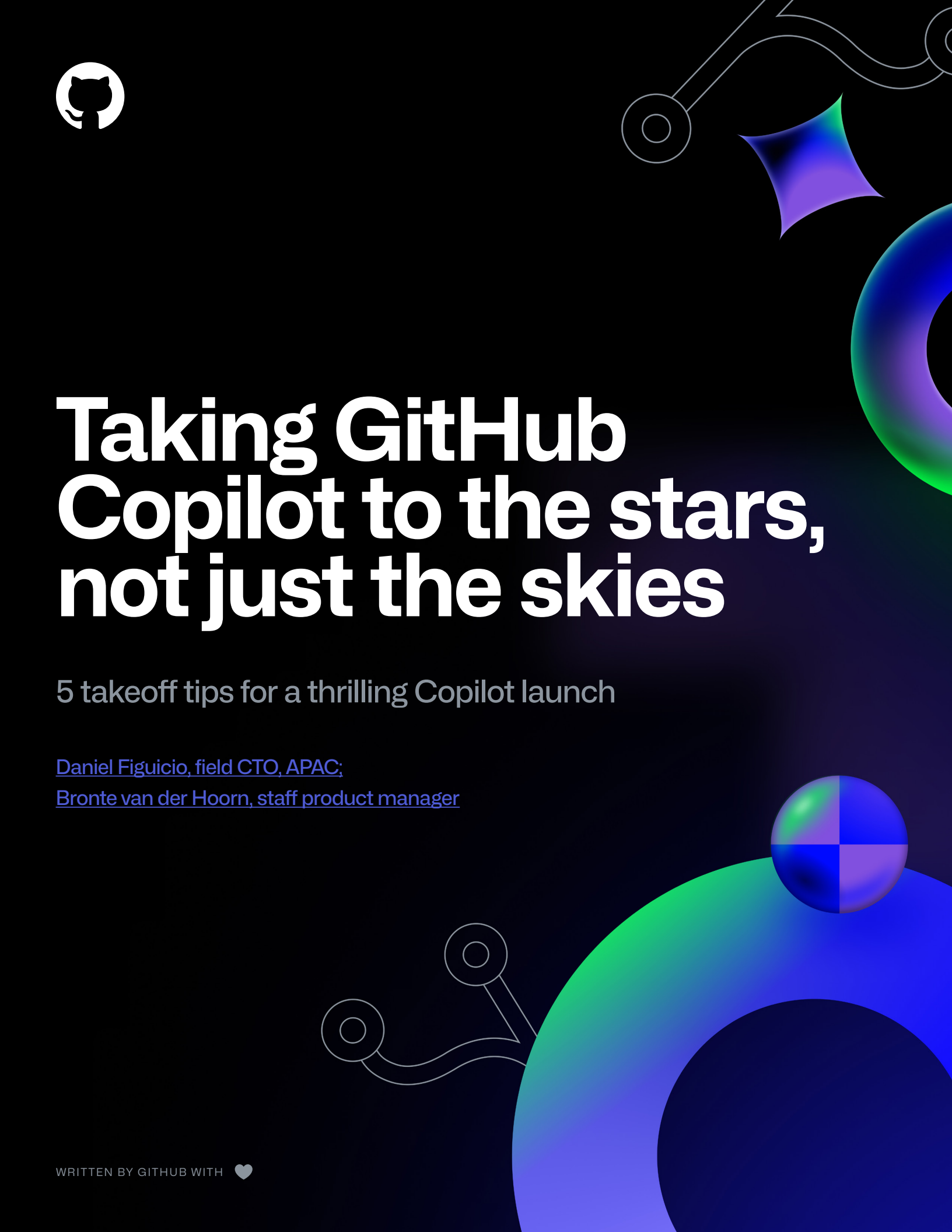


Taking GitHub Copilot to the stars, not just the skies

5 takeoff tips for a thrilling Copilot launch

Daniel Figuicio, field CTO, APAC;
Bronte van der Hoorn, staff product manager

WRITTEN BY GITHUB WITH ♥

Table of contents

Executive summary

AI-assisted coding can transform your software development processes and outcomes. This article discusses five tips to support the successful scaling of GitHub Copilot across your organization to enable the realization of these outcomes.

