ATLASSIAN

State of the Developer Report

2022

Table of contents

Summary

The future of software development will be defined by greater autonomy

Atlassian's first State of the Developer report looks into the trends shaping software development and their impact on development teams in markets around the world.

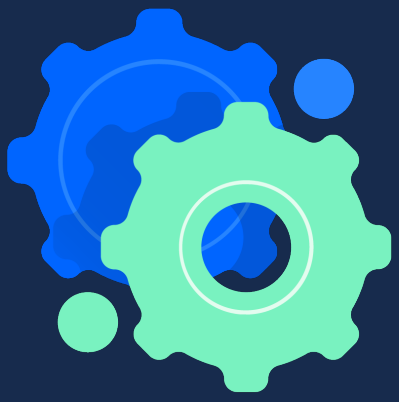
With organizations managing increasingly distributed teams, giving software developers and development teams greater autonomy has become crucial. 'You build it, you run it' (YBIYRI) models are on the leading edge of this trend.

Insights gained from over 2,000 developers across Australia, Germany, India and the US clearly demonstrate that greater autonomy makes developers happier at work, despite more frequent context switching and greater job complexity. Developers who enjoy more autonomy tend to spend more time coding and work on a greater number of products.

Read on to learn more about:

- The rise of "You build it, you run it" (YBIYRI) as a software development methodology
- Coding from scratch versus increased use of tools and platforms
- Why flexibility is key when adopting more tools
- The need for greater autonomy within development teams

Atlassian has been making software that development teams rely on to get their jobs done for 20 years now. As the industry continues to evolve, we remain committed to helping shape the future for our customers and our people. This report has been designed to spark and inform the conversations behind ongoing industry change.



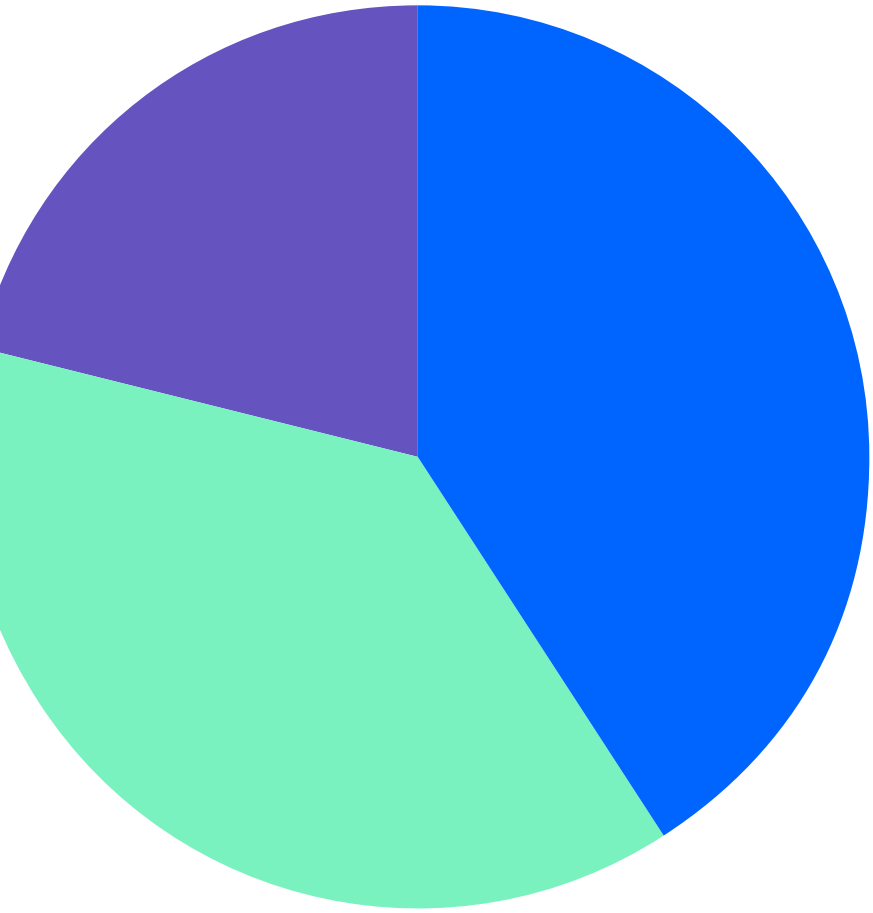
01

The rise of “You build it, you run it”

Over the last 13 years, the “You build it, you run it” (YBIYRI) philosophy has become increasingly prevalent. Its rise has coincided with that of collaborative DevOps, representing a shift in the way that companies think about their development lifecycle. Simply put, this expands the role of software developers to include supporting the products they build from start to finish. This brings developers closer to customers and leads to better products through a feedback loop.

This study shows that 59% of teams have adopted a YBIYRI approach. This can be further divided into two groups: those that have been working this way for some time (established YBIYRI - 21%) and those that are new to the practice (new to YBIYRI - 38%).

You build it, you run it



41%

NON-YBIYRI: 41%

38%

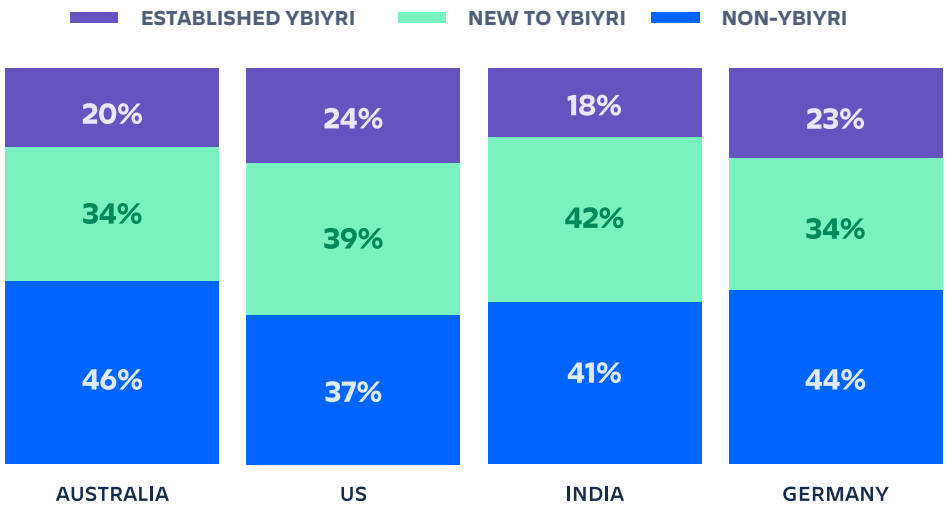
NEW TO YBIYRI: 38%

21%

ESTABLISHED YBIYRI: 21%

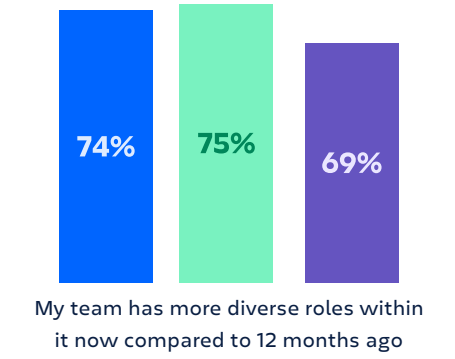
By country level, YBIYRI has a longer history in the US and Germany where 23-24% of teams are working this way. The practice is clearly growing, with 34-42% of teams falling into the new to YBIYRI classification.

YBIYRI by Country

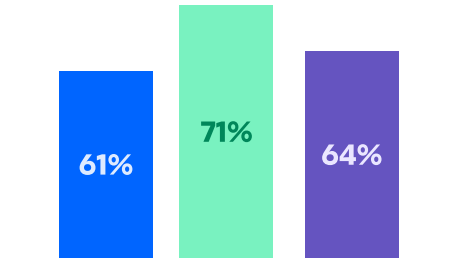


Team roles

Since established YBIYRI teams have been working this way for more than 12 months, they are less likely than new YBIYRI teams to feel that the mix or diversity of their roles has changed over the last 12 months, nor do they think that more diverse roles are needed now. Teams that are new to YBIYRI feel most strongly that the diversity of team roles should change, reflecting the increased scope of work that YBIYRI requires.



My team has more diverse roles within it now compared to 12 months ago



I think my team should have more diverse roles than we currently do

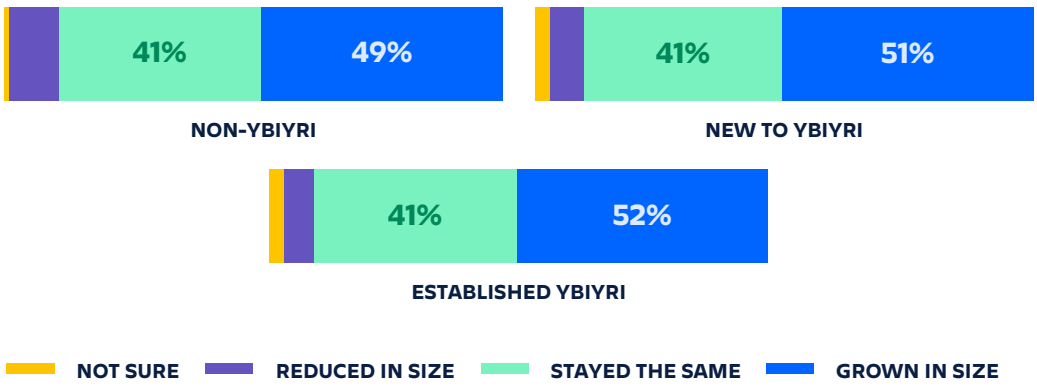
TAKEAWAYS

- YBIYRI is an increasingly common practice, with almost 60% of teams currently working this way.
- Teams working in YBIYRI require new and diverse roles, especially when they are transitioning into the practice.

