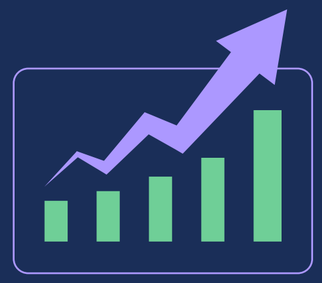
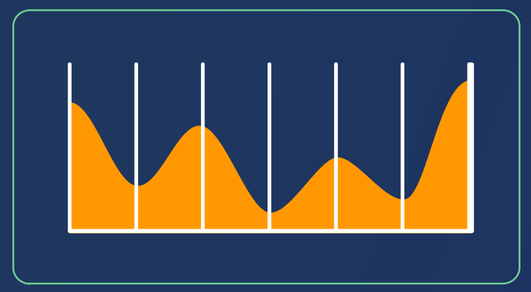
LINEARB



Software Develoment Metrics Runbook

Welcome, Dev Leader. This book has everything you need to start and run your engineering metrics program. Check it out and let us know if you have any questions.



LINEARB

Starting a Metrics Program

A personal note from Dan Lines, former VP of Engineering, Co-Founder and COO of LinearB

Thanks for coming! This site is full of ideas for what you can measure and how you can measure it. Before we get into that, Let's start with the "why".

The first time I ever thought about using data to help run my team was in 2015. The start-up I worked for was growing fast, and overnight (it felt like) I went from leading a single team of 6 devs to leading 5 teams with 50+ devs as VP of Engineering.

I had no experience dealing with an organization of that size. The processes that worked for us when we were smaller weren't scaling. I knew I needed to be doing more to help my team but I wasn't sure what.

I dove head first into the problem by talking to every expert I could find. Other VPs of Engineering, agile consultants, start-up investors – anyone who could offer a clue about what scaling successfully looked like. They all said the same thing, you need data to lead a rapidly growing dev team.

So I reassigned one of our data engineers to do nothing other than compile internal data about how our team worked. I learned a lot and made a ton of mistakes like focusing on individual performance metrics. Spoiler alert: don't do that! And eventually I came up with a runbook that we now use at LinearB to run a data-driven development organization. Everything I learned and everything we use today is here in this web site. I hope it helps.

Thanks for coming,

Dan

LINEARB

3 Metrics We Really Like

Cycle Time

Cycle Time measures the time from first commit to production release. At LinearB we call this our super metric because not only does it show team efficiency, it exposes bottlenecks in the development cycle.

>>Explore more

Rework Ratio

Rework ratio measures the percentage of your code that is being rewritten within 21 days of being merged. It's a predictor measurement of downstream quality issues or an indication that a requirement was missed.

>>Explore more

Investment Profile

Investment profile measures how much time your team spends on different types of work. It helps you determine if you are striking the right balance between customer-facing stories, non-functional requirements, and bug-fixing.

>>Explore more

LINEARB

Scorecard Breakdown

1

These are the LinearB engineering department's top line metrics. Each of these metrics and the weekly trend below them tell their own story. For example we track Cycle Time constantly because it indicates how well our development process is optimized. Shorter Cycle Time means faster time to market.

2

Time Investment is an analysis of the project board story types that have been completed each week. This is a concise way to show business leaders where engineering resources were spent, and how much bugs or hot fixes cost the company in effort.

3

The LinearB Team Performance dashboard displays 17 metrics, but not all are relevant each week. We customize this section every week based on where major changes occurred, topics that need discussion, or results from optimization efforts. Between sections 1 and 3, Engineering Leaders are able to give other business leaders comprehensive updates on engineering activity.

4

The Major Efforts section acts as delivery forecasting talking points for both engineering and product. At LinearB we include the project board stories that have received the most amount of attention during the week as well as any stories with a high risk of delay. This is a great section to help soothe the nerves of a CEO or sales leader asking when a feature is going to be ready.

