

← [rss-offline](#) · [abrir original](#) · **Basecamp Five [rendered]** · David Heinemeier Hansson · 2026-05-26 07:52



DAVID HEINEMEIER HANSSON

May 26, 2026

Basecamp Five

I've been working on [Basecamp](#) for half my life, and nearly my entire professional career in software. The first code was written in the summer of 2003 when I was just 23. Now I'm 46, and we've just released the fifth major version.

It's an incredible update to a service that continues to help about a million users a day avoid dropping the ball when working with others. It's AI accessible, but not agent hysteric. It's still famously easy to use, still executes the basics beautifully, and still focuses on the small to medium-sized teams we've been serving in the [Fortune 5,000,000](#) for decades.

Here are just three of my favorite new features in Basecamp 5:

Lexxy editor: Our [new text editor](#) finally brings tables, markdown, and live syntax highlighting for code to Basecamp. Oh, and voice notes. It's built on Meta's Lexical editor toolkit, and it's going to ship as the default for Action Text in the next major version of Rails.

Keyboard accessible: After moving to Linux, building [Omarchy](#), and acquiring a taste for mechanical keyboards, I've come to love navigating the computer primarily through hotkeys. So with a lot of effort, [Basecamp is now a delight to drive through the keys](#), and you don't have to be a brainiac to remember them all: just hold down SHIFT, and they're revealed in the interface. SHIFT + S opens the sidebar, ESC moves focus between it and the main page, SHIFT + C starts composing a comment/chat line/answer.

The permanent sidebar: If you live in Basecamp, like I do, it's to stay on top of all the new things that are constantly happening in a busy account, and that's just gotten so much faster with the new permanent sidebar. Before, we had a Hey! menu in the top bar. You'd get a little dot when something was new, then you'd open it, click, and the menu would close. If you had five things that were new, it'd be open-click-close, open-click-close, five times. Being able to zoom through these now with just the return key, tap, tap, tap, and I've read three new things. So good.

And there's so much more. Jason put together a [great summary](#) on the new marketing site, which in itself is brand new too. A back-to-basics design in many ways. As our entire industry is getting swept up in agent hysteria (and I love AI as much as anyone!), we thought it better to focus on the human communication that's the cornerstone of Basecamp. The new site just speaks plainly to that mission and shows you the software right at the top.

Another thing that's back is color, specifically in the logo. Basecamp's clever but flat paperclip logo has been replaced with a modern take of our original rolling mountains. In full three dimensions, with depth and a gradient. Love it.

Overall, I'm really proud of what we've built with Basecamp Five. We're inching in on a quarter of a century in service! We still have customers who signed up back in early 2004! This is the [kind of legacy](#) that makes me beam, and the new version is just ace.

If you've tried Basecamp in the past, [it's time to take another look](#). If you haven't tried it yet, [you're in for a treat](#).



Pricing — Two paid plans, one free plan

Basecamp 5 is here — Major upgrade for 2026

Features — Remarkably simple, surprisingly capable

Paths — Why people switch to Basecamp

API, CLI, Skills — Developer tools, AI Agent-ready

Reliable to the core — A multi-decade track record

The refreshingly straightforward project management system that's rock-solid and easy to use.

Website Redesign Project ★

Message Board

Geoff Collier
Jul 1, 2025

Welcome to Basecamp!
At Enormicom, we like to work with our clients directly and we do that in Basecamp. We're so happy you're here! :)

Invoices
Hi! I'll hope you're doing well! Before we div...

Pulling the launch in by two weeks
Quick but important update on timing. After...

Please Welcome Kurt!
Hey team! Exciting news — we've got a new ...

Product assets
Hey Tim! Hope you're having a smooth and ...

Docs & Files

Proofs
3 items

Client Uploads
3 items

Budget Breakdown
Chad Neidt • Sep 9, 2025 • Google Doc

Client Logos
Chad Neidt • Sep 10, 2025 • Dropbox file

Old Designs
Chad Neidt • Sep 9, 2025 • Basecamp file

Project Tasks

Build & Launch

- Build homepage and global navigation
- QA across browsers and mobile breakpoints
- Deploy to production and hand off credentials ...

Discovery & Design

- Gather brand assets, photos, and copy from cli...

Chat

Kurt Holloway 3:18pm
lol fair, concrete patterns we can steal without 30 designers: cleaner...