Team size

Non-YBIYRI teams are more likely to have declined in size over the last 12 months, while the vast majority of development teams have grown. This indicates high demand for coding skills regardless of the approach taken to the development process.

How has your team size changed in the last 12 months?



Teams that are new to YBIYRI are especially likely to have expanded the number of roles they have in numerous areas such as cloud or server engineers, developers (front-end, back-end, and full-stack), security engineers, project managers, solution architects, and machine learning (ML) engineers. Established YBIYRI teams do not show the same pattern of role growth, likely because they have already added people to these roles. These established YBIYRI teams are more likely to have increased the number of hardware/network and DevOps engineers, indicating the need for these roles as teams mature their YBIYRI practice and require greater operational support.

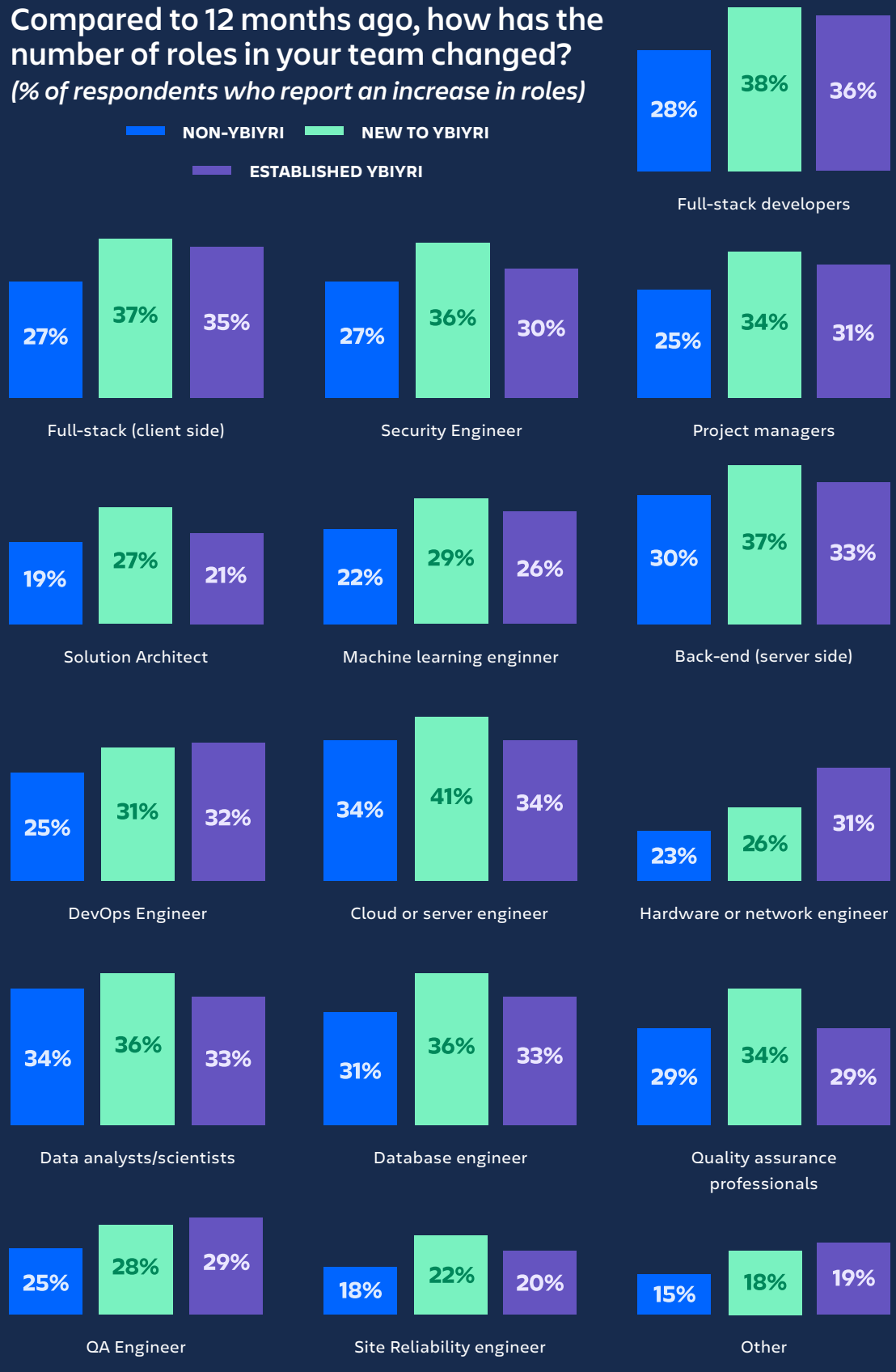
TAKEAWAYS

YBIYRI teams are more likely to continue growing than traditional development teams.

YBIYRI teams are likely to require additional developer capacity initially and more operational roles as they become established.

Compared to 12 months ago, how has the number of roles in your team changed?

(% of respondents who report an increase in roles)

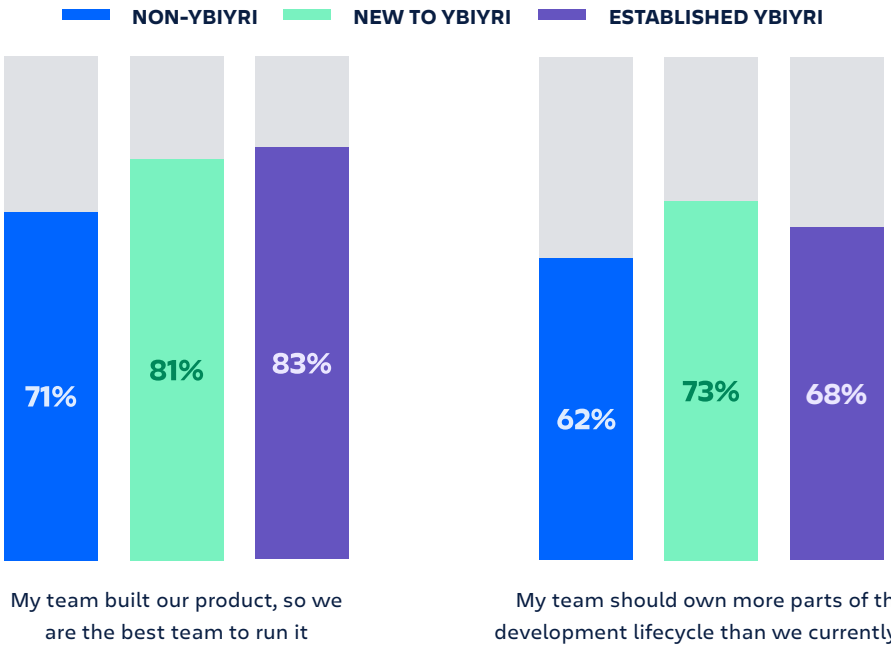


Model ownership

Teams adopting the YBIYRI model are aligned with this philosophy and way of working. YBIYRI teams are much more likely to feel that they are the best people to run a product since they built it. This point of view becomes more established over time, with those working in established YBIYRI teams most likely to agree with this perspective (83%). YBIYRI teams are also more likely to feel that their responsibilities should be increased further, especially those who are new to the practice and still trying to find the optimal responsibility mix. Almost three-quarters of new to YBIYRI teams (73%) agree that their team should be responsible for more parts of the software product lifecycle than they currently are.

YBIYRI Team Responsibilities

(% of respondents who agree with the following statements)



TAKEAWAYS

The longer a team works in a YBIYRI environment, the more likely they are to feel that they are best placed to run their products.

Teams new to YBIYRI are especially likely to think that they should own more of the lifecycle than they have in the past.

02

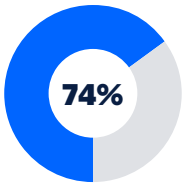
Coding from scratch versus increased use of tools and platforms

The value of coding

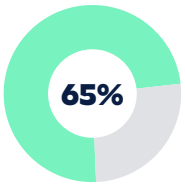
Two-thirds of developers (65%) say that writing new code is the most valuable skill in their role, while 74% feel that being able to read code is vital. Yet 58% of developers don't feel that writing code from scratch will be required as part of their roles in the future, and 51% say they mainly assemble code written by others. These findings look contradictory at first glance.

To what extent do you agree or disagree with the following statements about your code skills in your role?

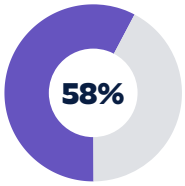
(% of respondents who agree with the following statements)



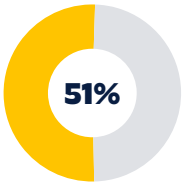
To be effective in my role, it is critical I can read other proples' code



Writing new code is the most valuable skill needed in my role



In the future, writing code from scratch will not be a skill required in my role



I mainly spend time 'assembling' code written be others

TAKEAWAYS

Developers think that coding and reading code are both valuable skills, but this doesn't mean that those skills can never be replaced by tools.

The value of tooling

This is an area of contention for developers. However, we get a more nuanced view on where developers think the future of coding is headed based on their sentiments toward tooling and value of compiling code. One-third (32%) say that coding is the most valuable skill and is unlikely to be replaced by tools, a similar number (33%) are undecided, and 22% say tools will ultimately make coding obsolete. A smaller group (13%) values coding but also thinks it will eventually be replaced by tools.

These divided opinions mean that we can't assume that all developers, or even teams of developers, feel the same way about coding or the tools and platforms that may replace it from scratch in the future.

Managers and leaders of development teams should think carefully about their toolchains, what tools they add, and when to add them.

This helps ensure that developers feel they have an adequate voice in their work, especially as they navigate complex transitional periods.

