI, development teams are encouraged to focus on delivering small iterations quickly. This provides the confidence that they are building something of quality, nothing is broken, and they've made the right choices in building features for our customers.

Employee morale

MORALE IS A METRIC THAT MEASURES THE OVERALL SATISFACTION OF EMPLOYEES AT WORK.

Tracking code quality confidence isn't the only metric that shows how employees feel about their work. Measuring morale is another helpful way to determine if your development platform supports developers in feeling happy, productive, and proud of their work.

While it isn't easy to accomplish this in a dashboard tool, it's possible to track morale quantitatively. Distributing a survey or questionnaire gives developers the freedom to express their thoughts and ideas.

Mendoza explains how Zapproved tracked morale to monitor retention. He says, "We measured morale at work by responding to surveys, having more conversations, and asking managers to dive deeper in one-on-one meetings to find out how employees felt."

If responses are overwhelmingly positive, you'll want to know what is working and how to replicate that positive work environment. Often this will show up in higher adoption rates and usage of platform tools. Similarly, if responses are negative, it's helpful to find out directly from your team why they feel that way and what you can do to fix the problem.

Transparent solutions over heroic individual action

Another value CircleCI uses to create a productive and happy development ecosystem is to focus on transparent solutions over heroic individual action.

Platform engineers should strive to create an environment where developers have deep visibility into the state of everything they are working on. It should be clear where teams are in their process, what tools are available to support their work, and how team members can access and use those shared tools in a collaborative and efficient way. This is much more important than having one developer building something in isolation and gives teams a sense of ownership and shared responsibility.

Platform engineers can use a couple of different approaches when it comes to measuring openness and transparency, including time-to-PR and gathering feedback.

Measuring the time that a feature or ticket moves from being in progress to having the first PR up is an indicator of working openly and transparently. It shows how well the organization's technology platform and processes support rapid movement from feature development to opening a pull request and having other team members' eyes on it.

It's also important that the organization's tools and structure facilitate collaboration and feedback early in development, before changes are formalized. When more team members are involved and individual engineers get feedback, it helps teams stay on track, minimizes the amount of work that has to be done, and allows engineers time to incorporate feedback to produce more polished work.

Business metrics

Since everything a platform engineer does should propel the company forward, it's also essential to track business metrics.

BUSINESS METRICS INCLUDE:

- Growth of the company
- Funnel metrics
- End-user value

> We're in a process to align with bigger business goals and reviewing the impact of current metrics values on deliverables together with leaders in our company. And since we are in a growth mode right now, we would like to see how those metrics could support us in scaling efficiently."
>
> — **SERGIY TUPCHIY,** Former Engineering Manager at Contentful

Growth of the company

THE GROWTH OF THE COMPANY TELLS YOU HOW QUICKLY A BUSINESS IS SCALING.

Platform engineers don't only look at metrics that pertain to individual processes or team efficiency. Looking at overall business metrics will allow your team to accommodate for user growth effectively.

Yixin Zhu, formerly of Uber, explains that while it's essential to look at engineering execution metrics, it's also important to be dialed into the business's overall goals and to measure the growth of the company.

"When you're talking about [doubling every six months], you have to be tracking that to know what you need to build, what sort of degradations [to expect], how many data centers you need, how many boxes, etc.," Zhu says.

In short, platform engineers have to keep an eye on real-time business metrics to project out and plan appropriately. "You can't just be reactive; you also have to be proactive," says Zhu.

Funnel metrics

Platform engineers also need to look at the big picture to determine how users are actually behaving and whether or not the way they behave is positive for business.

Investing in funnel metrics is one way to accomplish this. Funnel metrics give you a comprehensive look into the whole data story. In other words, they tell you how the changes you're making affect the way users engage with your product. For a platform team, the focus will be on how developers are engaging with the organization's central technology platform and adhering to the "golden path" of preferred tools and services. For product teams, the focus will be on the experience of the end user.

Zhu offers an example of how looking at funnel metrics pertains to engineers.

