2023

Software Engineering Benchmarks Report

For the first time, engineering teams can benchmark their performance against data-backed industry standards.

LinearB

TABLE OF CONTENTS

Introduction

Yishai Beeri | LinearB CTO

In 2018, the authors of *Accelerate* surveyed more than 23,000 employees at 2,000 different software companies – ranging from startups to non-profits to Fortune 500 companies – with the purpose of unveiling the methodologies that set elite organizations apart.

In their research, **DORA** asserts that “speed and stability” facilitate each other and that any engineering organization can measure and improve these outcomes. Additionally, *Accelerate* points out that the “highest performers are twice as likely to meet or exceed their organizational performance goals.”

The Continuous Delivery (CD) practices they advise, such as transitioning to shorter development cycles, automating testing, and keeping PR size small, all improve speed and quality while ingraining a culture of continuous improvement in teams.

While most of the industry is in the process of adopting CD, few have established standardized processes that measure their progress.

Alongside qualitative measures, tracking quantitative metrics, like those found in this report, is an essential step to improving operational efficiency and aligning engineering resources more closely to company goals.

It’s difficult to set realistic targets if you don’t have a baseline understanding of what “good” means for your team. Software engineering benchmarks provide a standardized and objective way to evaluate performance. By measuring high-impact metrics against industry peers, you can identify your team’s strengths, weaknesses, and workflow bottlenecks.

Gaining a deeper understanding of where you stack up in the industry also plays an essential role in:

- Establishing strategies to improve performance
- Advocating for more headcount or financial resources
- Reporting engineering efficiency and health to the business

These are the reasons we put together the 2023 Software Engineering Benchmarks Report. For the first time since DORA published its first State of DevOps Report in 2014, engineering teams can benchmark their performance against data-backed industry standards.

The following data was compiled from a study of 2,000+ teams, 100k+ contributors, and 3.7 million PRs. In this report, you'll find all metrics segmented by the following criteria: organization size, geography, and industry.

DATA SOURCED FROM

2,000+	3.7M	64
TEAMS	PRS	COUNTRIES

It's important to note that all data has been anonymized, aggregated, and normalized using a P75 (75th percentile) calculation. P75 is less sensitive to extreme values or outliers in the data, providing the most robust and reliable measure possible.

We've organized the data into the following levels of performance for each metric:

Elite	**Top 10%** of the LinearB community
Good	**Top 30%** of the LinearB community
Fair	**Top 60%** of the LinearB community
Needs Focus	**Bottom 40%** of the LinearB community

The future of software engineering is bright indeed, and we hope that the insights presented in this report can serve as a north star for dev teams aiming to ship higher quality code faster.

Yishai Beeri | LinearB CTO

Software Engineering Benchmarks

Category	Metric	Elite	Good	Fair	Needs Improvement
Efficiency	**Merge Frequency** (per dev/week)	> 2	2 - 1.5	1.5 - 1	< 1
	Coding Time (hours)	< 0.5	0.5 - 2.5	2.5 - 24	> 24
	PR Pickup Time (hours)	< 1	1 - 3	3 - 14	> 14
	PR Review Time (hours)	< 0.5	0.5 - 3	3 - 18	> 18
	Deploy Time (hours)	< 3	3 - 69	69 - 197	> 197
DORA	**Cycle Time** (hours)	< 19	19 - 66	66 - 218	> 218
	Deployment Frequency (per service)	> 1/day	> 2/week	1 - 2/week	< 1/week
	Change Failure Rate (%)	< 1%	1% - 8%	8% - 39%	> 39%
	MTTR (hours)	< 7	7 - 9	9 -10	> 10
Quality and Predictability	**PR Size** (code changes)	< 98	98 - 148	148 - 218	> 218
	Rework Rate (%)	< 2	2% - 5%	5% - 7%	> 7%
	Refactor Rate (%)	< 9%	9% - 15%	15% - 21%	> 21%
	Planning Accuracy (per sprint)	> 85%	85% - 60%	60% - 40%	< 40%
	Capacity Accuracy (per sprint)	Ideal Range **85% - 115%**	Under Commit **above 130%**	Potential Under Commit **116% - 130%**	Potential Over Commit **70% - 84%**

2022 Orgs | **3,694,690** Pull Requests | **103,807** Active Contributors | **Time Frame 08/01/22 - 08/01/23** | **At Least 400** Branches In Org

DORA Benchmarks

Cycle Time

Cycle time (aka Lead Time for Changes) measures the time it takes for a single engineering task to go through the different phases of the delivery process from 'code' to 'production'.

Deployment Frequency

Deployment frequency measures how often code is released. Elite deployment frequency represents a stable and healthy continuous delivery pipeline.

MTTR (Mean Time to Recovery)

The average time it takes to restore from a failure of the system or one of its components.

CFR (Change Failure Rate)

The percentage of deployments causing a failure in production.

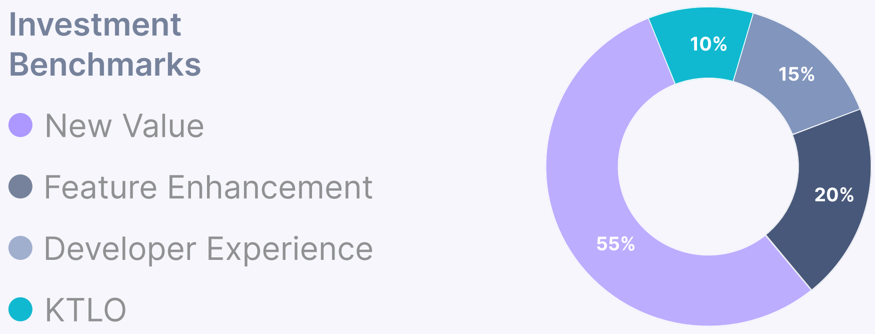
Category	Metric	Elite	Good	Fair	Needs Improvement
DORA	Cycle Time (hours)	< 19	19 - 66	66 - 218	> 218
	Deployment Frequency (per service)	> 1/day	> 2/week	1 - 2/week	< 1/week
	Change Failure Rate (%)	< 1%	1% - 8%	8% - 39%	> 39%
	MTTR (hours)	< 7	7 - 9	9 - 10	> 10

2022 Orgs | **3,694,690** Pull Requests | **103,807** Active Contributors | **Time Frame 08/01/22 - 08/01/23** | **At Least 400** Branches In Org

Engineering Investment Benchmarks

The Engineering Investment Benchmarks provide a high-level view into where and how engineering teams are investing their resources. Unlike our metrics benchmarks, you'll see the investment benchmarks do not include proficiency levels. This is intentional due to the unique needs and requirements of each organization.

We recommend using these categories and investment percentages as a starting point when aligning R&D resource investment with the board and executive team.



2022 Orgs | **3,694,690** Pull Requests | **103,807** Active Contributors
Time Frame 08/01/22 - 08/01/23 | **At Least 400** Branches In Org

Engineering teams can use the investment benchmarks to help answer questions like:

- What type of work are we spending most of our resources on?
- Are we investing enough in new features vs. keeping the lights on?
- Are we balancing our investment in new value with our investment in the tools and processes that allow new value to be built more effectively?

New Value

Work on new features that increases revenue and fuels growth by new customer acquisition or expansion.

Feature Enhancements

Incremental enhancements to existing features and work to deliver a product that ensures customer satisfaction.