If we segment these groups further, it shows a clear difference in their views.

Code>tools

- High value of coding and code review
- Low likelihood of tools replacing code
- Unlikely to be assembling code written by others

Code<tools

- High value of coding and code review
- Low likelihood of tools replacing code
- Unlikely to be assembling code written by others

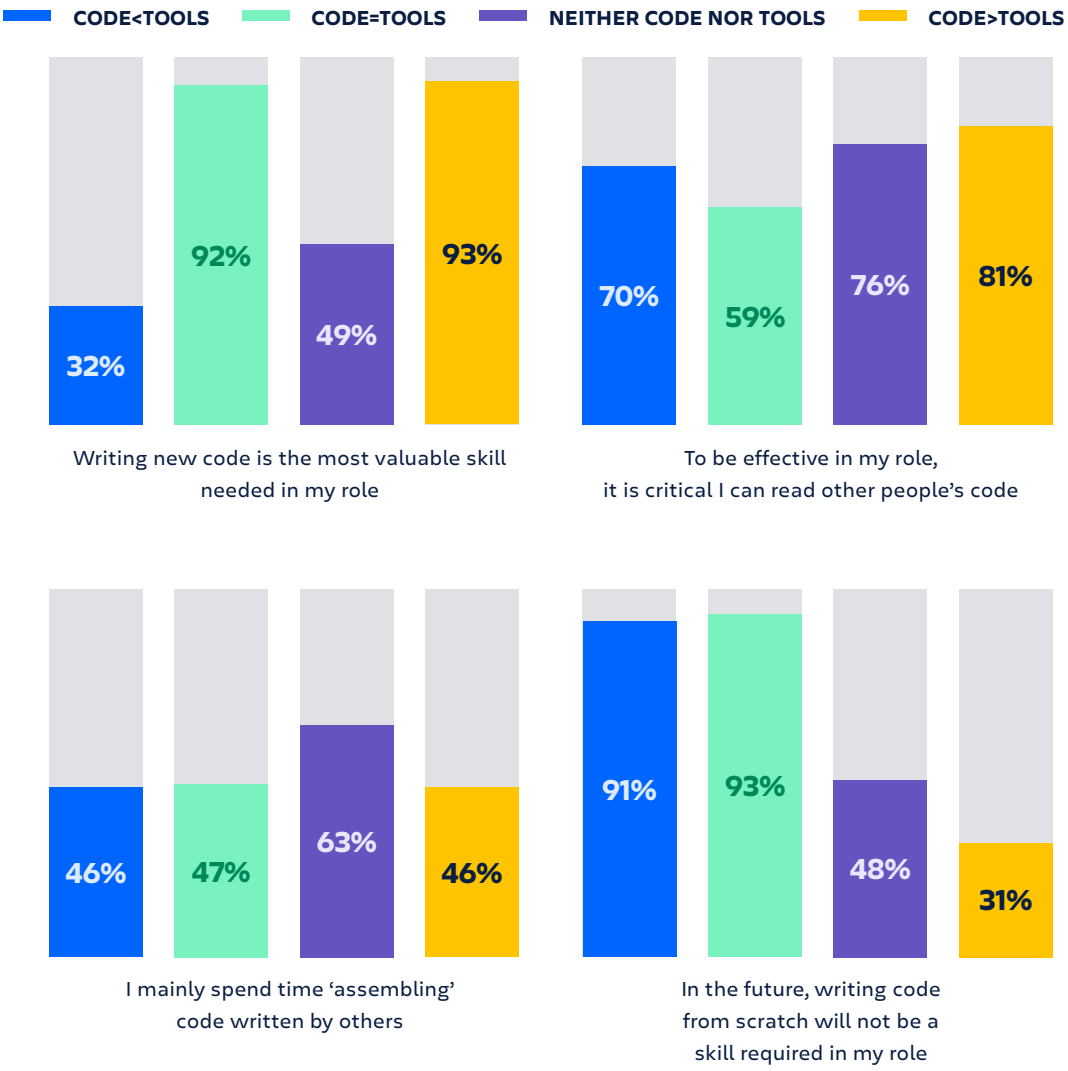
Code=tools

- High value of coding
- High likelihood of tools replacing code
- Unlikely to be assembling code written by others
- Moderate value of code review

Neither code nor tools

- Moderate value of coding
- Moderate likelihood of tools replacing code
- Most likely to be assembling code written by others
- High value of code review

To what extent do you disagree or agree with these statements about your code skills in your current role?
(% of respondents who agree with the following statements)



TAKEAWAYS

- We cannot assume homogeneity in how developers feel about the value of coding.
- Some have strong views on coding and tools, while others operate in a place of plurality.

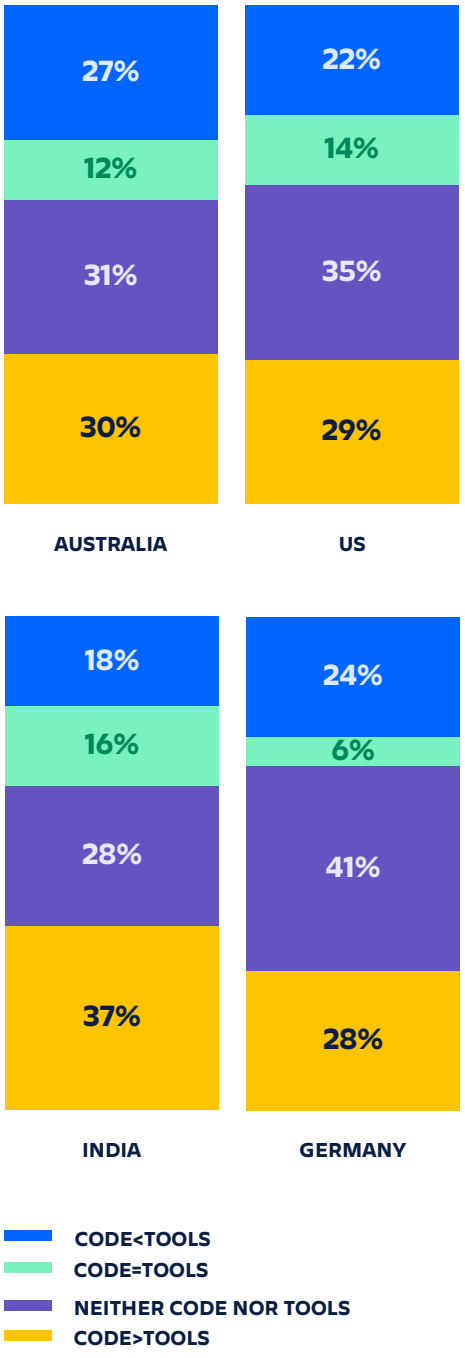
What influences tooling value for developers?

These groups are not equally distributed by country. Developers in India are more likely to be in the code>tools group (37%) and are less likely to fall into the “neither” group (28%). Those in Germany are more likely to be in the “neither” group (41%) and are much less likely to be in the code=tools group (6%). This trend is similar for developers in the US, with a 35% majority in the “neither” group and a minority 14% in the code=tools group. Developers in Australia are more likely to think that tools will eventually replace coding (27%).

Despite their prioritization of code, the code>tools group say the platforms and tools that they use are more adaptable (78%) and simplify their work (76%). They feel that tools still require a high level of technical skill, which is why they don’t think tools will replace coding outright. The code<tools group have moderate views on tools and platforms across most measures, but are less likely to think that they are spending more time on these tools than they did before (54%).

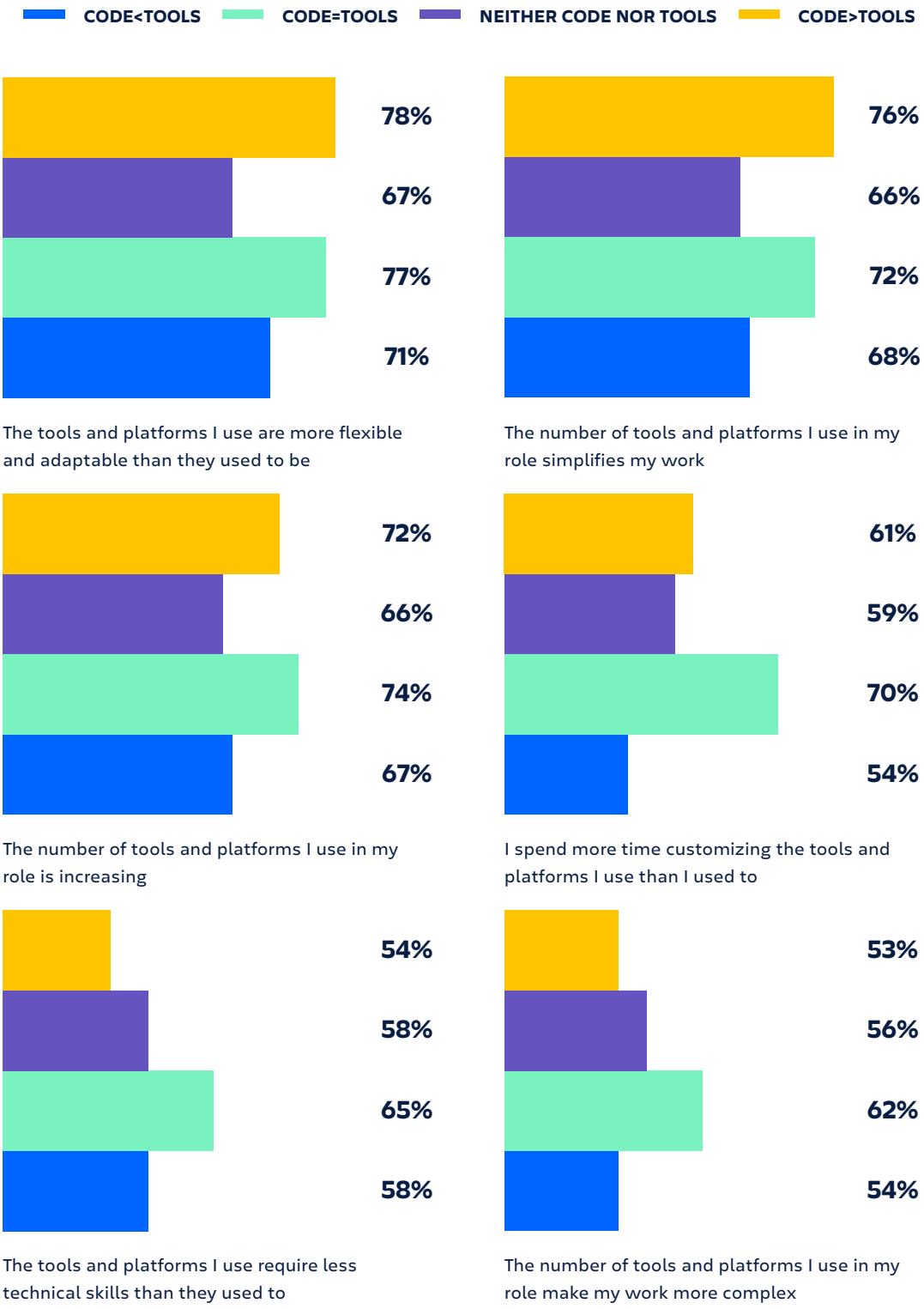
This is an indication that the trade-off is more manageable and that using these tools make them think that they will eventually replace coding from scratch. The neither code nor tools group is much less likely to think that the platforms they use are flexible or simplify their work. They also don’t feel that the number of tools has increased for them.