"The engineering team [at Uber] is very engaged with the customer experience," Zhu says. "What engineers build directly influences other engineers and businesses. If you're working on B2B software, other engineers and businesses are still your customers. How they are engaging with your product, and how your changes affect that, are critical to understand. You have to think of metrics from the lens that everyone is your customer, and you have to measure that. You have to see the whole story."

End-user value

A SHARED VALUE AMONG ENGINEERS IS DELIVERING CODE THAT IS USEFUL TO THE END-USER.

Platform engineers should be hyper-focused on maximizing developers' ability to deliver value to customers. Ultimately, their work is to transform an organization's engineering department from one that simply ships code to one that continuously delivers value. The focus should not be on pushing code for the sake of pushing code, but on ensuring the organization is delivering changes that are reliable, secure, and create moments of delight for customers.

The team at Livestorm has a similar approach for measuring the quality of their work. Forlini explains how Livestorm's engineering team connects its customers' NPS with the engineering metrics to measure success.

"If our customers report many bugs or low product quality in the NPS, our engineering team needs to act fast to solve those issues. We adapt our roadmap and sprints to have our engineers work on bug fixes rather than developing new features," says Forlini.

Getting the most value out of meaningful metrics

As you create a data story to guide team productivity, don't be afraid to start small. Practice by measuring a few of the most critical metrics correctly. Ultimately, every team and business is different, so the metrics you choose to measure will vary. Start by understanding company goals and business metrics, and select your most critical metrics based on that information. Then, drill deep into those metrics and review them often with your team. Above all, check in with your team regularly to make sure morale is high.

DEFINE BUSINESS GOALS

The first step in gleaning insights from your engineering data story is to define business goals properly. Platform engineers should know what the business is trying to accomplish. When platform teams understand the direction the company is going, they understand what they're working towards. and can better align development processes with business goals.

KEEP MORALE UP

Most metrics show how productive a team is but don't always tell how happy team members are with their work. In addition to measuring development success, it's essential to take a look at employee morale. Consider following the Zapproved model by administering morale surveys and responding to them with one-on-one conversations. Developers that are happy with their jobs do better work and think more proactively about problems.

START SMALL

As Bilmes says, "you just need one to three metrics to get started. Realistically speaking, it's the iterative process of tracking something over time that brings value. If you are intentional about tracking simple metrics, trying new things, experimenting with different approaches, and seeing those metrics change shape, you can add additional metrics over time."

REVIEW METRICS OFTEN

Metrics aren't something you only talk about at quarterly meetings. Your metrics should be at the forefront of every business meeting. Start every meeting by reviewing metrics so the team is perfectly aligned on what you're doing right and what needs improvement.

LOOK BEYOND EXECUTION METRICS

The success of a platform engineering team hinges on more than engineering metrics. Indeed, engineering metrics will help you measure productivity, troubleshoot, and fine-tune your processes, but looking at business metrics will help you understand why you're building what you're building every day. The best platform engineering teams aren't only reactive to metrics; they are proactive in finding out where the business is heading and how the development team can project and plan accordingly for growth.

BLAMELESS CULTURE

Roopak Venkatakrishnan, former Director of Engineering at Bolt says, "We see tracking of metrics as an opportunity to run experiments and look into trends, but never to blame or benchmark people based on those. [We also use metrics] to find techniques, by comparing, which helps to move them in a positive direction."

Platform engineers agree that tracking metrics is integral to the development process of every software engineer.

With all the different metrics available to engineers, it can be challenging to know where to start. According to advice from the top software engineering leaders we interviewed, don't overthink which metrics to track right out of the gate; just start somewhere.

Spend time clearly defining your business goals and how they connect to your team's work. Then, pick three or four metrics that are most closely tied to achieving those goals. Start small and don't aim for rapid transformation.

Building an insightful data story doesn't happen overnight. It happens as you start tracking a few metrics, slowly drill deeper and add more meaningful metrics to your dashboard over time.

With CircleCI, you can make industry-leading CI/CD a central part of your development platform and track critical performance metrics in our Insights dashboard. To get started, **talk to us** or sign up for **a free trial**.