Developer Experience

Work performed to improve the productivity of development teams and their overall work experience.

KTLO (Keeping the Lights On)

The minimum tasks required in order to maintain stable operations, keep high service levels, and meet compliance & regulatory requirements.

Capacity Accuracy Benchmarks

Capacity Accuracy

Measures how many issues/story points your team completed in an iteration (planned and unplanned) compared to the planned amount.

Capacity Accuracy is a unique benchmark that is based on project data. It helps teams answer the question: “Are we taking on an amount of work that we can reasonably accomplish?”

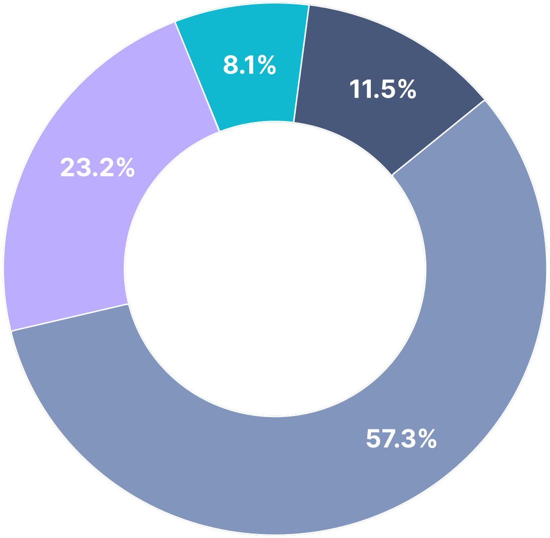
For most engineering leaders, knowing whether or not you’ll deliver a new feature set on time is a matter of experience, intuition, manual work, and lots of time spent in meetings. Unfortunately, without quantitative data to back up anecdotal evidence, engineering leaders are unable to predictably forecast project delivery timelines.

To begin accurately forecasting project delivery and determine if timelines can be moved up, we recommend tracking quality and predictability metrics such as capacity accuracy.

Actual Distribution for Capacity Accuracy

- ● Ideal Range
- ● Under Commit
- ● Potential Under Commit
- ● Potential Over-Commit

● Ideal Range	85% - 115%
● Under Commit	Above 130%
● Potential Under Commit	116% - 130%
● Potential Over Commit	70% - 84%



2022 Orgs | **3,694,690** Pull Requests | **103,807** Active Contributors
Time Frame 08/01/22 - 08/01/23 | **At Least 400** Branches In Org

Software Engineering Benchmarks by Org Size

Enterprise

Enterprise | 1000+ Employees

Category	Metric	Elite	Good	Fair	Needs Improvement
Efficiency	Merge Frequency (per dev/week)	> 1.5	1.5 - .85	.85 - .50	< .50
	Coding Time (hours)	< .25	.25 - 1	1 - 21	> 21
	PR Pickup Time (hours)	< .5	.5 - 2	2 - 10	> 10
	PR Review Time (hours)	< 1	1 - 3	3 - 17	> 17
	Deploy Time (hours)	< 4	4 - 102	102 - 221	> 221
DORA	Cycle Time (hours)	< 29	29 - 113	113 - 291	> 291
	Deployment Frequency (per service)	> 1/day	> 2/week	1 - 2/week	< 1/week
	Change Failure Rate (%)	< 1.25%	1.25% - 5.75%	5.75% - 32%	> 32%
	MTTR (hours)	< 8	8 - 9	9 - 10	> 10
Quality and Predictability	PR Size (code changes)	< 89	89 - 135	135 - 182	> 182
	Rework Rate (%)	< 10%	10% - 15%	15% - 21%	> 21%
	Refactor Rate (%)	< 9%	9% - 15%	15% - 21%	> 21%

2022 Orgs | **3,694,690** Pull Requests | **103,807** Active Contributors | **Time Frame 08/01/22 - 08/01/23** | **At Least 400** Branches In Org

Software Engineering Benchmarks by Org Size

Scale-up

Scale-Up | 200 - 1000 Employees

Category	Metric	Elite	Good	Fair	Needs Improvement
Efficiency	Merge Frequency (per dev/week)	> 1.75	1.75 - 1	1 - .50	< .50
	Coding Time (hours)	< .50	.50 - 3.25	3.25 - 35	> 35
	PR Pickup Time (hours)	< 1.5	1.5 - 4	4 - 15	> 15
	PR Review Time (hours)	< 1.5	1.5 - 4	4 - 19	> 19
	Deploy Time (hours)	< 3	3 - 50	50 - 170	> 170
DORA	Cycle Time (hours)	< 27	27 - 86	86 - 215	> 215
	Deployment Frequency (per service/week)	> 1/day	> 2/week	1 - 2/week	< 1/week
	Change Failure Rate (%)	< .70%	.70 - 4.5%	4.5 -21.5%	> 21.5%
	MTTR (hours)	< 8	8 - 9	9 - 10	> 10
Quality and Predictability	PR Size (code changes)	< 98	98 - 139	139 - 209	> 209
	Rework Rate (%)	< 3%	3 - 5%	5 - 8%	> 8%
	Refactor Rate (%)	< 13%	13 - 17%	17 - 21%	> 21%

2022 Orgs | **3,694,690** Pull Requests | **103,807** Active Contributors | **Time Frame 08/01/22 - 08/01/23** | **At Least 400** Branches In Org

Software Engineering Benchmarks by Org Size

Startup

Startup | 0-200 Employees

Category	Metric	Elite	Good	Fair	Needs Improvement
Efficiency	Merge Frequency (per dev/week)	> 2	2 - 1.35	1.35 - .85	< .85
	Coding Time (hours)	< .35	.35 - 3	3 - 26	> 26
	PR Pickup Time (hours)	< 1	1 - 3	3 - 15	> 15
	PR Review Time (hours)	< 0.5	0.5 - 3	3 - 20	> 20
	Deploy Time (hours)	< 3	3 - 70	70 - 194	> 194
DORA	Cycle Time (hours)	< 21	21 - 71	71 - 233	> 233
	Deployment Frequency (per service)	> 1/day	> 2/week	1 - 2/week	< 1/week
	Change Failure Rate (%)	< .75%	.75 - 5%	5 - 20%	> 20%
	MTTR (hours)	< 8	8 - 9	9 - 10	> 10
Quality and Predictability	PR Size (code changes)	< 98	98 - 150	150 - 218	> 218
	Rework Rate (%)	< 2%	2- 4%	4 - 7%	> 7%
	Refactor Rate (%)	< 10%	10 - 15%	15 - 20%	> 20%

2022 Orgs | **3,694,690** Pull Requests | **103,807** Active Contributors | **Time Frame 08/01/22 - 08/01/23** | **At Least 400** Branches In Org

GENERAL METRICS INSIGHTS

Benchmark Definitions

Efficiency

Merge Frequency

The average number of pull or merge requests merged merged by one developer in one week. Elite merge frequency represents few obstacles and a good developer experience.

Coding Time

The time it takes from the first commit until a pull request is issued. Short coding time correlates to low WIP, small PR size and clear requirements.

Pickup Time

The time a pull request waits for someone to start reviewing it. Low pickup time typically represents strong teamwork and a healthy review process.

Review Time

The time it takes to complete a code review and get a pull request merged. Low review time represents strong teamwork and a healthy review process.