Code groups by country



Tools and platform views by code group

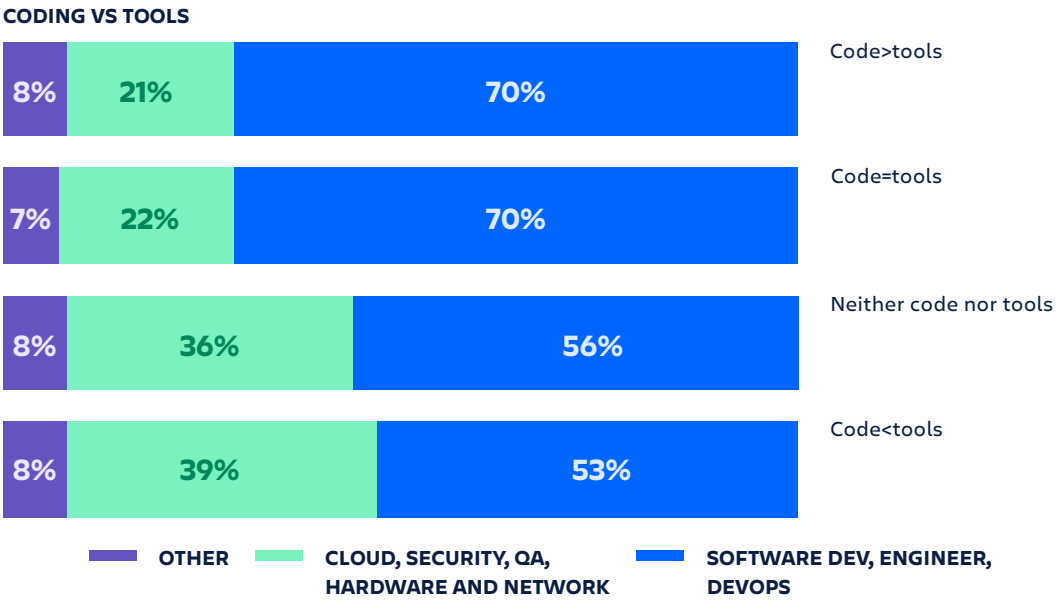
(% of respondents who agree with the following statements)



Preference depends on role and proficiency

The groups that favor tools over code or who value neither highly are less likely to be working in development roles like software engineer, software developer or DevOps. They are more likely to be in adjacent roles such as network engineer, cloud engineer, security, or quality assurance. This explains part of the pattern because these groups are less likely to be working directly with code on a daily basis.

Role by code group

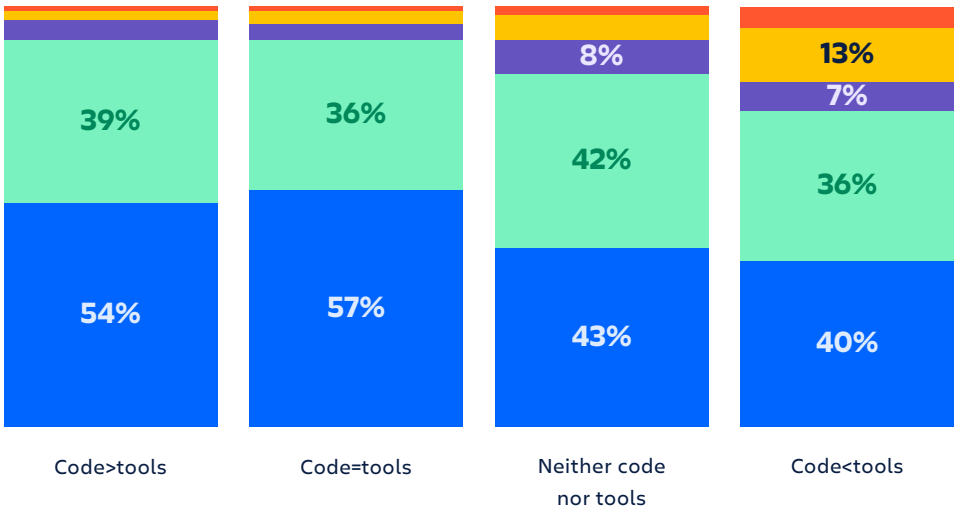


This trend is clear when we examine the activities that each of these groups frequently perform as part of their role. The neither group is more likely to be reusing code, managing code dependencies, and monitoring applications. The code<tools group spends more time planning and scoping. The code>tools group are the most frequent coders of these groups, showing strong association with all code-related activities we measured. The code=tools group spends their time maintaining repositories and deploying and documenting code. They are more likely to design solution architectures and to review code, but are less likely to write code directly.

The two groups that most commonly work with code (Code=tools and Code>tools) are also the most proficient coders. More than 90% consider themselves proficient or expert coders.

Those that think tools will eventually replace coding (Code<tools) are the least proficient coders, with 76% claiming to be proficient or expert.

Coding proficiency by code group



TAKEAWAYS

- The flexibility and adaptability of tools, as well as the technical skill required to use them, are key factors in whether developers favor tools or code in their work.
- Developers that favor code over tools are more proficient coders and are more likely to spend time in coding-heavy roles.
- Those that favor tools over code tend to work in roles adjacent to software development such as security, quality assurance or hardware/network.

03

Flexibility is key when adopting more tools

Is tool sprawl rising?

Much has been written in recent years about the horrors of tool sprawl in IT and technology companies. This is most often mentioned in the context of wasted effort, lost productivity, and lack of efficiency, but our research reveals more nuanced attitudes within development teams. A majority of developers (61%) are currently using more than 6 different tools.

By region, tool sprawl may be worst in India with 78% of developers saying they are using more than 6 tools. In the US, this number is 72%. Only half of the developers in Germany (50%) use more than 6 tools. Developers in Australia remain middle of the pack with 61% saying they use more than 6 tools.