Whether you are looking to accelerate code generation, streamline problem-solving or improve code maintainability, by implementing Copilot thoughtfully and systematically, you can maximize the benefits of Copilot while helping to mitigate potential risks—supporting a smooth integration that propels development teams to new heights of productivity and innovation.

Introduction: Preparing for a successful GitHub Copilot launch

The impact of GitHub Copilot on the developer community has been nothing short of transformative. Our data reveals that Copilot significantly boosts developer efficiency by up to 55% and enhances confidence in code quality for 85% of users. With the rollout of Copilot business in 2023, and the introduction of Copilot Enterprise in 2024, it's our priority to support every organization in seamlessly integrating Copilot into their workflow.

To establish a successful launch, securing endorsements from management and security teams, allocating budgets, completing purchases, and adhering to organizational policies are essential. However, there's more you can do to foster a smooth launch.

The excitement around the impact of Copilot is palpable. It's not just about speeding up development; it's about enhancing the quality of work and boosting developer confidence. As we introduce Copilot to more businesses and organizations, our focus is on helping to facilitate a seamless integration for everyone.

Early planning is crucial for smooth adoption. Initiating discussions with management and security teams, planning budgets, and navigating the purchase process should begin well ahead of time. This foresight allows for comprehensive planning and ensures adherence to your organization's policies, paving the way for less friction for Copilot integration.

By starting these discussions and planning phases early, you can ease the transition and proactively address potential hurdles. This preparation ensures that by the time Copilot is ready to be rolled out to your teams, everything is in place for a successful launch.

In this guide, we'll share strategies gathered from organizations of all sizes that have successfully integrated Copilot into their development processes. By following these steps, you can not only streamline your Copilot rollout but also maximize its long-term benefits for your teams.

Don't wait until the last minute—begin preparing now to unlock the full potential of Copilot and create a seamless experience for your developers from day one.

Tip #1: To build trust, transparency's a must

It's natural for teams to be curious (and sometimes skeptical) about the adoption of a new tool like GitHub Copilot. To create a smooth transition, your announcements should clearly articulate the reasons for adopting Copilot -- be honest and transparent. This is an excellent opportunity for leaders to reinforce the organization's engineering goals, whether they are focused on improving quality, increasing development speed, or both. This clarity will help teams understand the strategic value of Copilot and how it aligns with organizational objectives.

Key strategies for building trust:

- **Clear communication from leadership:** Clearly state the reasons for adopting Copilot. Explain how it will help the organization achieve its goals, whether that is enhancing code quality, speeding up development cycles, or both.

  Use relevant organizational channels to announce the adoption. This could include emails, team meetings, internal newsletters, and collaboration platforms.

- **Regular Q&A sessions:** Hold regular Q&A sessions where staff can voice concerns and ask questions. This encourages open communication and addresses any skepticism or uncertainties.

  Use the insights from these sessions to update your rollout program, continuously refining your FAQs and other support materials based on your team's feedback.

- **Align measurements with goals:** Ensure that the metrics you track align with your Copilot adoption goals. For instance, if your goal is to improve code quality, track metrics related to code review efficiency and defect rates.

  Demonstrate consistency between what you say and what you measure - this builds trust and shows that you are serious about the benefits Copilot can bring.

- **Ongoing reminders and training:** Use reminders and training materials to continually reinforce the adoption goals. This could include periodic updates, success stories, and practical tips on leveraging Copilot effectively.

  Provide comprehensive resources, such as guides, tutorials, and best practices, to help teams get up to speed with Copilot (more on this below).