Deploy Time

The time from when a branch is merged to when the code is released. Low deploy time correlates to high deployment frequency.

DORA

Cycle Time

(AKA Lead Time for Changes) measures the time it takes for a single engineering task to go through the different phases of the delivery process from 'code' to 'production'.

Deployment Frequency

Measures how often code is released. Elite deployment frequency represents a stable and healthy continuous delivery pipeline.

MTTR (Mean Time to Recovery)

The average time it takes to restore from a failure of the system or one of its components.

CFR (Change Failure Rate)

The percentage of deployments causing a failure in production.

Quality and Predictability

PR Size

The number of code lines modified in a pull request. Smaller pull requests are easier to review, safer to merge, and correlate to a lower cycle time.

Rework Rate

The amount of changes made to code that is less than 21 days old. High rework rates signal code churn and is a leading indicator of quality issues.

Refactor Rate

Refactored work represents changes to legacy code. LinearB considers code "legacy" if it has been in your code-base for over 21 days.

Planning Accuracy

The ratio of planned work vs. what is actually delivered during a sprint or iteration. High planning accuracy signals a high level of predictability and stable execution.

Capacity Accuracy

Measures all completed (planned and unplanned) work as a ratio of planned work.

PR Review Process Insights

Key Takeaway

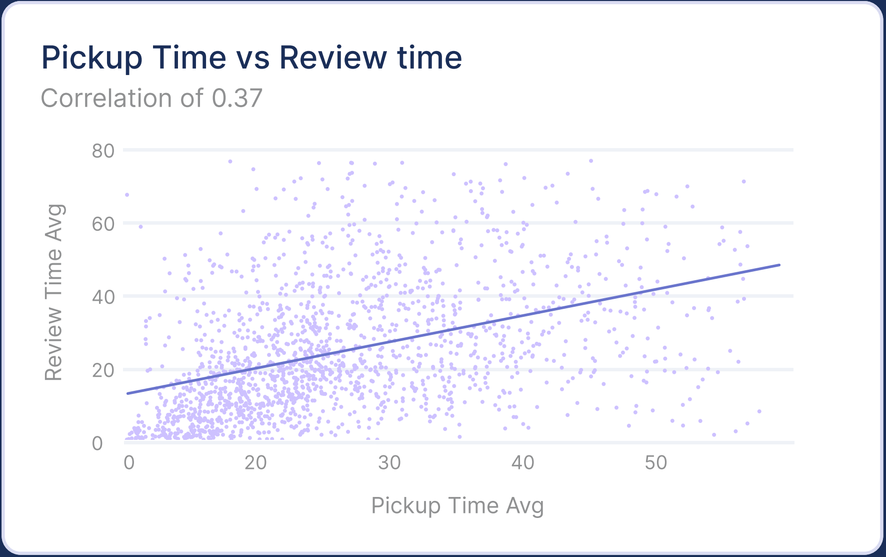
There is a positive correlation between longer PR pickup/handoff times and longer review times.

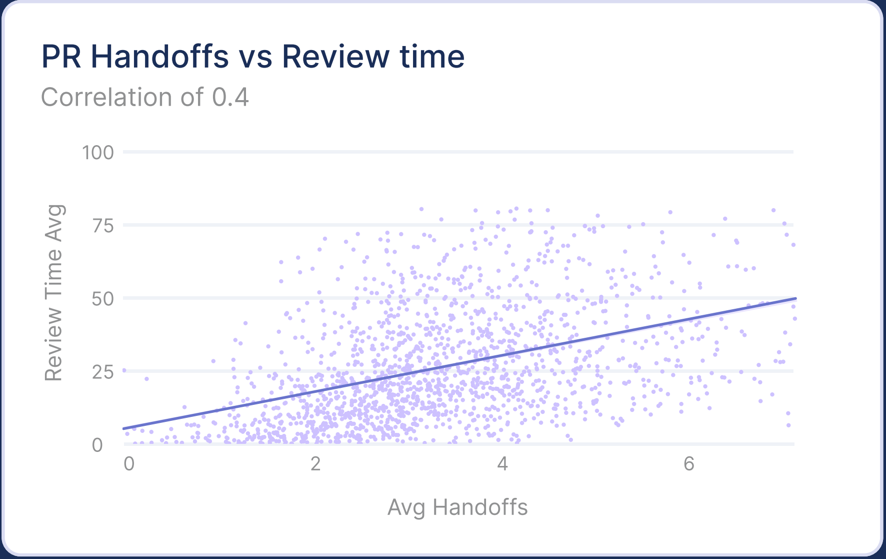
It's important to note that correlation does not indicate causation. However, this data's insights align closely with the qualitative and anecdotal research we've gathered from LinearB users over the past year.

The longer a PR sits waiting for a review, the more likely it is that the developer will move onto another line of work. And this type of context switching almost always poses a risk to review time – the longer a developer spends away from a PR they authored, the less fresh the context will be in their mind when they return to it. When the review actually starts, it will take the developer longer to address comments and move the PR along.

Key Takeaway

Pull requests with a higher number of handoffs (back-and-forth touches between contributors) will have a higher review time.





Time in hours. Every data point represents one engineering organization.

A higher number of review cycles for a pull request can indicate a number of underlying inefficiencies that ultimately increase a team's cycle time. Pull request size, code complexity, and developer experience levels are common factors that cause higher review times. Each of these factors and more are quickly amplified with the introduction of pull request idle time - the amount of time a pull request sits idle between review phases - due to the effect time has on context switching and flow state.

Reducing PR Pickup Time

Tool Tip

Add estimated review time labels to PRs, so developers can gain context into reviews and prioritize their time accordingly.

Reducing PR pickup time is a crucial element of reducing cycle time and streamlining the software development life cycle. When it comes to reducing this metric, here are some best practices we recommend:

Assign the right reviewer

Leverage workflow automation to route PRs to developers with the most relevant recent activity and knowledge on the code being modified.

Encourage devs to author small PRs

When PRs contain fewer lines of code, they present a less daunting undertaking for the reviewer and are far more likely to get picked up quickly.