Number of tools used

61%

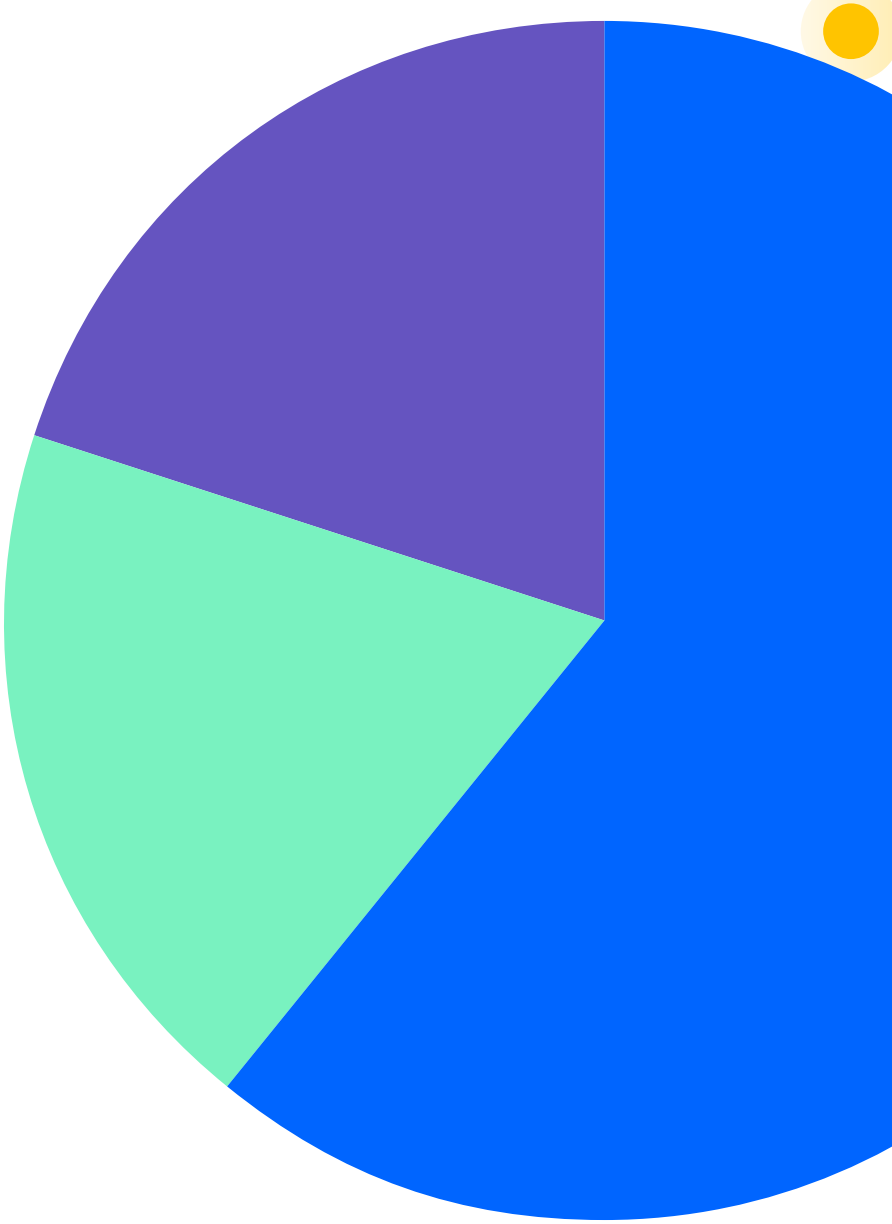
61% = 7+

19%

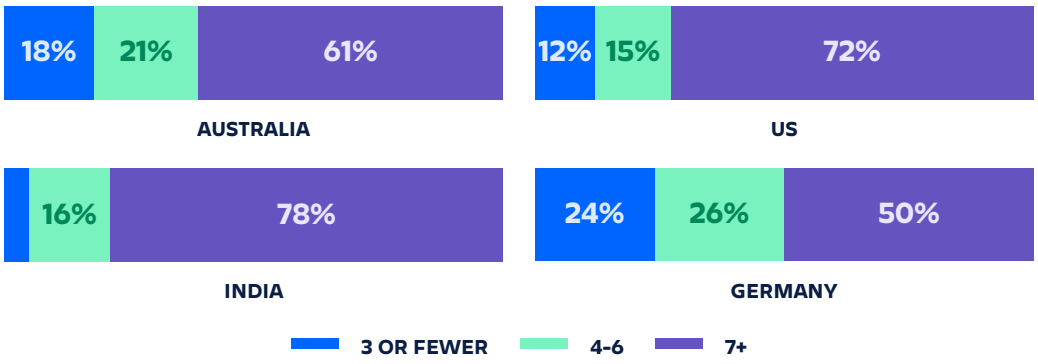
19% = 4-6

20%

20% = 3 OR FEWER

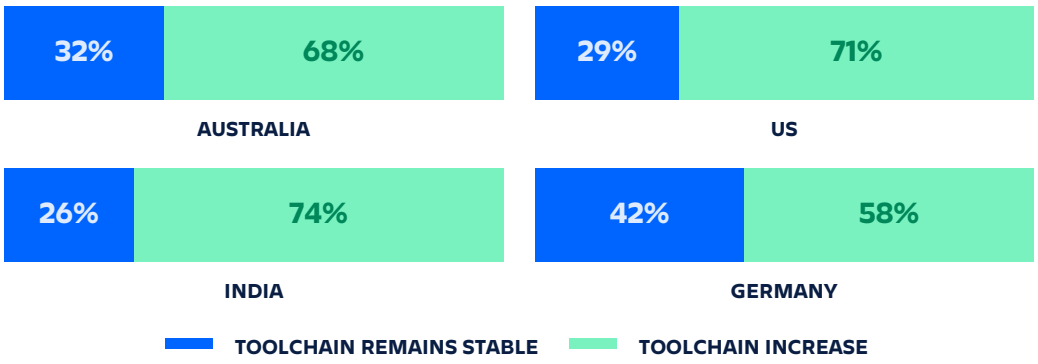


Number of tools used - by country



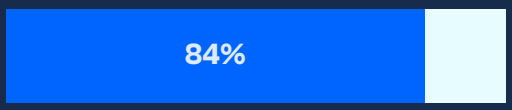
It's clear that teams are adding more tools to their stack over time, with 69% of developers telling us they have increased the number of tools they use in their role. This is at least true in Germany, where only 58% of developers say they have more tools in use.

The number of tools and platforms I use in my roles is increasing - by country

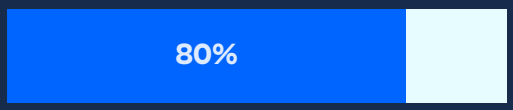


Almost two-thirds of respondents claim to use every tool we mentioned in the survey, while 84% say they use planning, tracking, and task management tools.

Which of the following types of tools have you used in the last month?

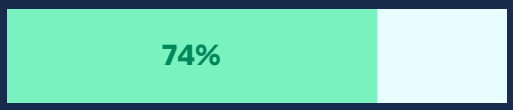


Tools for planning tracking and managing tasks and projects (i.e. Jira, Trello, Asana)

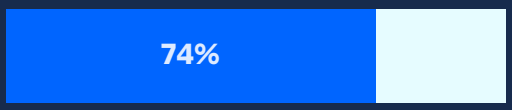


Source code managment (SCM) and version control (VC) tools (i.e. GitHub, Bitbucket)

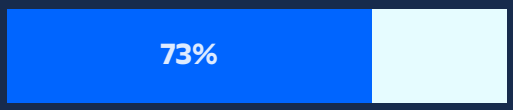
Cloud deployment tools (i.e Amazon EC2, Azure, Google cloud)



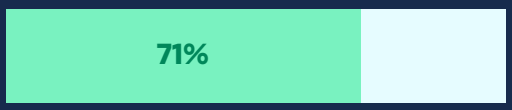
Create, manage and run functional tests through testing tools (i.e. Selenium, JUnit and Pytest)



Tools that compile or build software (i.e. SBT)

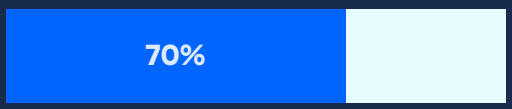


Tools to monitor track and fix crashes and errors (i.e. Sentry, Sumo Logic, Splunk)

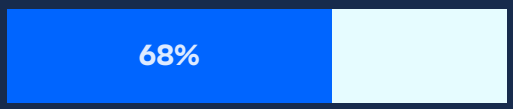


Tools that help manage configuration and application deployment (i.e. Ansible)

CI/CD tools that automate development, deployment and testing (i.e. Jenkins, CircleCI)



Tools that monitor and notify me of security threats and ensure data integrity (i.e. Tripwire)



Mapping, designing and prototyping tools (i.e. Sketch, Miro, Mural)

TAKEAWAYS

The majority of developers say that the number of tools they use is growing.

The tools that developers use are similar, with most using those we measured.

Tool sprawl remedied by flexible tooling

The issue with tool sprawl has less to do with the number of tools that developers use or their increased usage, with our research showing the focus should be on how flexible and adaptable those tools are.

This becomes clearer when we divide our sample into those with a stable toolchain, those using an increased number of tools, those with more flexible tools, and those with more inflexible tools. These groups are based on how they responded to statements about the flexibility and stability of their toolchain:

- Stable toolchain – Score low (neutral or disagree) on the statement, “The number of tools and platforms that I use in my role is increasing”
- More tools – Score high (agree or strongly agree) on the statement, “The number of tools and platforms that I use in my role is increasing”
- Flexible tools – Score high (agree or strongly agree) on the statement, “The tools and platforms that I use are more flexible and adaptable than they used to be”
- Inflexible tools – Score low (neutral or disagree) on the statement, “The tools and platforms that I use are more flexible and adaptable than they used to be”