Sample communication plan

- **Initial announcement:**

  Message: "We are excited to announce the adoption of GitHub Copilot to enhance our development processes. This tool will help us achieve our goals of improving code quality and accelerating our release cycles. Your participation and feedback are crucial for a successful rollout."

- **Channels:** Email, internal newsletter, team meetings.

- **Regular Q&A sessions:**

  Message: "Join our Q&A session to learn more about GitHub Copilot and how it can benefit our team. Share your questions and feedback to help us address any concerns and improve the integration process."

- **Channels:** Video conferences, company intranet.

- **Progress updates and metrics:**

  Message: "We are tracking key metrics to ensure GitHub Copilot is helping us achieve our goals. Here are the latest updates on our progress and how Copilot is making a difference."

- **Channels:** Monthly reports, dashboards.

- **Training and resource distribution:**

  Message: "Check out our new training materials and best practices guide for using GitHub Copilot. These resources are designed to help you make the most of this powerful tool."

- **Channels:** Internal wiki, email, training sessions.

Don’t just listen to us...

Writing tests is one realm where Accenture’s developers have found GitHub Copilot to be extremely useful. “It’s allowed us to take the time to create all the unit tests, functional tests, and performance tests that we want in our pipelines without having to go back and effectively write double the code. There’s never been enough time in the past to go back and get to all of them,” said Schocke.

In addition to writing tests, Copilot has also allowed Accenture’s developers to tackle the ever-increasing technical debt that challenges any organization of its size.

“We have more work than developers. We just can’t get to it all,” said Schocke. “By augmenting our developers’ skills and helping them to produce features and functions more quickly with higher quality, we’re able to get to more of the work that just didn’t happen before.”

Daniel Schocke | Application Architect, Accenture | Accenture

Accenture & GitHub case study

Summary

To build trust, clearly communicate the reasons for adopting GitHub Copilot and how it aligns with your organization’s goals. Providing regular updates, open Q&A sessions, and ongoing training will help your team feel at ease and address any concerns.

Tip #2: Tech readiness, in this, we entrust

Leverage GitHub's comprehensive documentation to help streamline the onboarding process for GitHub Copilot, ensuring it's as smooth as possible for your developers. Engage a group of early adopters to identify potential friction points (e.g., network settings) and address these issues before a wider rollout.

Key strategies for nailing tech readiness:

- **Early adopter observation:** Treat your early adopters like customers, closely observing their onboarding experience. Look for any friction points that might hinder the process, such as configuration issues or network settings.

  Establish a feedback loop for early adopters to share their experiences and suggestions. This will provide valuable insights into potential obstacles and areas for improvement.

- **Resolve issues promptly:** Consider forming a small task force dedicated to resolving any issues identified by early adopters.
  This team should have the authority and resources to act quickly on feedback.

  Use the feedback to update and enhance the organization's tailored onboarding documentation, making it more comprehensive and user-friendly.

- **Gradual rollout:** Start with a small group of users to better support an onboarding process that is smooth and efficient. Gradually scale up as you mitigate most issues, leaving only edge cases.

  Continuously refine the process based on feedback and observations, ensuring a seamless experience for the broader team.

- **Feedback mechanism:** Provide easy-to-use feedback forms or surveys for those onboarding to Copilot. Regularly review this feedback to identify trends and common issues.

  Act on feedback swiftly to show that you value user input and are committed to improving their experience.

Hear it from them...

“We built an automated seat provisioning and management system to meet our specific needs. We wanted any developer at ASOS who wants to use GitHub Copilot to be able to with as little friction as possible. But we didn’t want to turn it on for everyone at the organization level as that would be a pretty inefficient use of resources. So we built our own self-service system.

We have an internal website where every employee has a profile. To receive a GitHub Copilot seat, all they have to do is click a single button on their profile. Behind the scenes, that kicks off a Microsoft Azure Functions process that validates the developer’s Azure token and calls the GitHub Copilot Business API to provision a seat. Developers can even do this from the command line, if they prefer.