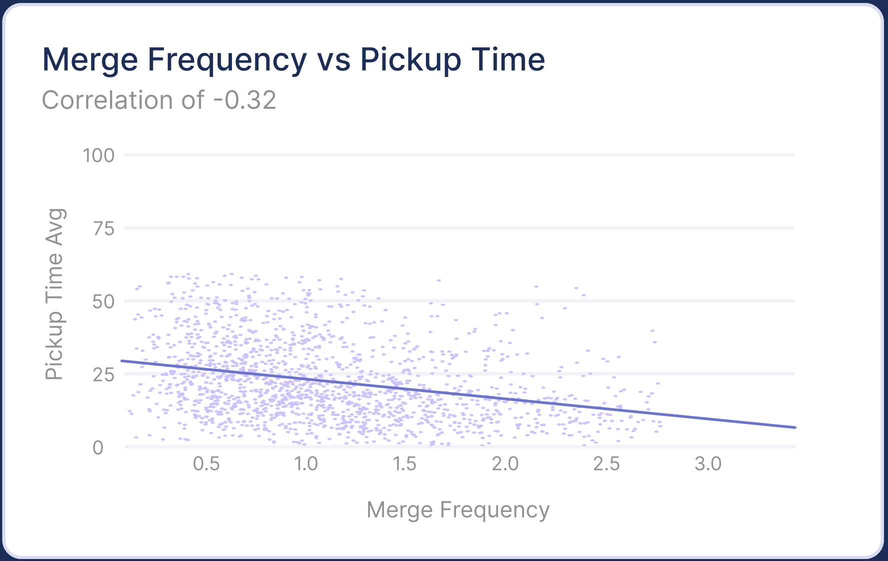
Get real-time alerts on PR activity

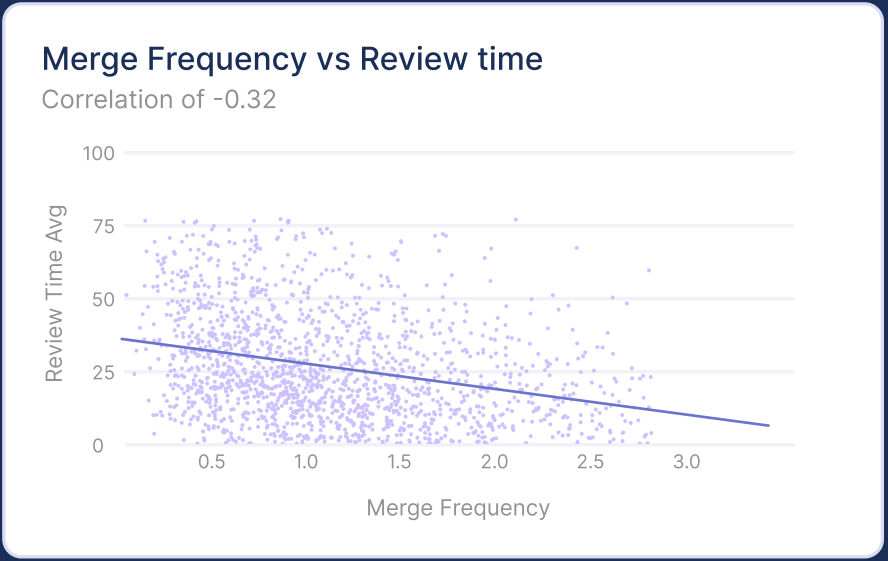
Setting up real-time notifications will provide you with immediate context about your PRs, including review assignments, approvals, comments and change requests.

> "Visibility into our engineering metrics has given our business critical telemetry and attribution for our engineering teams. In the past 6 months, we were able to reduce our cycle time from an average of 6 days to 2 days."

Matt C.
Engineering Chief of Staff

Super.com





Every data point represents one engineering organization's average.

Merge Frequency Insights

Key Takeaway

Long PR review cycles are a key obstacle to achieving high merge frequency.

The data displays a negative correlation between both merge frequency and pickup time, as well as merge frequency and review time, meaning that the longer a PR sits before getting picked up, the lower an organization's merge frequency will be. Conversely, when PR review & pickup time lag, your team will merge fewer PRs over the same amount of time.

Our research revealed that PR pickup time has the strongest correlation with quantitative productivity measures relative to the other cycle time metrics. Ultimately, this suggests that PR pickup time is the component metric of cycle time that perhaps mostly affects inefficiencies and idle time.

Merge frequency is a leading indicator of developer experience and software delivery pipeline health. Teams with higher merge frequencies have fewer review cycle bottlenecks that ultimately frustrate developers and slow code delivery. Optimizing for merge frequency is one of the most important pieces when it comes to creating an elite developer experience and improving retention.

Improving Code Reviews

Here are some best practices we recommend to dev teams looking to streamline the PR Review process:

Automate reviewer assignment

Leverage a workflow automation tool like **gitStream**, which will route PRs to the developer with the most commit activity and knowledge on the files in question.

Request changes on deprecated APIs

Set up automated alerts whenever a PR includes use of a deprecated API. This allows engineering managers to encourage best practices while remaining hands-off.

Set expectations with PR Labels

Using automated labels to categorize PRs (e.g., bug fix, high risk, documentation) can help reviewers prioritize the most critical work and plan their days accordingly.

Tool Tip

We recommend engineering leaders implement a set of merge standards on their teams to establish policy-as-code and ensure consistent adoption across the organization.

Check out 13 of our favorite merge standards that enforce quality and boost efficiency in **The Continuous Merge Guide to Merge Standards.**

> "High performance teams need a psychologically safe environment. They don't gel if someone is getting a whack on the knuckles every time they forget a Pull Request review. Workflow automation helps manage our Pull Request process with simple alerts. It's just a nice way of doing it."

Jon Sowler
VP of Engineering

Unbabel

SEGMENT INSIGHTS

Cycle Time by Org Size

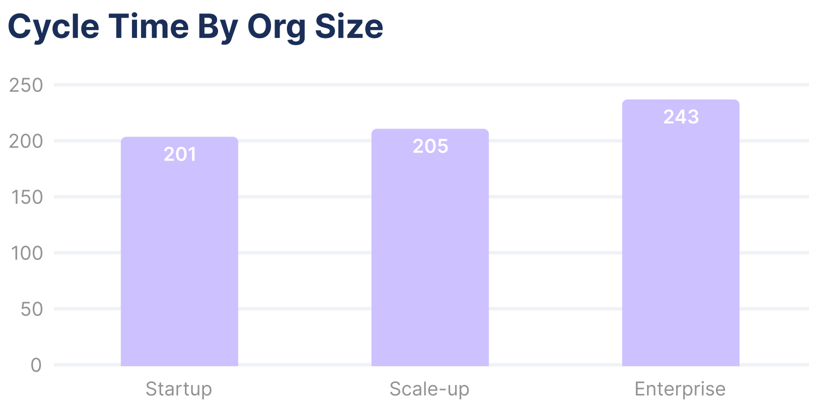
Key Takeaway

Enterprises have longer cycle times than startups and scale-ups.

One of the most interesting findings in this year's report was that enterprises have a notably slower cycle time than startups and scale-ups. Conversely, startups and scale-ups are neck and neck when it comes to this metric, with only a 2% difference in speed.

This could, of course, be for a variety of reasons, but is most likely due to the fact that smaller and mid-size companies are more flexible, and consequently, more agile. They can make decisions and adapt to changes more swiftly, reducing the time it takes to approve and merge PRs.

Additionally, smaller companies tend to have leaner processes due to less bureaucracy and fewer layers of management. A more minimal organizational structure allows for quicker decision-making and less red tape when it comes to PR reviews and approvals.



Startup: 0-200 employees
Scale-up: 200-1000 employees
Enterprise: 1000+ employees

Cycle Time by Org Size

Conversely, large enterprises often have more complex organizational hierarchies and formalized processes. Their PRs may need to go through multiple layers of management and reviews, leading to delays.

What's more, large enterprises typically have more extensive and complex codebases as well as legacy systems that may require additional scrutiny and testing. They also cater to a significantly larger customer base and as such, face higher stakes when merging code.

Despite these challenges, however, large organizations are increasingly adopting DevOps practices to streamline their development processes. By implementing automation, fostering cross-functional teams, and encouraging a culture of efficiency, even Fortune 500 companies can work towards reducing cycle time and remaining agile in today's fast-paced software development landscape.

Assignar

"Software projects can be unpredictable due to a multitude of reasons - from unforeseen technical challenges to scope changes. Engineering metrics, such as Planning Accuracy, and workflow automation tools have helped us increase predictability in release schedules and timelines."

Marko T.