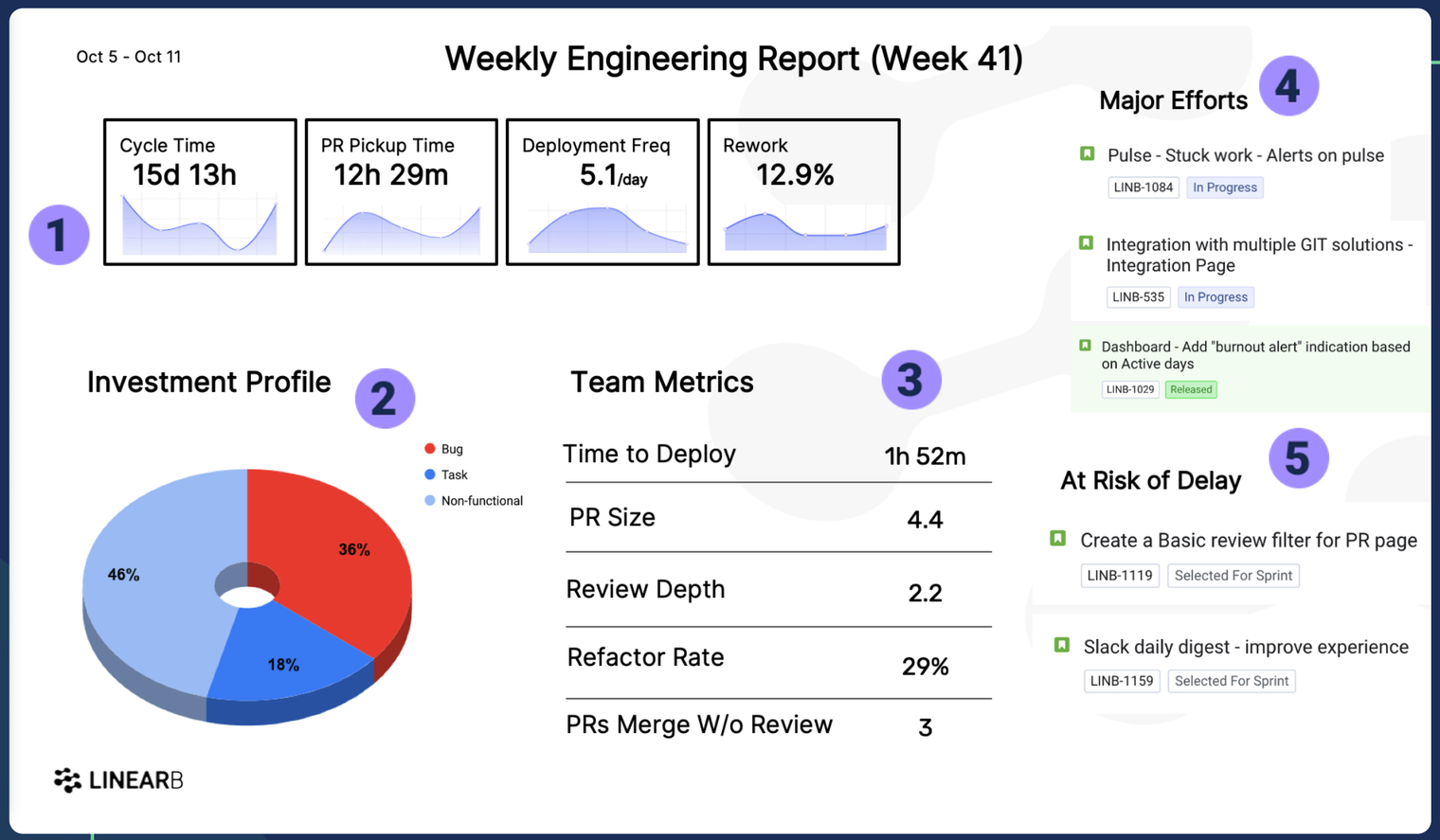
5

The At Risk of Delay section gives you the opportunity to add context to upcoming features that will likely be delayed. You can set the expectation before a delay happens, explain the issues facing the team, and how and when they will be resolved.

LINEARB

The LinearB Metrics Scorecard for Engineering Leaders

DOWNLOAD PDF



LINEARB

Explaining Engineering to Execs
5 Lessons Learned

Think of yourself as a translator

It's hard being the ranking dev leader. Your peers think of you as technical. Your team thinks of you as a suit. You're trapped in between two worlds. And that's our opportunity to provide value to our organization. I learned to embrace my role as translator. My job was to bring business context to the dev team and visibility to the exec team. Without empathy and understanding on both sides, your company will never see the full value in your team and your people will never reach their full potential.

LINEARB

Explaining Engineering to Execs
5 Lessons Learned

Focus on process as well as outcomes

As a dev leader, the #1 question we get from non-technical execs is "when will feature ABC be ready?" Of course delivering new value is our job. But allowing all of the conversations to just be about outcomes is a dangerous precedent. The more my CEOs understood how software is built and delivered, the less likely they were to have unrealistic expectations and more the likely they were to offer help (instead of criticism) when times got tough.