At the same time, we have an Azure function that checks for inactive accounts nightly by pulling the seat usage data. If a seat hasn’t been used for 30 days, we mark it for deletion before the next billing period begins. We check one last time for activity before deletion and then send an email to all developers whose seats are revoked. If they want a seat again, they can just click that button and start the process over again.”

Dylan Morley | lead principal engineer | ASOS

ASOS & GitHub case study

Summary

To create a smooth GitHub Copilot onboarding, leverage GitHub’s documentation and involve early adopters to identify potential issues before rolling it out to the entire organization. Implementing a robust feedback mechanism will help you refine the process and continually enhance the experience.

Tip #3: Training tips, a guiding light

Providing training materials in the engineer's native coding language is incredibly impactful, especially when it demonstrates GitHub Copilot in contexts relevant to their daily workflows. Moreover, training doesn't have to be limited to formal videos or learning modules; peer-shared 'wow' moments and practical tips can be particularly powerful. Ensure these resources are readily available as you roll out Copilot across your teams. If you need help building the right training program or tailoring training specific to your organization, our GitHub Experts are available to help.

Key strategies for supercharging training:

- **Tailored training materials:** Create training materials that are specific to the coding languages and frameworks your engineers use daily. This contextual relevance makes the training more engaging and practical.

  Make these materials easily accessible, whether through an internal portal, shared drive, or directly in the tools your developers use. Providing links to these resources when provisioning seats is a great practice.

- **Peer sharing:** Encourage a culture of sharing within your team. Have developers share their 'wow' moments and tips with Copilot in team meetings, chat groups, or through internal blogs.

  Compile these peer experiences into a repository of success stories that others can learn from and be inspired by. Start building your own Community to share successes, best practices and governance for Copilot for your own organization

- **Regular updates and communication:**
  Keep everyone informed about what Copilot is achieving within your organization (including any milestones that your measurements have shown you've reached). Use email newsletters, organizational newsfeeds, or internal social platforms to provide regular updates.

  Highlight specific successes and improvements (either qualitative or quantitative) brought about by Copilot. This not only builds enthusiasm but also demonstrates the tool's value in real-world scenarios.

- **Implementation steps:**

  Provisioning resources: When providing a Copilot seat, include links to role-specific training materials in the developer's native language.

  Frequent communication: Be proactive in communicating the benefits and successes of Copilot within your organization. Regularly update the team on new features, user tips, and success stories through newsletters or internal newsfeeds.

  Encourage peer learning: Foster an environment where developers can share their positive experiences and tips with each other. Organize informal sessions where team members can discuss how they're using Copilot effectively.

Success speaks for itself...

"When we went to roll out GitHub Copilot to Cisco's 6,000 developers in our business group, they were eager and excited, but had plenty of questions. We partnered with our GitHub Premium Support team to host a series of training sessions where they explained how to get started with GitHub Copilot, provided best practices for writing useful prompts, and demonstrated its unique capabilities, followed by a Q&A. Soon enough, our developers were confidently using GitHub Copilot throughout their day-to-day development. What really helped us was getting a sense of our developers' questions and concerns beforehand, and keeping our sessions high level, to address initial concerns during our Q&A session."

Brian Keith | head of engineering tools, Cisco Secure | Cisco

Cisco & GitHub case study

Summary

Training materials are crucial–tailor them to the languages and frameworks your developers use daily. Foster a culture of sharing 'wow' moments among your team and make sure to provide regular updates on achievements and milestones your organization has reached using GitHub Copilot.

Onboarding to a new technology tool takes time, and while we've streamlined the process as much as possible, engineers still need dedicated time to set up GitHub Copilot in their work environment. It's essential to create excitement and opportunities for engineers to experiment with Copilot and see how it fits into their workflow. Expecting engineers to onboard to GitHub Copilot while under unrealistic delivery pressure is impractical; everyone needs time to integrate new tools into their practice effectively.