CTO

Efficiency Metrics by Org Size

Key Takeaway

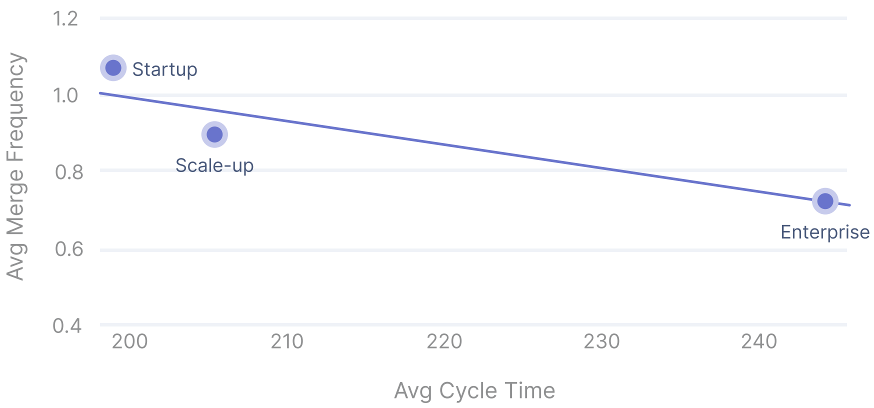
Teams with shorter cycle times are able to achieve higher merge frequency.

Although the PR review process comprises just one leg of the full cycle time, it's perhaps not surprising that longer cycle time correlates with lower merge frequency.

This highlights the toll that higher PR WIP can take on lead time, due to longer cycles and the resulting need for context switching.

Startups and scale-ups tend to fall on the left side of the graph – with lower cycle times and higher merge frequencies – whereas enterprises fall to the right, with higher cycle times and lower merge frequencies.

Merge Frequency vs Cycle Time



Startup: 0-200 employees
Scale-up: 200-1000 employees
Enterprise: 1000+ employees

● Merge Frequency
— Trendline for Merge Frequency

Efficiency Metrics by Org Size

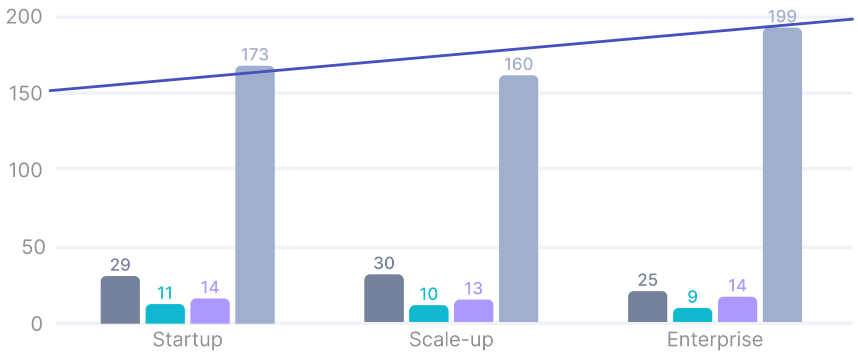
As seen in the Cycle Time Average Breakdown graph, enterprises are comparable to startups and scale-ups across the coding, pickup, and review segments of cycle time. Only in deploy time do we see slower cycles for enterprises

Notably, deploy time dominates cycle time. Yet the deploy process typically doesn't directly overlap with merge activities. So why do enterprises have a lower merge frequency?

One possible answer is that undeployed code (and the attention required to coordinate deploys) is a significant burden on developers that detracts from their ability to merge code frequently. Another answer is that other factors are at play – such as the team's work hours and overlap between work days. We'll touch on these in more detail later in this report.

As merge frequency appears to be impacted by indirect factors, and not just pickup and review times, it is important to track this metric directly to ensure you are improving it incrementally so as to unblock your teams.

Cycle Time Avg Breakdown



Startup: 0-200 employees
Scale-up: 200-1000 employees
Enterprise: 1000+ employees

- ● Coding Time
- ● Pickup Time
- ● Review Time
- ● Deploy Time

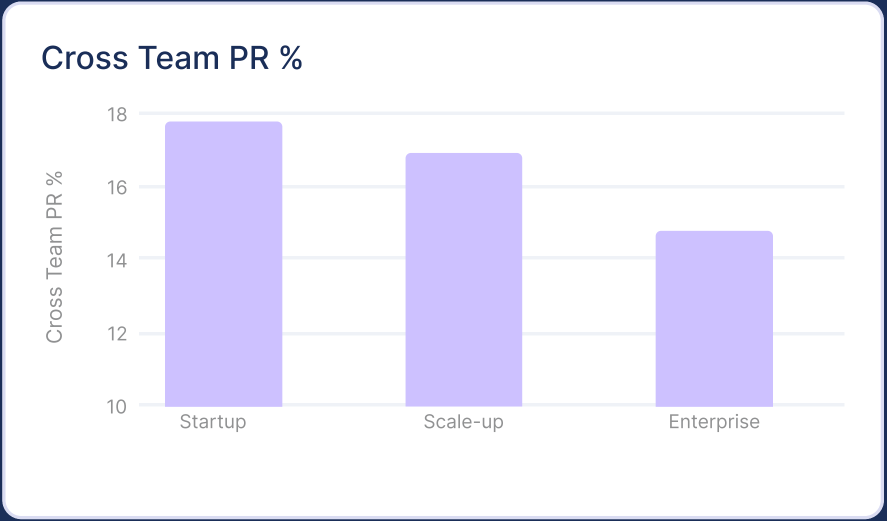
Did You Know

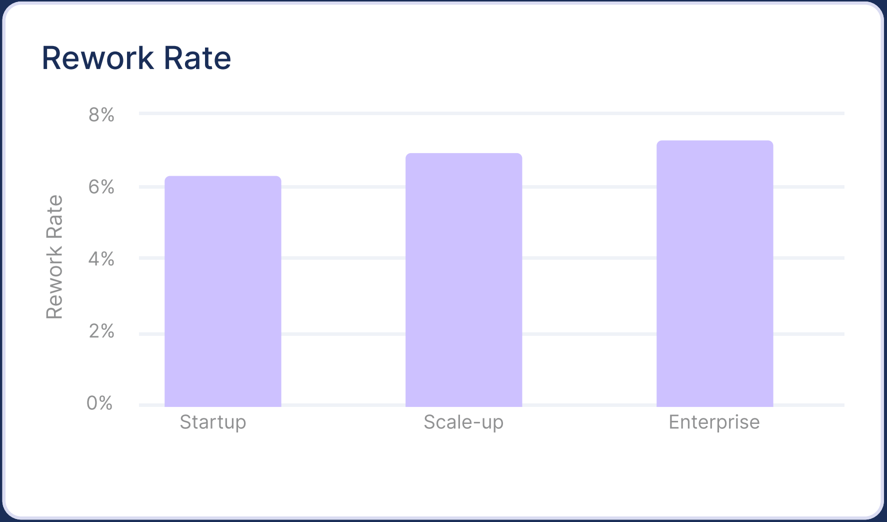
Enterprises have a deploy time that is 18% higher than their start-up and scale-up counterparts.

Quality Metrics by Org Size

Key Takeaway

Enterprises have far less cross-team collaboration on PRs than startups and scale-ups.