LINEARB

Explaining Engineering to Execs
5 Lessons Learned

Educate your business with metrics

Metrics are already the language of sales, marketing and finance departments. ARR, LTV, MQL, CAC... Your business not only understands what these metrics mean, they also understand how these terms fit in to the overall business process. There's no reason they can't understand key engineering terms and metrics too. I brought metrics to the CEO's staff meeting every single week so to teach my peers about how we work and how to measure our successes and failures.

LINEARB

Explaining Engineering to Execs
5 Lessons Learned

Bring your execs into the weeds

Trust me. They can handle it. It's a big mistake to say "This stuff is too technical. They won't get it." What's technical about Cycle Time? It's just a process with phases that represents time and efficiency. Just like a Sales Cycle, which they completely understand. I'm not say I showed code to our CEO. But I did teach them how to talk to our developers about individual branches, PRs, and the work that gets done in the weeds. Non-technical leaders will like the additional sense of confidence it gives them.

LINEARB

Explaining Engineering to Execs
5 Lessons Learned

Explain the trade-offs

There's nothing the exec team hates hearing more than "we just don't have enough resources to do that right now." It leaves them with a powerless feeling. Instead educate them on the dependencies for building the new feature they want. I always like to start with "yes" and then go through a step by step process to show the team what would need to be moved around to accomplish the thing they want. I found that approach actually led to me getting approval for more hires and more non-functional investment.

LINEARB

17 Metrics for Dev Team Success

Measuring the right indicators will help you accelerate delivery, maintain positive culture, and translate dev activity to business value. So what should modern dev leaders measure?

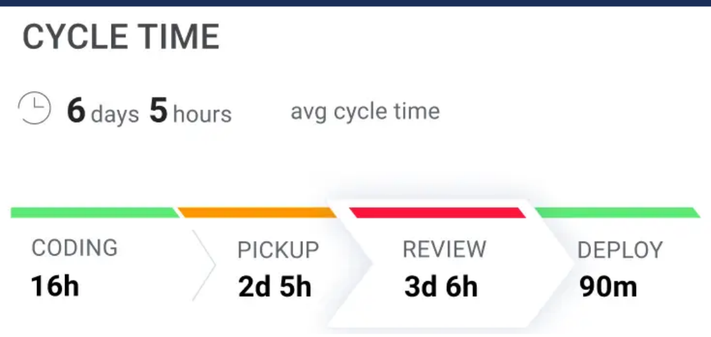
Keep reading to discover 17 metrics that matter for dev leaders and how to use them.

- Delivery Efficiency
- Delivery Quality
- Priority Focus
- Team Health

LINEARB

Delivery Efficiency

Efficient delivery pipelines lead to predictable value delivery, happy developers, happy product owners, and happy customers. Measuring the stages of your delivery pipeline allows for bottleneck detection. This provides a high leverage point to increase your delivery performance because it impacts all teams and contributors



#1 Cycle Time

Definition
The amount of time from work started until work finished

Why it matters
Cycle time is the #1 indicator of your speed to value and efficiency ratio.

>> Learn more about Cycle Time

#2 Coding time

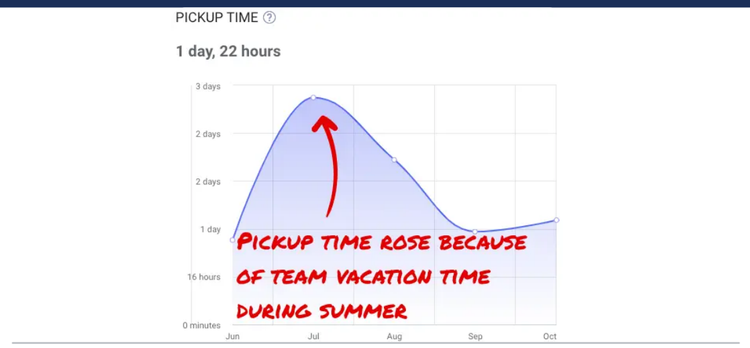
Definition
The amount of time from work (coding) began until PR is issued.

Why it matters
This is where the magic happens. Shorter code times indicate appropriately sized chunks of deliverable work.

>> See your coding time in LinearB Free

Delivery Efficiency

Efficient delivery pipelines lead to predictable value delivery, happy developers, happy product owners, and happy customers. Measuring the stages of your delivery pipeline allows for bottleneck detection. This provides a high leverage point to increase your delivery performance because it impacts all teams and contributors



#3 PR Pick-up Time

Definition
The amount of time it takes from the pull request submitted until review begins

Why it matters
Efficient teams have a low pickup time. The less time PRs spend waiting for review, the faster they are released. This metric is important for all dev teams, but is even more critical for remote dev teams.