Toolchain stability and flexbilty

46%

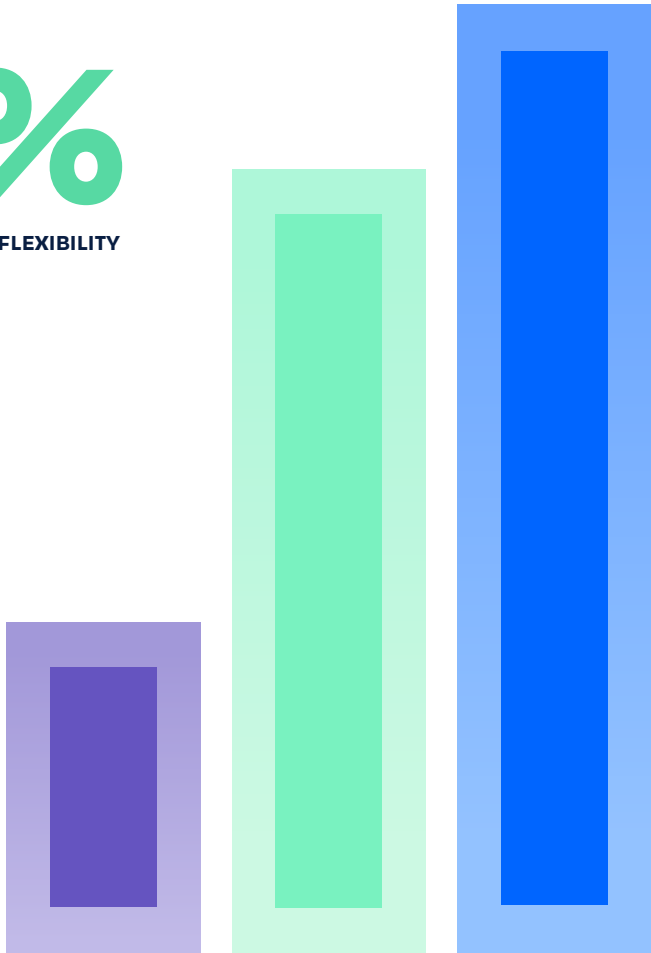
STABLE TOOLCHAIN

38%

MORE TOOLS, INCREASED FLEXIBILITY

16%

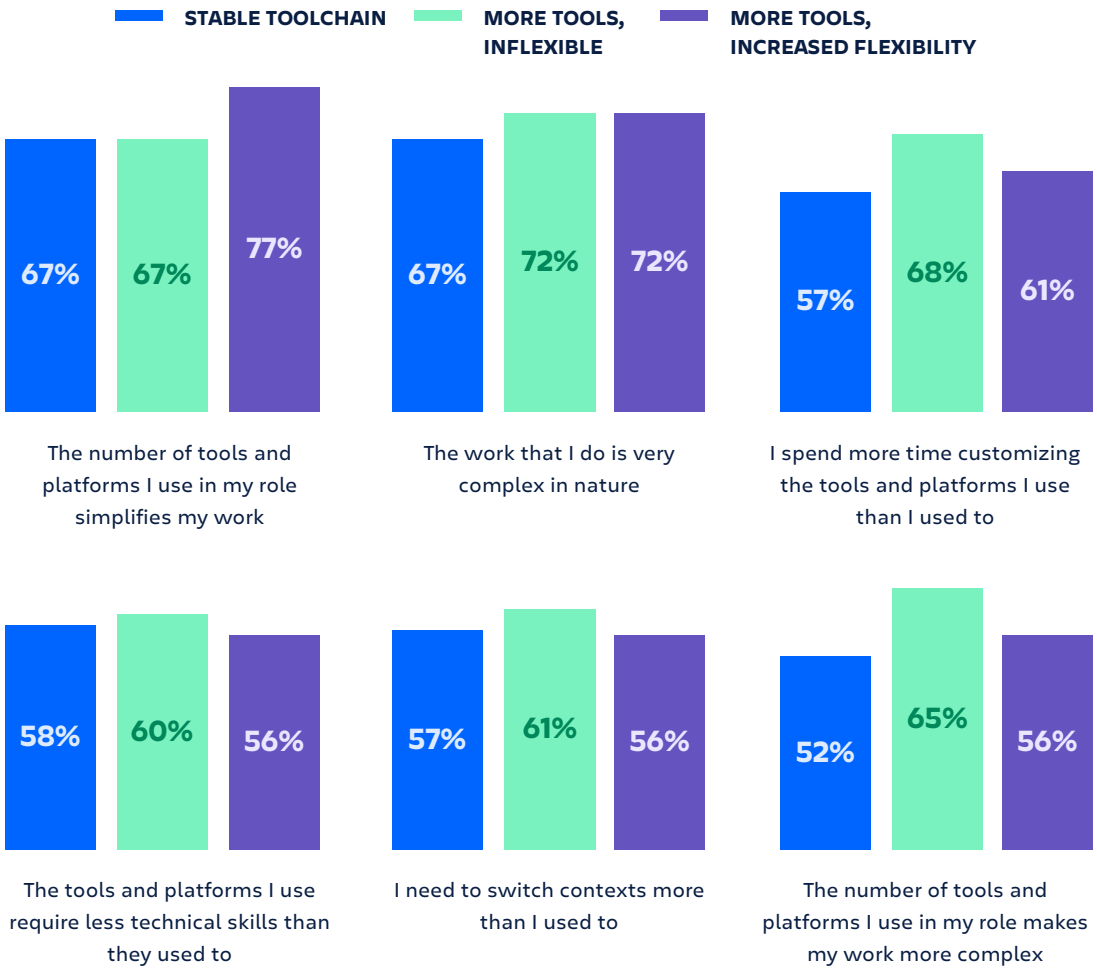
MORE TOOLS, INFLEXIBLE



TAKEAWAYS

Teams with increasing and expanding toolchains are more likely to find these tools flexible and adaptable rather than inflexible.

Attitudes on toolchain complexity
(% of respondents agreeing with the following statements)



TAKEAWAYS

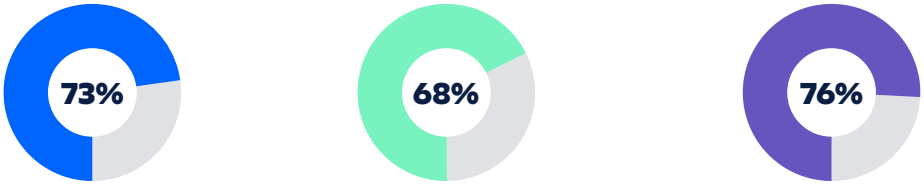
Developers with increased access to flexible tools say this simplifies their work.

Those now using more inflexible tools complain that this complicates their roles and requires excessive maintenance.

Flexible tools make developers happy

The group that has grown their toolchain with more flexible and adaptable tools is happiest (76%). Those now using a sprawling and inflexible toolchain are significantly less happy with their roles and responsibilities (68%).

Developer happiness - split by toolchain stability and flexibility



Stable toolchain

More tools, increased flexibility

More tools, inflexible

"I am happy with the current mix of tasks and responsibilities I have"

TAKEAWAYS

Increased access to flexible tools makes developers happier with their role and responsibilities, while those burdened with inflexible, difficult to integrate tools are less happy.

04

The need for greater autonomy within development teams

Autonomy

Empowering employees with a strong sense of autonomy has the potential to improve job performance and motivation. When applied to software development, an autonomous culture has the potential to improve job satisfaction, speed of delivery, and team agility.

In our study we examined three areas of autonomy:

- Choosing the tools used at work
- How you work
- What you work on

Using these three metrics we created three levels of autonomy:

- **Strong autonomy** – Score at least of 4 out of 5 for each area of autonomy listed above
- **Moderate autonomy** – Strong autonomy, or score of at least 4 out of 5, in some of the areas listed above, but not all
- **Weak autonomy** - Score of 1, 2, or 3 out of 5 for each area of autonomy listed above

Using this allocation, roughly half of the developers we surveyed indicated strong autonomy. The remainder is split fairly evenly between weak and moderate autonomy. Autonomy is highest in the US and India (57% and 56% respectively) and lowest in Germany (29%).

Autonomy Level

50%

STRONG AUTONOMY: 50%

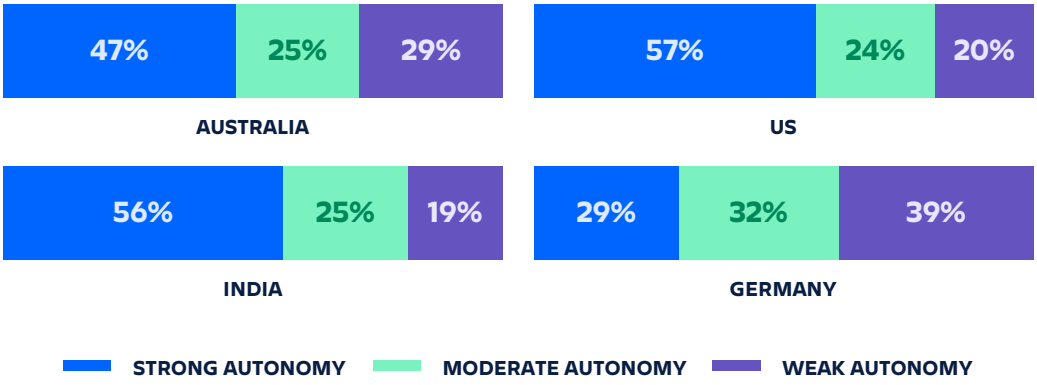
26%

MODERATE AUTONOMY: 26%

24%

WEAK AUTONOMY: 24%

Autonomy level by country



TAKEAWAYS

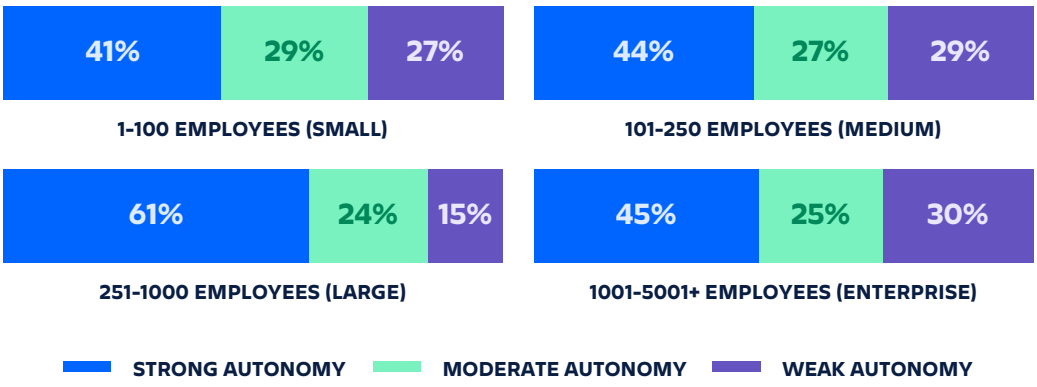
Autonomy levels for developers is high, with 50% claiming strong autonomy.

Only 29% of developers in Germany indicate strong autonomy.