***Startup:** 0-200 employees* | ***Scale-up:** 200-1000 employees* | ***Enterprise:** 1000+ employees*

Pull request collaboration across teams is substantially lower in enterprises, compared to scale-ups or startups, possibly due to more work silos and rigid boundaries around code ownership at the enterprise level.

On the other hand, rework rate is around 17% higher in enterprises compared to startups. One possible explanation is that siloed knowledge – coupled with collaboration and reviews only happening within team boundaries – are delaying the detection of integration issues with other teams' work, leading to more rework required to successfully integrate.

Enhanced cross-team visibility and collaboration during the PR review process may help reduce this rework.

Enhancing Security with Workflow Automation

For teams looking to enable cross-departmental collaboration and improve code quality, here are some tips from the LinearB dev team:

Require Extra Reviewers for Complex PRs

Automatically assign two reviewers those few PRs that are especially complex or affect risky components in the codebase.

✓ Auto-approve documentation changes

Proper documentation plays an integral role in facilitating cross-departmental collaboration. We recommend leveraging a workflow automation tool like gitStream to verify and auto-approve all documentation changes.

⚠ Let DevSecOps take the reins on high-risk work

Flag risky code changes with workflow automation. Automatically pull in DevSecOps as required reviewers for PRs with security implications.

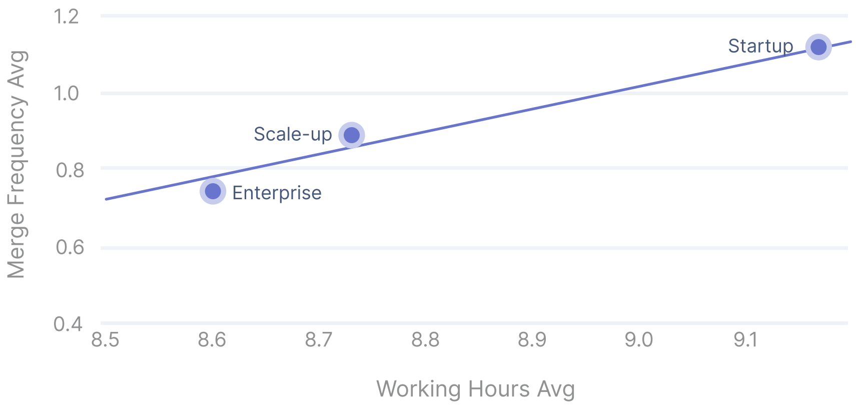
flipdish

"Workflow automation has had a cascading impact on improving business outcomes while enabling my team to focus on solving the correct problems. Tracking delivery metrics helped us pinpoint areas of flow that could be enhanced, dramatically boosting team morale and engagement."

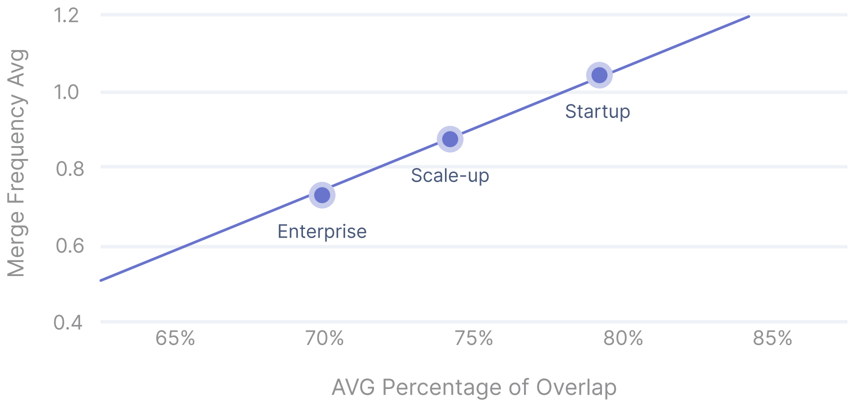
Craig W.
Head of Engineering

Working Hours and Merge Frequency by Org Size

Merge Frequency vs Avg Working Day



Work Hours Overlap vs Merge Frequency in Teams



Benchmarks by number of employees

Startup: 0-200 employees
Scale-up: 200-1000 employees
Enterprise: 1000+ employees

● Merge Frequency
— Trendline for Merge Frequency

Key Takeaway

Startups work longer hours than scale-ups and enterprises, and have a higher Merge Frequency.

There is a positive correlation between merge frequency and working hours, with startups working the longest days and exhibiting a higher merge frequency.

This could be for a variety of reasons, namely:

High Stakes

90% of startups fail, and most startups find themselves in a precarious position, especially in early stages. The success or failure of a company may very well hinge on achieving critical goals quickly. This pressure can lead to longer workdays to ensure those goals are met.

Resource Constraints

Startups typically have limited resources – including time, money, and headcount. To make up for this deficit, employees may work longer hours to meet tight deadlines.

Increased Competitive Pressure

Early stage companies often operate in highly competitive markets where merge frequency can be a crucial factor. Long hours may be necessary to outpace competitors and gain a foothold in the market.

Cultural Factors

Startup culture often encourages a strong work ethic and dedication to the company’s success. As such, team members may feel a sense of camaraderie and commitment to working hard together.

Additionally, our data displayed a positive correlation between merge frequency and percentage of overlapping working hours. We define overlapping hours as those within the active work day for at least half of the team’s contributors.

That overlapping work hours (and conversely, timezone gaps) play a significant role in facilitating (or hindering) teamwork should be no surprise.

Key Takeaway

Merge frequency is higher when team members are working similar hours.

Here are some of the specific ways in which higher overlap improves merge frequency:

Real-time collaboration

Engineers working during the same hours can collaborate more effectively in real time. They can schedule meetings, initiate pair programming sessions and conduct code reviews without significant delays. This real-time collaboration can lead to quicker issue resolution and faster progress.

Reduced communication barriers

When developers are working at the same time, there are fewer obstacles to effective communication. Engineers can have spontaneous discussions, ask questions and seek clarification without worrying about inconveniencing their teammates.

Faster feedback loops

Engineers with overlapping hours can provide quicker feedback to their colleagues. This is especially important during code reviews and testing phases, during which faster feedback loops mean that code changes can be improved and merged more rapidly.

REGION AND INDUSTRY INSIGHTS

DORA Metrics by Region

Region	Metric	Elite	Good	Fair	Needs Improvement
Europe	Cycle Time (hours)	18.61	85.44	230.10	> 230.10
	Deployment Frequency (per service)	> 1/day	> 2/week	1 - 2/week	< 1/week
	Change Failure Rate (%)	0.47%	2.65%	11.98%	> 11.98%
	MTTR (hours)	7.48	9.00	10.26	> 10.26
North America	Cycle Time (hours)	22.25	69.98	237.02	> 237.02
	Deployment Frequency (per service)	> 1/day	> 2/week	1 - 2/week	< 1/week
	Change Failure Rate (%)	0.82%	5.26%	21.96%	> 21.96%
	MTTR (hours)	7.38	9.09	10.50	> 10.50
Rest of the World	Cycle Time (hours)	24.67	89.77	227.94	> 228
	Deployment Frequency (per service)	> 1/day	> 2/week	1 - 2/week	< 1/week
	Change Failure Rate (%)	0.92%	5.27%	21.41%	> 21.41%
	MTTR (hours)	7.23	8.79	10.53	> 10.53

2022 Orgs | **3,694,690** Pull Requests | **103,807** Active Contributors | **Time Frame 08/01/22 - 08/01/23** | **At Least 400** Branches In Org