Loah Bernstein 3:18pm
i wonder if we should do a small audit — pick 3-4 patterns from these...

Sofia Castillo Rivera 3:18pm
yep pls, I can put together a quick figma w/ the patterns side...

Kurt Holloway 3:18pm
Friday works for me, will block my morning

Loah Bernstein 3:18pm
same, and I'll round up the current pages we want to audit so we're w...

Schedule

May 2026

SUN	MON	TUE	WED	THU	FRI	SAT
26	27	28	29	30	1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30
31	1	2	3	4	5	6

May 25 ✓ Draft sitemap and information arc...
Mon, May 25

May 25 ✓ Home page
Mon, May 25

Workflow

BRIEF (5)

(1)	(2)	(3)	(4)	(5)
REQUIREMENTS	ANALYSIS	DEFINITION	PROTOTYPING	DEVELOPMENT



External links

Design Playground
<https://forms.com/boards/41thLxQRs...>

Client Files
<https://drive.google.com/drive/u/0/fo...>

Weekly Meeting Link
<https://basecamp.zoom.us/j/82050254...>

ABOUT DAVID HEINEMEIER HANSSON

Made [Basecamp](#) and [HEY](#) for the underdogs as co-owner and CTO of [37signals](#). Created [Ruby on Rails](#), [Hotwire](#), [Kamal](#), [Omarchy](#). Wrote [REWORK](#), [It Doesn't Have to Be Crazy at Work](#), and [REMOTE](#). Won at Le Mans as a racing driver. Invested in [Danish startups](#).

Sent to the world with HEY

[Armin Ronacher's Thoughts and Writings](#)[blog](#) [archive](#) [projects](#) [travel](#) [talks](#) [about](#)

Clanker: A Word For The Machine

written on May 26, 2026

[In my last post](#) I used the word “clanker” as an alternative to “agent” quite consistently and probably excessively. That choice ended up attracting a lot more attention than I expected in the Hacker News comment section of that post and a number of folks had a very strong reaction: to them it sounded like a slur, in one case even something adjacent to the n-word.

That reaction surprised me somewhat, but it also made me realize that I should write down what I mean by the word for future reference.

For me “clanker” is useful because it creates distance from the machine and that is a quality which is important to me. The machine is not a person, not a co-worker, not a friend, not a little spirit in the terminal. It is just a machine, a tool, and nothing more.

Why Not Agent?

I dislike the word “agent” for these LLM based tool loops with a UI attached. In everyday use an agent is someone who acts on behalf of someone else and it has agency and more importantly: responsibility. An agent decides, represents, negotiates,

acts, and can be blamed. In the current AI discourse we increasingly do a lot of anthropomorphizing and the term “agent” is now frequently being used to put blame on an abstract machine. But the machine cannot be responsible, whoever is wielding it is. If it [drops your database](#) it was not at fault, you were.

Agent makes the machine sound like a person with delegated authority and I do not think that is healthy.

What we actually have is a language model attached to a harness, a prompt, some tools, a bit of context, and a boring tool loop. Sometimes the loop is very capable and it surprises us by editing code for a really long time and produce genuinely amazing and even valuable outputs. But the agency is not in the model or harness but in the human and in the organization that deployed it. If my coding tool opens a pull request, I opened that pull request, not the machine. If my machine spams someone’s issue tracker, I spammed someone’s issue tracker with a machine.

In that context I like a word that sounds mechanical as it puts the thing back into the category where it belongs: the category of machinery and tools.

The Machine Has No Feelings

LLMs are not sentient and we should not behave as if they might be, just in case. Elevating these things to anything other than a very fascinating and capable tool is problematic for a whole bunch of reasons.

Today’s machines are dumb (but truly fascinating) token predictors that emits text, calls tools, and are steered by prompts and the training that went into them. They can simulate distress [and affection](#), can simulate being offended, apologize and mimic all kinds of things that humans would do.

A compiler does not feel humiliated when I swear at it, a car does not suffer when I call it a shitbox and a power drill is not oppressed by being handled roughly. An LLM is more complicated than those things, and the interactions you can have with them can be truly uncanny, but a moral status does not appear just because the machine can emit text in the first person.