[>> Track your Pickup Time in LinearB Free]

#4 PR Review Time

Definition
The amount of time from when the first PR comment is posted until it is merged.

Why it matters
Long reviews delay delivery and pose quality risks.

[>>Discover which PRs have long living reviews]

LINEARB

Delivery Efficiency

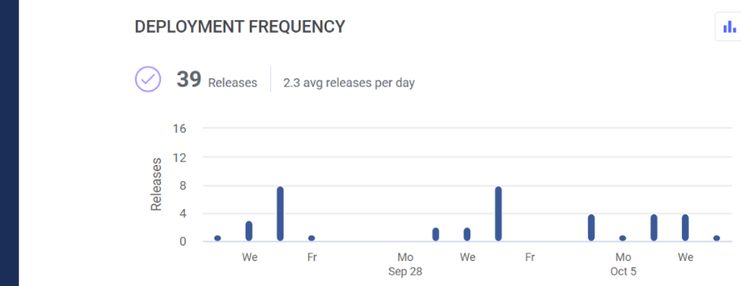
Efficient delivery pipelines lead to predictable value delivery, happy developers, happy product owners, and happy customers. Measuring the stages of your delivery pipeline allows for bottleneck detection. This provides a high leverage point to increase your delivery performance because it impacts all teams and contributors

#5 Deploy Time

Definition
The amount of time from Pull Request Merged to Production Release

Why it matters
If time to release is high, it could mean you need to invest more in continuous deployment (CD) or that you have an opportunity to move to a micro-service architecture.

>>See your Deloy Time in LinearB Free



#6 Deployment Frequency

Definition
The number of times your team deploys a release per day.

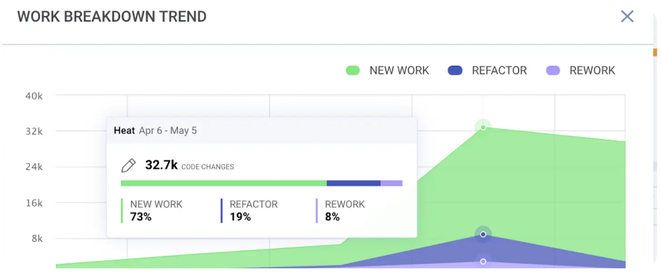
Why it matters
This is a strong indicator of how much value your team is capable of getting into the hands of customers. Even if you have an efficient pipeline, if your deployment frequency is low, you may not be delivering enough value.

>>Learn why deploy frequency is part of good retros

LINEARB

Delivery Quality

Most teams have experienced the situation where low quality leads to missed delivery dates, iteration interruptions, long hours, unhappy customers, and a frustrated engineering organization. We have found that there are a few metrics that really help to measure delivery predictability.

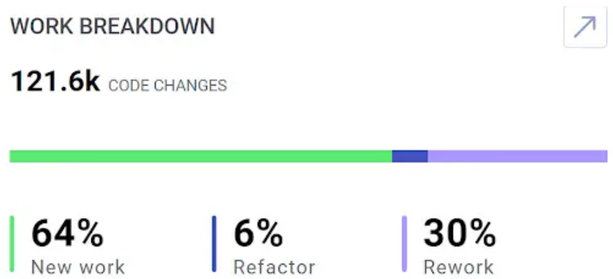


#7 Rework Ratio

Definition
Percentage of recently delivered code your team is already rewriting.

Why it matters
A high rework percentage is a predictor of future quality issues. It could also happen when a requirement was missed, indicated a communication issue with product management or the customer.

[>>See your Rework in LinearB Free]



#8 Refactor Ratio

Definition
Percentage of previously delivered (>21 days) code your team is rewriting.

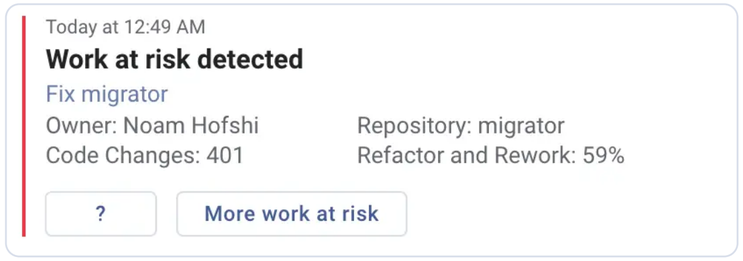
Why it matters
The implication of a high refactor ratio is context dependent. You could be fixing a section of code that has created bugs, or you simply could be updating to accommodate a new feature.