DORA Benchmarks by Industry

2022 Orgs | **3,694,690** Pull Requests | **103,807** Active Contributors
Time Frame 08/01/22 - 08/01/23 | **At Least 400** Branches In Org

Industry	Metric	Elite	Good	Fair	Needs Improvement
Finance & Banking	Cycle Time (hours)	21.58	67.98	242.60	> 242.60
	Deployment Frequency (per service)	> 1/day	> 2/week	1 - 2/week	< 1/week
	Change Failure Rate (%)	0.63%	3.17%	17.46%	> 17.46%
	MTTR (hours)	7.48	9.20	10.55	> 10.55
Other	Cycle Time (hours)	27.09	84.33	230.53	> 230.53
	Deployment Frequency (per service)	> 1/day	> 2/week	1 - 2/week	< 1/week
	Change Failure Rate (%)	0.59%	3.37%	14.46%	> 14.46%
	MTTR (hours)	7.60	9.05	10.45	> 10.45
Professional Services	Cycle Time (hours)	22.59	72.01	282.57	> 282.57
	Deployment Frequency (per service)	> 1/day	> 2/week	1 - 2/week	< 1/week
	Change Failure Rate (%)	0.65%	5.86%	28.29%	> 28.29%
	MTTR (hours)	7.22	8.67	10.77	> 10.77
Software & IT Services	Cycle Time (hours)	16.72	71.93	227.36	> 227.36
	Deployment Frequency (per service)	> 1/day	> 2/week	1 - 2/week	< 1/week
	Change Failure Rate (%)	0.95%	5.76%	22.08%	> 22.08%
	MTTR (hours)	7.51	9.10	10.30	> 10.30

Region and Industry Insights

Our research didn't observe a notable difference in metrics across regions or industries, and it's important to note that not all of the region and industry data is statistically significant. However, there are a few interesting points we'd like to highlight.

Firstly, our data indicated that Software and IT Services have a shorter deployment time, which can likely be attributed to workflow automation and/or early adoption of CI/CD methodologies.

Point of Interest

Software companies have a shorter deployment time than other industries.

Moreover, even with shorter working days and fewer employees clocking in on weekends, Europe displayed outlier performance across several metrics categories, namely: coding time, deploy time, and CFR.

Point of Interest

Europe has a 28% shorter deploy time than the rest of the world.

> **"Tracking engineering metrics has helped me as the VP of R&D to have educated discussions with my direct reports and with my CEO. I can identify bottlenecks quickly, measure team efficiency and the developer experience, then improve based on data."**

Idan Lavy
VP of R&D

SOLIDUS LABS

Conclusion

Pressure is the one constant in an engineering leader's day-to-day life. Pressure to deliver more features. Pressure to deliver them faster. Pressure to take on an extra priority project or customer RFE. All with a flat or shrinking budget. These responsibilities all revolve around driving operational excellence.

But in the last few years, the role and core responsibilities of engineering changed, mirroring shifts in the business landscape. Software development and delivery became a key driver of business value. With that paradigm shift comes new added pressure to deliver business results.

This is the dual mandate of engineering leaders: continue delivering operational excellence while simultaneously driving the business forward.

Average 120 day improvement with LinearB

47% Decrease in **Cycle Time**

10.2x Return on **Investment**

We hope that this Software Engineering Benchmarks Report will give engineering leaders and their teams a better understanding of their performance today, such that they can build a strategy that helps them improve and hit their business goals in the future.

Teams can make the biggest improvements when they know what to expect. And a predictable pipeline is one that allows for incremental shifts in the right direction.

With standardization and repeatability built into your development workflows, teams of all sizes can move faster and scale efficiently.

Did You Know

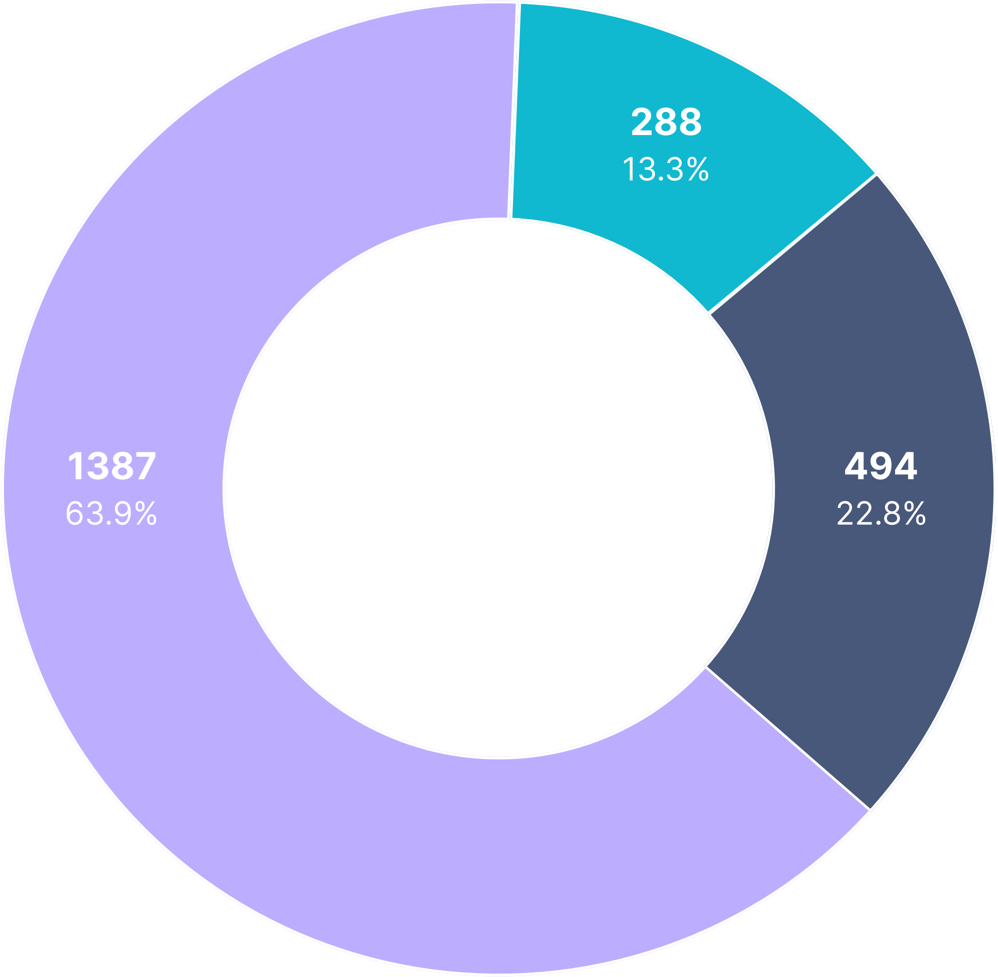
LinearB metrics and programmable workflows have already saved developers thousands of hours, with the average repo seeing a 61% decrease in Cycle Time.