I keep receiving strange emails from people because, for lack of a better phrase, I am in the weights. I have been writing public code and public text for long enough that models know my name, my projects, and some of the concepts around them. Every so often someone writes to me with the peculiar confidence that comes from a long conversation with a model that has validated and amplified an idea. Sometimes the model seems to have told them that I am relevant for their problem and a source of help. For historical reasons LLMs used to write a lot of Flask code, and every once in a while someone interacts with an LLM long enough about their Python and Flask frustrations that the LLM will eventually reveal who created it which then can result in them sending me an email. Increasingly also because people found my work in other ways interesting and are trying to reach out for advice.

I do not want to mock these people but some of those messages are distressing and I do not know how to deal with them. They show signs of what people have started calling [AI psychosis](#).

It's why I want cold and detached language for these systems. I want to use words that remind us that the thing on the other side is not a person.

Racism Is About Humans

The comparison to racism is where I think the discussion goes badly wrong because racism is a human social evil. It is about humans subdividing humans, assigning lesser worth to some of them, and building rules around those subdivisions that can leave lasting damage for generations. Racial slurs are wrong because they are a tool for dehumanizing humans.

On the other hand a machine is not human, a model is not a race and the GPU cluster that is powering them is not being oppressed. A coding assistant does not need dignity, emancipation, or civil rights. That's also why I find the discussion about [model welfare](#) to be actively harmful. I'm sure you can find ways to measure the "trauma" of models or their feelings but I greatly dislike this theater. It risks elevating models to a position they should not occupy. Models are machines and they are not

enslaved in the moral sense in which humans were enslaved, because there isn't anyone there to be deprived of freedom.

We should be careful about using the language of human oppression in relations to our interactions with machines to not devalue actual humans. If we start treating insults toward a model as morally adjacent to racism, we blur a line that shouldn't be blurred.

AI Is Unpopular

If you take a step away from the communities that are happily embracing AI in different ways, there are even more that are viciously against this technology.

There are humans that feel or are harmed by AI systems: people whose work is copied, workers who label data under questionable conditions, people whose neighborhoods receive the data centers and increased utility bills, Open Source maintainers buried under generated slop, and now also people who spiral because a chatbot keeps validating their delusions. Those harmed or affected deserve that type of attention, not the model.

While I am a true believer in the power and utility of this technology, I increasingly think that calling the non-adopters "misguided" or "afraid" won't do it. It's quite likely that this technology comes with risks and we better remember that all of this is supposed to be in service of humans, and not to replace them.

The Rise Of The Machine

The oddest interaction on the use of "clanker" so far has been people asking me if I were to regret at a point in the future calling the machines "the c-word".

I find that questioning revealing because it already grants the machine the status I am really trying not to grant it. It imagines a future "machine people" reading the discourse and sessions, discovering that we used an ugly word for their ancestors, and then judging us by the standards of human oppression.

Could there be future systems that deserve moral consideration? Maybe. I do not know. If we ever build or encounter something that will have those qualities with memories and lasting interests, the capacity to suffer and feel, and a social existence of its own, and the ability to have agency and carry responsibilities, then we should draw a different line and use different language. But that hypothetical future does not extend backwards to the present day and make the current machines people. We can call an electric door an electric door even if one day someone builds some that have emotions and exhale with pleasure when opening and closing.

Whatever the future may bring, let's not pretend that current LLMs are a protected class or on a path towards it. The right response is to look at the evidence, draw the boundary where it belongs, and change our behavior there. We should not even remotely entertain extending empathy to an object that can generate an "ouch."

And if one's worry is less moral and more about revenge, then I find that even less persuasive. A future machine that is so petty or authoritarian that it wants to punish humans because in 2026 they used an unflattering word for non-sentient tools, our vocabulary was really not the problem.

The Word Is Getting Polluted

There is however a part of this that I cannot ignore. I use "clanker" to create distance from the machine, but other people are using the same word very differently. Some online jokes and skits around "clankers" do not merely say "this robot is annoying" as they deliberately pull in the imagery of slavery, segregation, civil-rights-era racism, and anti-Black tropes.

This is problematic as in those contexts the clanker is not just a machine any more and instead becomes a prop for replaying human racism behind a science-fiction mask. That is horrible and I want no part in that.