Key strategies for enabling bonding

- **Allocate dedicated time:** Ensure engineers have dedicated time to onboard to Copilot. This should be scheduled during periods when they are not under tight delivery deadlines to prevent multitasking and ensure full engagement.
- **Create excitement and encourage experimentation:** Foster a sense of excitement around Copilot by highlighting its potential benefits and encouraging engineers to experiment with it. Share success stories and examples of how it can enhance their workflow.

- **Provide comprehensive resources:**
  Offer a variety of resources to help engineers get started:
  - Share videos demonstrating how to install and set up the GitHub Copilot plugin.
  - Provide content showing relevant examples tailored to the developer's specific coding environment.
  - Encourage engineers to write their first piece of code using GitHub Copilot, starting with simple tasks and progressing to more complex scenarios.

- **Organize dedicated onboarding sessions:**
  Schedule onboarding sessions, such as a morning or afternoon, where engineers can focus solely on setting up and exploring Copilot. Make it clear that it's acceptable to dedicate this time to learning and experimentation.

- **Encourage peer support and sharing:**
  Create channels for engineers to share their onboarding experiences and tips with each other, such as Slack or Teams. This peer support can help address common challenges and enhance the onboarding experience. Consider organizing a GitHub Copilot hackathon to encourage collaborative learning and innovation.

- **Regular check-ins and feedback:**
  Conduct regular check-ins to gather feedback on the onboarding process and identify any areas that need improvement. Use this feedback to continuously refine and enhance the onboarding experience.

Sample onboarding schedule:

Day 1: Introduction and setup

- **Morning:** Watch a video tutorial on installing and setting up GitHub Copilot.

- **Afternoon:** Install and configure the plugin in your development environment.

Day 2: Learning and experimentation

- **Morning:** Watch content showing relevant examples of GitHub Copilot in action.
- **Afternoon:** Write your first piece of code using Copilot (e.g., a slightly more complex "Hello World" scenario).

Day 3: Practice and feedback

- **Morning:** Continue experimenting with GitHub Copilot and integrate it into your current projects.
- **Afternoon:** Post a "how did I do" entry in the Copilot onboarding channel (Slack, Teams, etc.) and provide feedback.

Read between the lines...

Mercado Libre invests in the next generation of developers by offering its own two-month "bootcamp" for new hires to help them learn the company's software stack and solve problems the "Mercado Libre way." While GitHub Copilot can help more experienced developers to write code faster and minimize the need for context switching, Brizuela sees a vast amount of potential in GitHub Copilot to accelerate this onboarding process and flatten the learning curve.

Lucia Brizuela | Senior Technical Director | Mercado Libre

Mercado Libre & GitHub case study

Summary

Allocate dedicated time for your team to onboard and experiment with GitHub Copilot when they're relaxed and not under pressure. Foster excitement and provide resources—including comprehensive guides and hands-on sessions—to help them integrate Copilot into their workflow effectively.

Tip #5: Teams share AI wins, in tools we trust

Most of us are influenced by peer pressure and the opinions of those we regard as experts -- similar to the impact of influencer endorsements and product reviews. GitHub Copilot is no different. Engineers seek validation from their peers and respected colleagues to ensure that using Copilot is valuable and supports their identity as accomplished professionals.

Key strategies for promoting collaborative AI adoption within teams:

- **Encourage peer-to-peer support and story sharing:** Allow your early adopter team to share their experiences with Copilot. Encourage them to discuss how it has enriched their professional lives beyond just increasing coding speed. What additional activities have they been able to undertake thanks to the time saved with Copilot?

  Highlight stories where Copilot has enabled engineers to focus on more creative or high-impact tasks that were previously time-consuming or overlooked. It's wonderful if there are linkages between Copilot and being able to better serve the organization's customers.

- **Share learnings and organizational tips:** Distribute tips and tricks specific to your organizational scenarios. Share practical advice on how GitHub Copilot can address unique challenges or streamline workflows within your team.