You can **see where your team stacks up with a free forever account** and begin building your engineering metrics program today! If you'd like to discuss any of what was covered in this report in more detail, or you want to see some of the more advanced features, you can **schedule a demo.**

APPENDIX

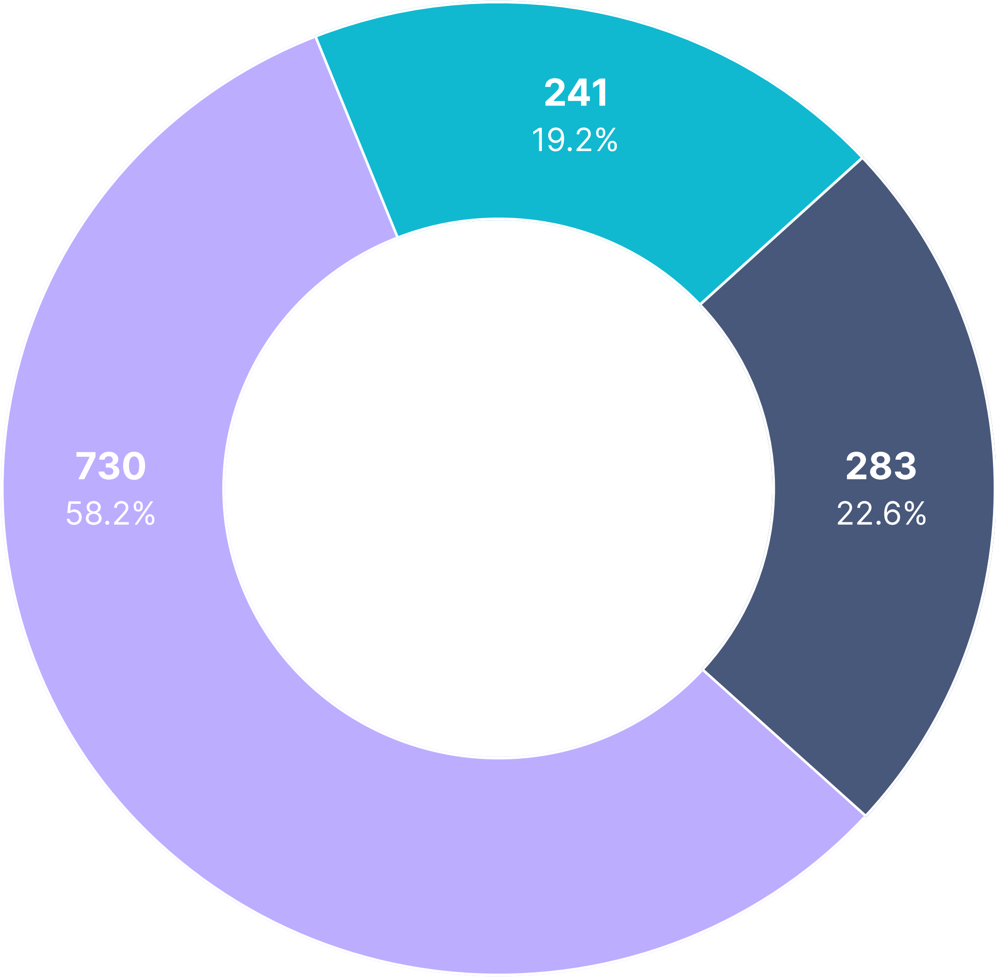
Org Size Distribution

Enterprise | Scale-up | Startup



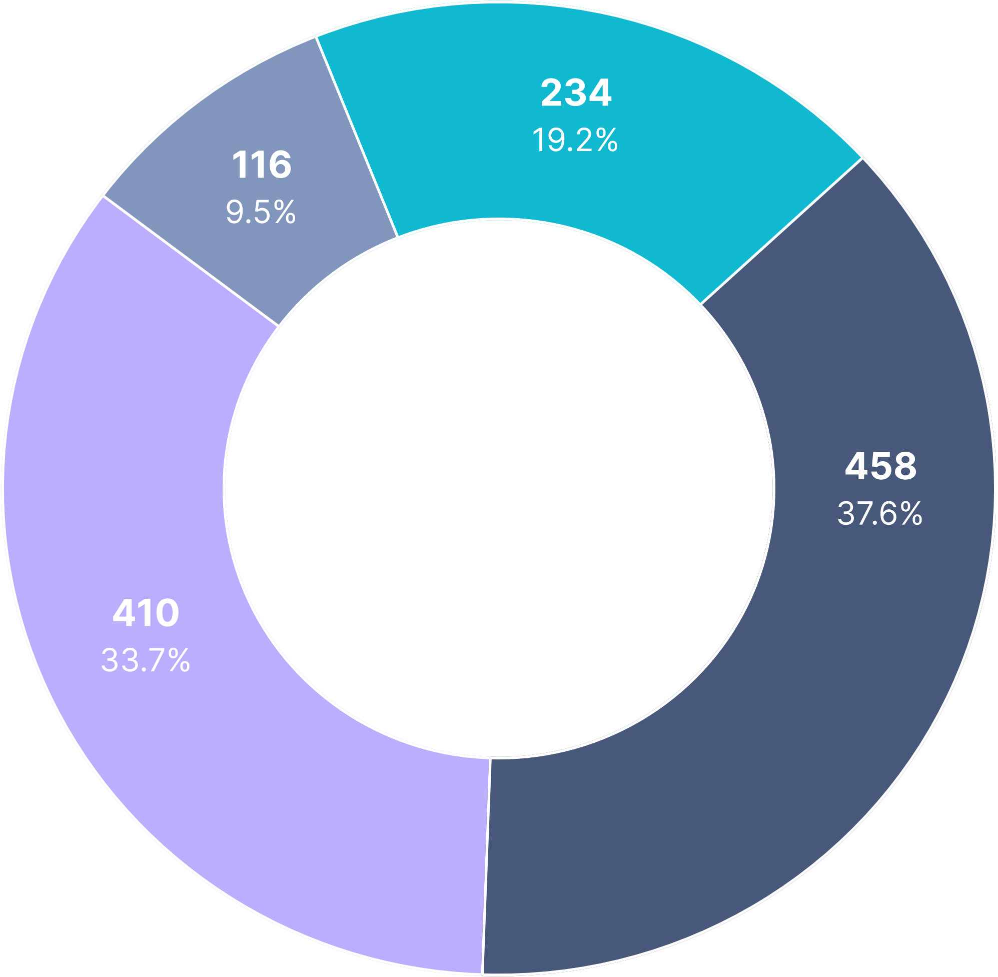
Org Regions Distribution

Rest of the world | Europe | North America



Org Industries Distribution

Professional Services | Software & IT Services | Other | Finance & Banking



Investment Metrics Glossary

New Value

Work on new features that increases revenue and fuels growth by new customer acquisition or expansion.

This might include activities such as:

- Adding a new feature
- Implementing roadmap work, etc.
- Supporting a new platform or partner application

Feature Enhancements

Incremental enhancements to existing features and work to deliver a product that ensures customer satisfaction.

This might include activities such as:

- Customer requested improvements
- Improved performance / utilization
- Improved product reliability or security, etc.
- Iterations to improve adoption/retention/quality

Developer Experience

Work performed to improve the productivity of development teams and their overall work experience.

This might include activities such as:

- Code restructuring
- Testing automation
- Better developer tooling
- Work to reduce size of the KTLO bucket in the future

KTLO (Keeping the Lights On)

The minimum tasks required in order to maintain stable operations, keep high service levels, and meet compliance & regulatory requirements.

This might include activities such as:

- Maintaining current security posture
- Service and ticket monitoring & troubleshooting
- Maintaining current levels of service uptime, etc.