[>>Refactor ratio is part of work breakdown in LinearB]

LINEARB

Delivery Quality

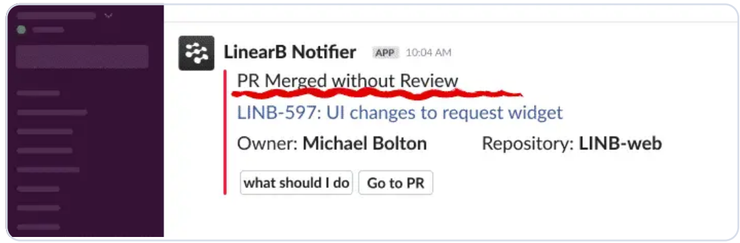
Most teams have experienced the situation where low quality leads to missed delivery dates, iteration interruptions, long hours, unhappy customers, and a frustrated engineering organization. We have found that there are a few metrics that really help to measure delivery predictability.



#9 Risky Branches

Definition
Number of branches with large changes and high rework or refactored work.

Why it matters
The general rule on most dev teams is that the larger the change, the higher the risk (i.e. branches with 300 lines of code are riskier than small branches). Branches with a high percentage of rework and refactored work are also riskier. High risk work is a leading indicator of quality.



#10 PRs Merged without review

Definition
The ratio of pull requests merged without review.

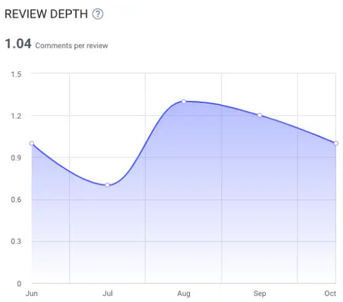
Why it matters
In addition to helping understand potential quality issues, analyzing unreviewed pull requests can sometimes be an indication of a training or culture issue on a given team.

>>Get alerts on PRs merged w/o review in LinearB

LINEARB

Delivery Quality

Most teams have experienced the situation where low quality leads to missed delivery dates, iteration interruptions, long hours, unhappy customers, and a frustrated engineering organization. We have found that there are a few metrics that really help to measure delivery predictability.



#11 Review Depth

Definition
The average number of comments per pull request review.

Why it matters
This metric is an indication regarding the quality of the review and how thorough reviews are done. Reviews are an important factor for improving code quality and finding quality issues in code before it is merged and deployed.

>>Track Review Depth as part of your quality metrics

LINEARB

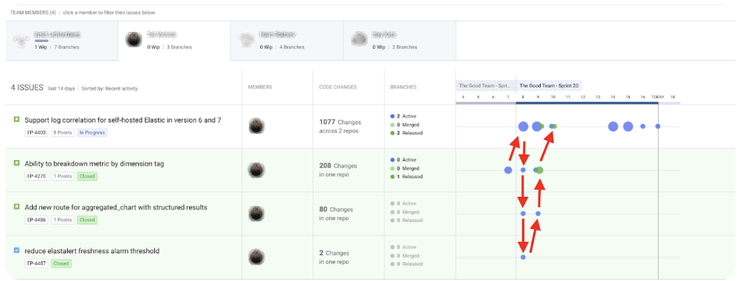
Priority Focus

Dev team spend a significant amount of time planning their work. Epics, stories, and issues that need to be completed by a certain time for the team to be successful. Priority focus metrics help the team ensure they are on track to meet objectives.

#12 Investment profile

Definition
Breakdown of how the dev team spends time across issue types (bugs, stories, tasks)

Why it matters
The most valuable asset that your organization possesses is your people's time. Your investment profile is a data-driven representation of the types of work in which your team is spending effort.



#13 Context switching

Definition
How often your team members switch from working on one issue to another because of being blocked or managerial randomization.

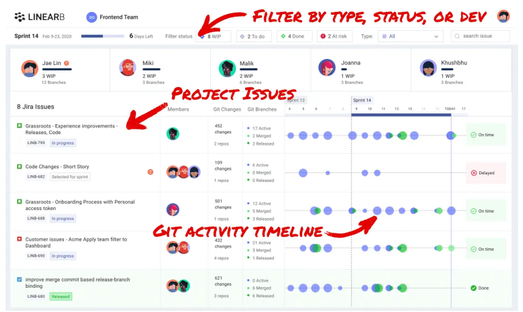
Why it matters
Large amounts of context switching is indicative of an inefficient team. As a manager, it shows you who on your team is blocked or if you need to help the team focus.