  Foster a culture of continuous learning by regularly updating and sharing best practices based on real user experiences.

- **Integrate Copilot into organizational culture and performance frameworks:** Make the use of Copilot and the sharing of Copilot practices a part of your organizational culture. Recognize and reward those who contribute valuable insights and improvements.

  Ensure engineers know that using Copilot is supported and encouraged by management. This assurance can come through endorsements from senior leaders and integration into performance reviews and goals.

Straight from the source...

Carlsberg's developmental workflow. GitHub Copilot seamlessly integrates within the development process, providing valuable coding suggestions directly from the IDE, further removing development roadblocks. Both Peter Birkholm-Buch, the company's Head of Software Engineering and João Cerqueira, one of Carlsberg's engineers, reported that Copilot significantly enhanced productivity across the team. The enthusiasm for the AI coding assistant was so unanimous that as soon as enterprise access was available, Carlsberg immediately onboarded the tool. "Everyone immediately enabled it, the reaction was overwhelmingly positive," shares Birkholm-Buch. It's now challenging to find a developer who wouldn't prefer to work with Copilot, he says.

Peter Birkholm-Buch | Head of Software Engineering | Carlsberg
João Cerqueira | Platform Engineer | Carlsberg

Carlsberg & GitHub case study

Summary

Encourage early adopters to share their experiences with GitHub Copilot and highlight the benefits they've experienced. Integrate Copilot into your organizational culture by sharing tips, recognizing contributions, and ensuring strong management support.

Putting it all together: Mission Control for GitHub Copilot success

You're now ready to undertake your preflight checks. Build trust in the tool's purpose, address technical barriers, provide resonant training materials, allocate time for setup and exploration, and foster team-wide usage. These checks will support achieving to the maximum the impact of Copilot in your organization. When you undertake these checks you help set up your engineers for success and enable your organization to derive maximum long-term impact from Copilot.

Additional resources

Looking for more GitHub Copilot goodness? Check out these additional resources to supercharge your Copilot journey:

- Setting up GitHub Copilot for your organization Docs page
- How to use GitHub Copilot Enterprise full demo video
- Subscribing to Copilot for your organization Docs page
- Introduction to GitHub Copilot Enterprise tutorial
- GitHub Copilot for Business is now available announcement blog
- Subscription plans for GitHub Copilot Docs page
- GitHub Copilot pricing page
- Found means fixed: Introducing code scanning autofix, powered by GitHub Copilot and CodeQL blog post
- How Duolingo increased developer speed by 25% with Copilot customer story