Autonomy by company size, tenure, and YBIYRI

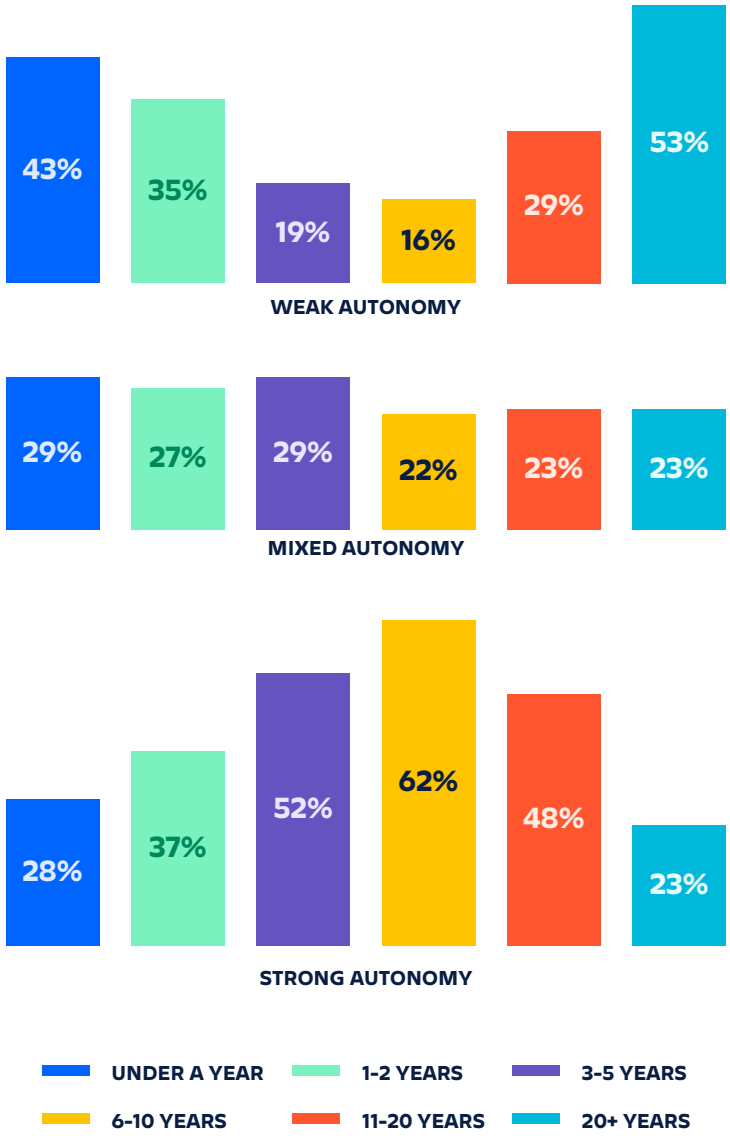
Larger companies (250-1000 employees) below enterprise size report much stronger autonomy than small-medium or enterprise organizations. This is likely because smaller companies can maintain greater levels of oversight, while enterprises have clearer procedures and toolchains.

Autonomy level by company size



This pattern is evident in all facets of autonomy that we measured, but is starkest in tooling and what these developers work on. Greater autonomy is also related to an increase in tenure, but only to a point. Autonomy is strongest for developers 6-10 years into their career, before declining again.

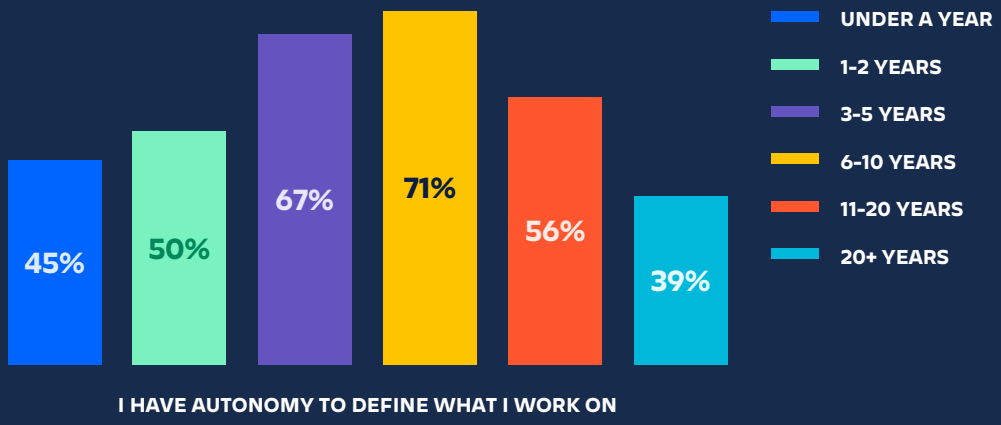
Autonomy level by tenure



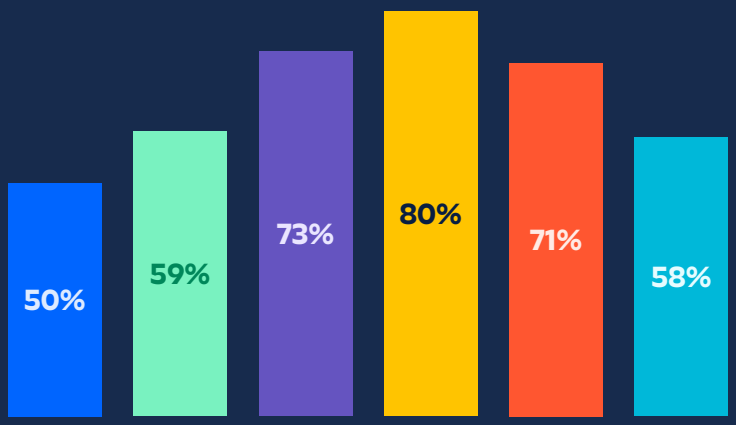
This pattern is evident in all facets of autonomy that we measured, but is starkest in tooling and what these developers work on.

Autonomy attributes by tenure

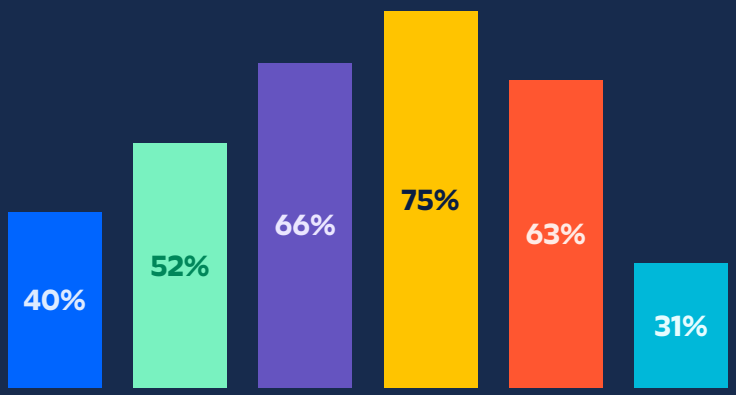
(% of respondents agreeing to the following statements)



I HAVE AUTONOMY TO DEFINE WHAT I WORK ON



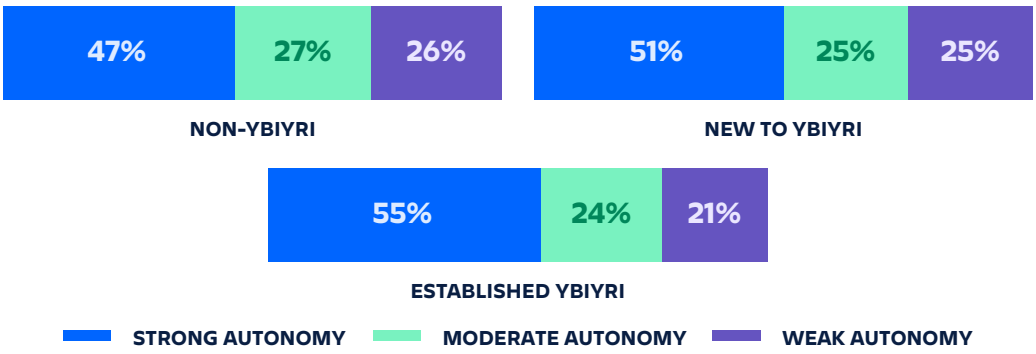
I HAVE AUTONOMY TO DEFINE THE WAY I WORK



I HAVE AUTONOMY TO CHOOSE WHICH TOOLS I USE IN MY WORK

Referring back to the YBIYRI groups that we discussed in Chapter 1, we see that those teams running YBIYRI have greater autonomy levels. Established YBIYRI teams have greater autonomy than those new to the practice, indicating that autonomy increases with time.

Autonomy level by YBIYRI groups



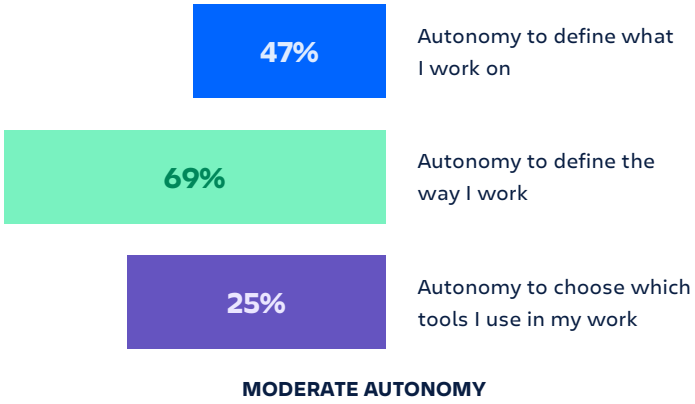
TAKEAWAYS

Autonomy levels are highest for developers who have been in their roles for 6-10 years, within larger companies (250-1000 employees), and in teams running YBIYRI.

Tool choices limit autonomy

Developers with moderate autonomy show much more autonomy in defining how they work (69%) than what they work on (47%) or the tools they use (53%).