[>>Which teams are unfocused or blocked?]

LINEARB

Priority Focus

Dev team spend a significant amount of time planning their work. Epics, stories, and issues that need to be completed by a certain time for the team to be successful. Priority focus metrics help the team ensure they are on track to meet objectives.



#13 Focus Work

Definition
Work spent on most important stories for the sprint.

Why it matters
Often called success criteria, in every sprint there are stories that must ship on time for the team and business to be successful.

>>See which stories are being worked on

41 unmatched branches

#14 Unmatched Branches

Definition
Branches that do not align to a story. Often called “Shadow Work”

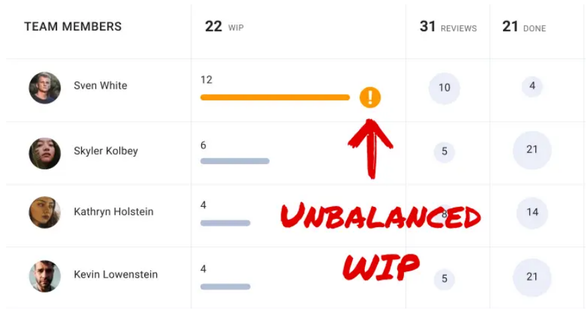
Why it matters
Shadow work can derailed the successful delivery of a sprint. It can be an indication that the lead needs to better communicate priorities or that a dev has been distracted by a request for an “urgent” issue.

>>Find your shadow work in LinearB Free

LINEARB

Team Health

Your team is crushing it. But how is the culture? While there's no single objective measure to represent culture, there are a few that shed light into how the team is working together and helping them avoid burnout.

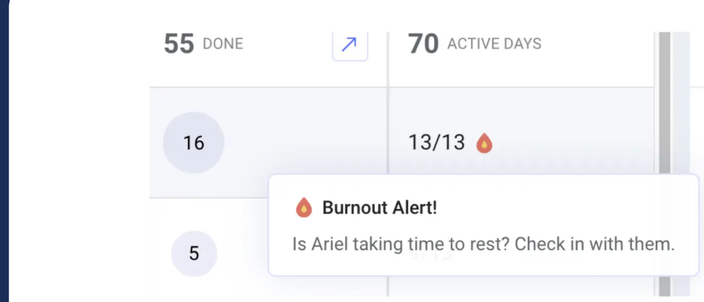


#16 WIP Balance

Definition
Measure of the distribution of work across your teams and devs.

Why it matters
Make sure you don't have one dev burning out and others that are underloaded

[>>Balance your WIP in LinearB Free]



#17 Days Worked

Definition
How many days during the iteration is your team active

Why it matters
Make sure your team isn't burning out

[>>Which team member is burned out?]

LINEARB

Dev Leader Resources

Listen & Subscribe

The Dev Interrupted podcast brings dev leaders insights on the go. Host Dan Lines talks team building, culture, leadership, and dev life with engineering and product leaders from top start-ups and scale-ups.

SUBSCRIBE ON SPOTIFY

Connect with Dev Leaders

Dev Interrupted is the largest community for dev leaders on Discord. Come together and discuss scaling teams, building culture, using metrics, and getting the most from LinearB.

JOIN US ON DISCORD

Read & Learn

The Dev Interrupted Blog is written by dev leaders, for dev leaders. How to scale your team, which metrics to use, how to grow your career... even how to explain software development to your CEO

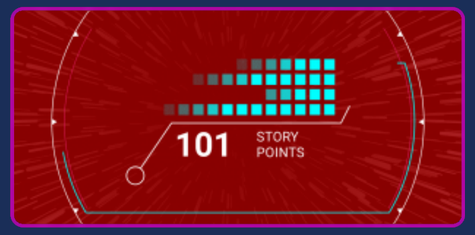
READ THE BLOG

LINEARB

Software Development Metric Potholes to Avoid

Avoid agile velocity

Using velocity as a performance metric is dangerous.



>>Read the blog

Never stack rank devs

It’s the quickest way to kill your team culture.

>>Read the blog

Don’t forget your team

Get buy-in from your team on the metrics that matter.

>>Read the blog

LINEARB

Get Started

See metrics that matter for your dev team in LinearB. Free for Dev Teams.

Click your Git provider to setup your free account

Sign up with Github

Sign up with Gitlab

Sign up with Bitbucket