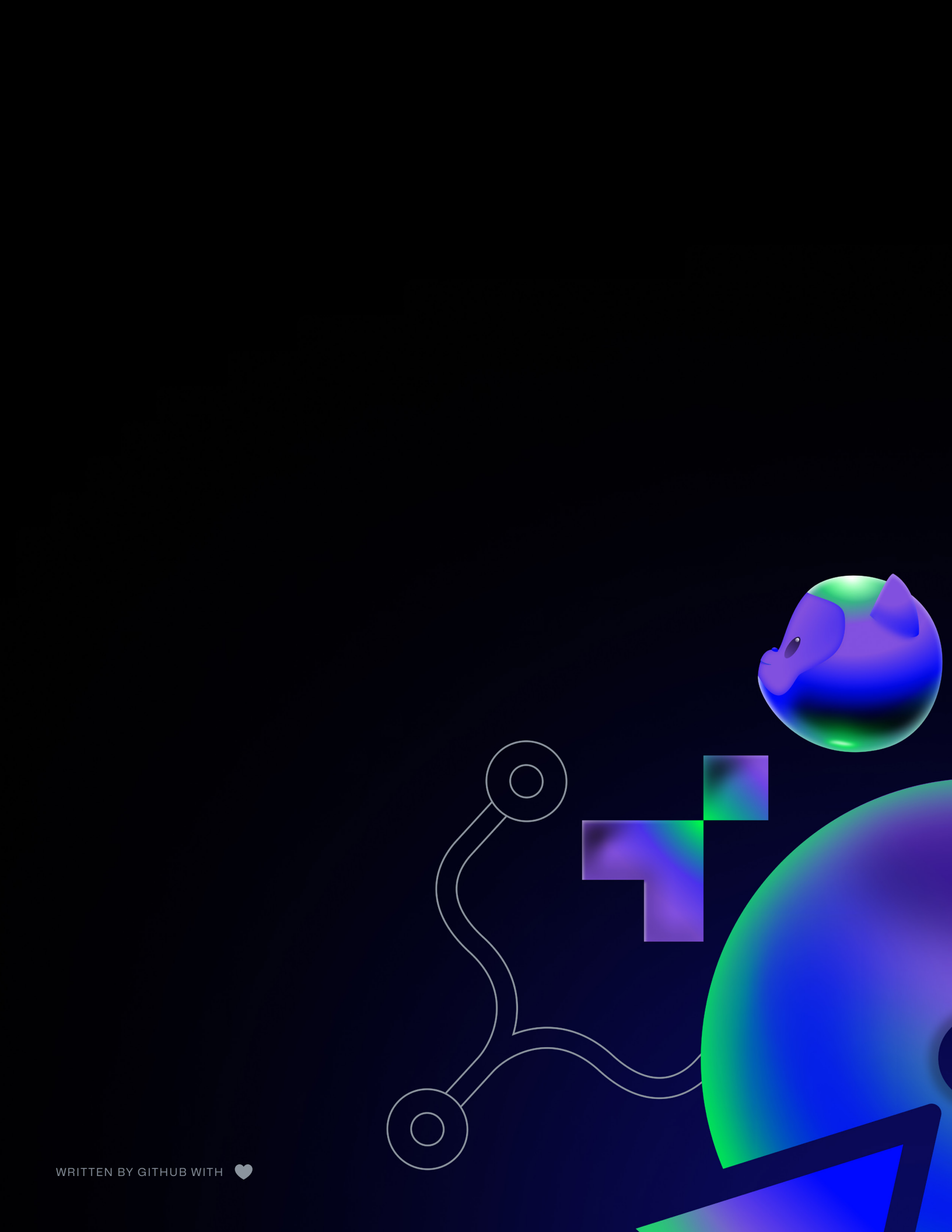
About the authors

Daniel Figucio is the field chief technology officer (CTO) for Asia-Pacific (APAC) at GitHub, bringing over 30 years of information technology (IT) experience, including more than 20 years in the vendor space. He is passionate about helping the hundreds of developer teams with which he gets to engage across the region through implementing strong developer experience methodologies and technologies. Daniel's expertise spans the entire software development lifecycle (SDLC), leveraging his background in computer science and pure mathematics to optimize workflows and productivity. His programming journey has evolved from C++ to Java and JavaScript, with a current focus on Python, enabling him to provide comprehensive insights across diverse development ecosystems.

As one of the founding members of GitHub's APAC team, Daniel has been instrumental in driving the company's growth in the region from its inception over 8 years ago, when the team consisted of just two people. Based in the Blue Mountains of New South Wales, Australia, Daniel balances his commitment to enhancing developer experiences with interests in gaming, outdoor activities like cycling and bushwalking, and culinary exploration.

Bronte van der Hoorn is a staff product manager at GitHub. She leads a diverse range of multidisciplinary projects across GitHub Copilot. Bronte is committed to helping customers unlock the full potential of AI, while enhancing engineers' satisfaction and flow through amazing tooling.

With extensive industry experience, a PhD, and a portfolio of publications on management topics, Bronte combines research insights with practical know-how. This approach supports her in designing and iterating on features that are aligned with the complex demands of modern business environments. An advocate of systems thinking and a champion of collaborative work practices, Bronte fosters innovation by promoting a holistic and contemporary perspective to organizational change.



WRITTEN BY GITHUB WITH ♥