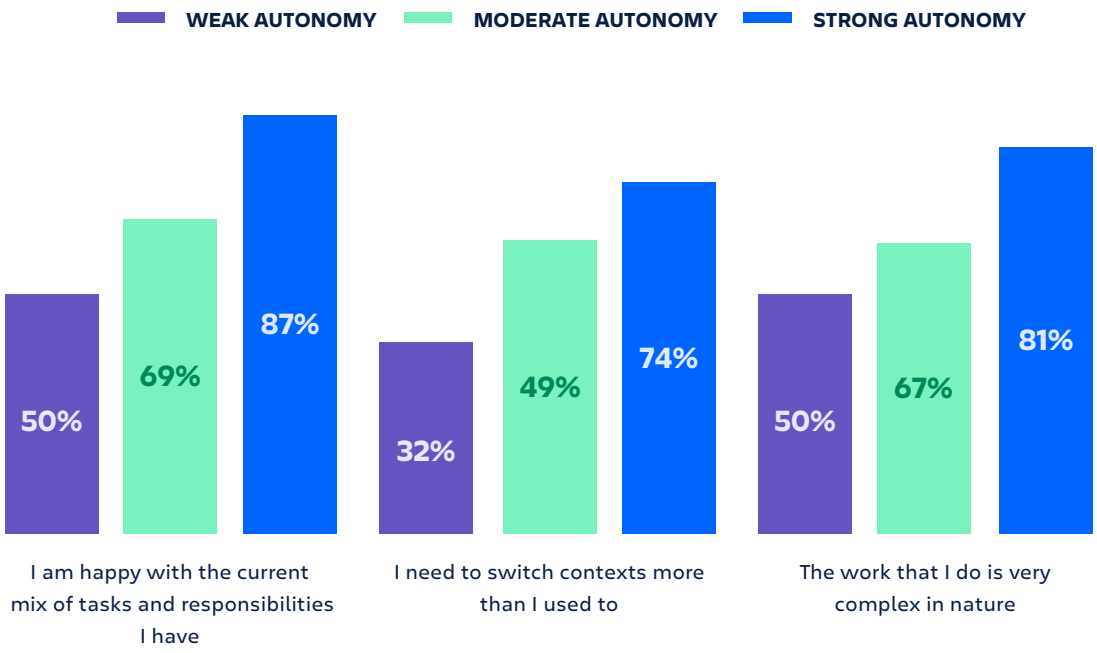
Autonomy for developers with moderate autonomy



Autonomy makes even complex roles feel more satisfying

Stronger autonomy is strongly correlated with positive feelings about work. Developers with more autonomy say they are happier than those with less autonomy, despite greater context switching and job complexity.

Happiness and complexity by autonomy level



TAKEAWAYS

Developers without strong autonomy are less likely to decide what they work on or what tools they use.

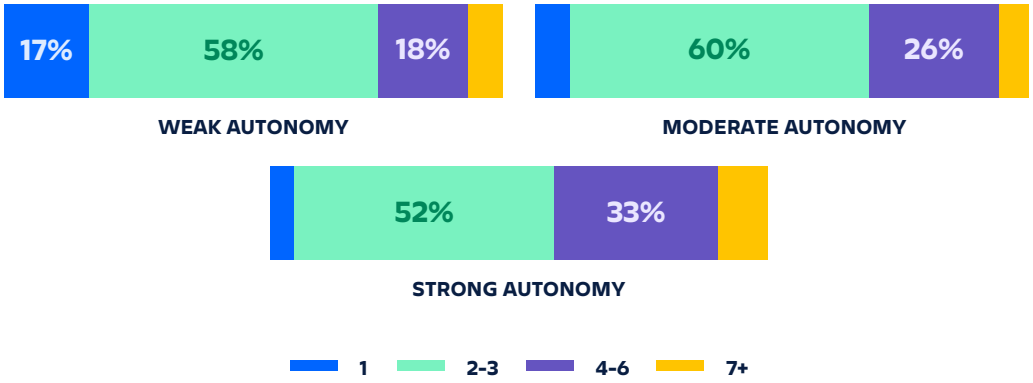
Developers with greater autonomy are happier, even though they are more likely to see their roles as more complex.

Autonomy and time spent on coding are positively correlated

Developers with stronger autonomy are also more likely to spend their time on code-related activities as well as learning new technical skills. Those with less autonomy are more likely to communicate and coordinate with their team more frequently, but tend to work on code less often.

This may explain why developers with more autonomy tend to work on a greater number of products than those with weaker autonomy.

Number of products worked on by autonomy level



TAKEAWAYS

Developers with strong levels of autonomy are more likely to spend their time on code-related activities and work on more products.

Conclusion

Software developers are a diverse group with a range of different perspectives on the future of software development. Team leaders need to take this into account when planning how work gets done, but our first State of the Developer report has highlighted some common themes to help frame those discussions.

The first is that the majority of development teams are now running “You build it, you run it” (YBIYRI) models, where developers expect to play a more prominent role in supporting the products they build. This brings them closer to customers and creates a feedback loop that improves products.

Teams transitioning to YBIYRI typically add a range of new roles including front-end and back-end developers, security engineers, project managers, and solution architects. As these teams mature, the focus switches to bringing in more network, hardware, and DevOps engineers.

Developers in established YBIYRI teams are more likely to feel that their work is complex. They want to spend more time writing new code and less time maintaining the product.

Some developers have a strong preference for coding or tooling, while others operate in a place of plurality. Two-thirds of developers say that writing code is the most valuable skill in their role, but more than half think writing code from scratch won’t be needed in the future.

Tool sprawl has typically been associated with negative outcomes including wasted effort, lost productivity, and lack of efficiency. Tool sprawl is worsening with a majority of developers saying they use more than 6 different tools – and it’s only increasing. However, developers with increased access to more flexible tools say it simplifies their work, making them happier with their roles and responsibilities.

Finally, managers and team leaders should note that greater autonomy improves job satisfaction, speed of delivery, and team agility. While half of developers say they have strong autonomy today, there’s still room for improvement. There are three ways to do this – giving developers more freedom to choose the tools they use, how they work and what they work on.

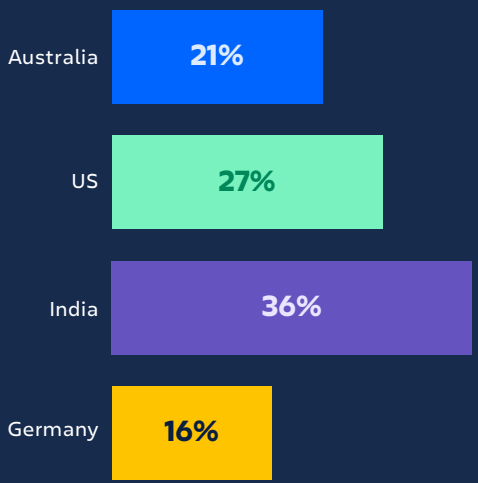
Background and methodology

A survey was run for approximately a month between August and September of 2021, targeting 2,182 respondents across 4 countries: Australia, India, Germany, and the United States. Respondents were screened to ensure that they were in a traditional software developer role. Quotas were used to ensure a mix of respondents from various organization sizes, levels of seniority, industries, and degrees of role tenure as well as a representative gender mix.

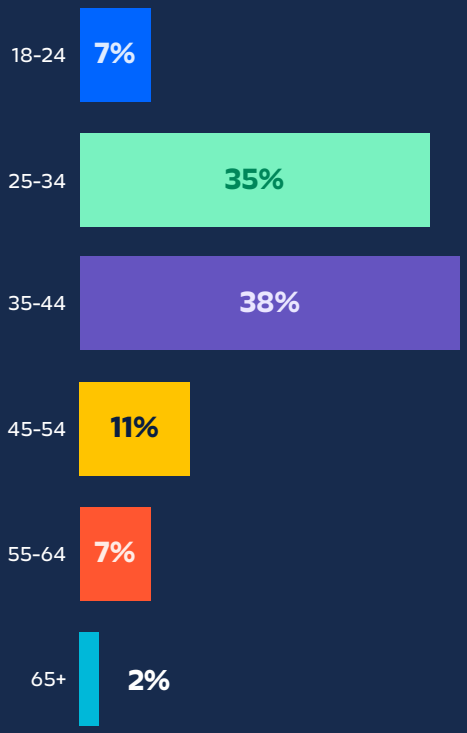
Sample

The final sample of respondents (developers working in teams that primarily produce software) was n=2,182. This was split across the four countries we included but does skew towards India (36%) and the US (27%).

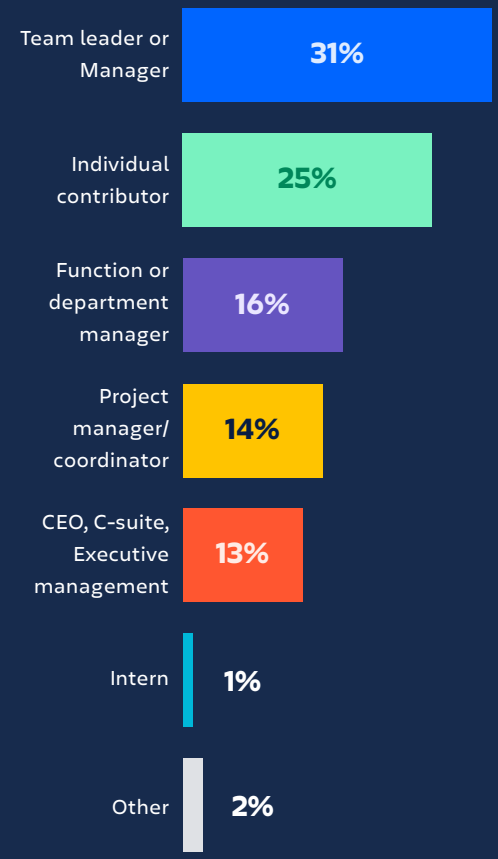
Sample - country



Sample - age



Sample - role seniority



Learn more at
atlassian.com

ATLASSIAN