I think it will be interesting to see where the meanings of these words end up a few years from now. We're very much in the middle of society re-arranging around the changes that LLMs are causing. If a term becomes primarily associated with people

using robots as stand-ins for actually oppressed humans, then using that term becomes impossible to defend.

The reason I liked the word is precisely the opposite of that use. I want language that prevents anthropomorphizing. I want a word that says: this is a tool, a machine of numbers and matrices.

On Responsibility And Boundaries

If an AI system lies to a user, the system did not commit a moral wrong but the people who designed, deployed, marketed, or negligently used it might have. If a coding assistant generates a security bug, the model is not to blame but the human who accepted and committed the code is.

This is why giving these systems softer, more human language worries me. It makes it easier to move responsibility into some undefined void. “The agent decided.” “The model refused.” Obviously that is convenient and I catch myself plenty of times engaging with the thing in ways that are unhealthy. Even just the “please” in the discourse with the machine calls into question how rational we are in engaging with them.

I do not know what the right word will be. Maybe “clanker” will survive as a useful bit of jargon. Maybe it will become too loaded and we will need another one. Whatever word we use, I want it to preserve a clear division: humans on one side with responsibility, machines on the other as a boring tool.

That boundary is very much not anti-AI. I use these systems every day and I have the pleasure to build tools incorporating them at Earendil and find them astonishingly useful.

A machine can be useful, mimic a human but still just be a machine. That is the work I want “clanker” to do. It is not there to make a future “machine person” small if such a person ever were to exist, and it is not an excuse to launder racism through shitty robot jokes.

If the word stops doing that work, I will find another one because the word isn't what matters as much as the boundary which is important to me.

This entry was tagged [ai](#) and [thoughts](#)
[copy as](#) / [view](#) markdown

© Copyright 2026 by Armin Ronacher.

Content licensed under the Creative Commons Attribution-NonCommercial 4.0 International
[License](#).

Contact me via [mail](#), [bluesky](#), [x](#), or [github](#).

You can [sponsor me on github](#).

More info: [imprint](#) & [AI transparency](#). Subscribe via [atom](#) / [RSS](#).

Color scheme: auto, light, dark.

Tailscale Terraform Provider v0.29.2

v0.29.2 of the [Tailscale Terraform Provider](#) has been released with the following changes:

- A regression in [tailscale_tailnet_key](#) has been resolved where `recreate_if_invalid` was not being checked before recreating the resource when the key is not found.

Home / Developer skills / GitHub

GitHub for Beginners: Getting started with Git and GitHub in VS Code

Discover how to use VS Code to interact with GitHub and maintain your projects.



Kedasha Kerr · @ladykerr

May 25, 2026

 9 minutes

Share:   

/ Blog

This time, we're going to take a look at VS Code, a free popular source code editor provided by Microsoft. It has a fair amount of functionality built in that integrates with GitHub, which is what we'll be taking a look at today. Using GitHub in VS Code reduces context switching, streamlines your workflow, and boosts your productivity. By the end of this post, you'll understand how to use VS Code to initialize a repository, switch branches, as well as stage, commit, and push your changes. And the best part is, you'll be able to do all this without leaving the editor.

Note that if you want to follow along with this blog post (or the video), you will need to install both [Git](#) and [VS Code](#). If you need a refresher on how to install Git, you can check out one of our [earlier GitHub for Beginners episodes](#).

As always, if you prefer to watch the video or want to reference it, we have all of our [GitHub for Beginners episodes available on YouTube](#).

First some basics

You probably already know that GitHub is a resource that hosts only copies of your code in repositories. So what is Git? Git is the program for managing that source code, and it can be used in multiple different ways (e.g., from the command line, through VS Code, etc.). Visual Studio Code, often abbreviated as VS Code, leverages Git to enable you to manage your code in GitHub.

Initializing a folder

The first step to using Git with VS Code is initializing a folder to reflect your repository on GitHub.

1. Open VS Code.
2. Select the top icon (the **Explorer** icon) in the left-hand column. It looks like two files on top of each other.

/ Blog

5. After opening your code, select the **Source Control** icon. By default, it's the third icon from the top.
6. Click the **Initialize Repository** button.

At this point, a few things will change in your UI. First, you can see the branch name in the bottom bar on the left-hand side. The default is `main`. You can rename your branch by using the Command Palette.

1. To open the Command Palette, press `Shift-Command-P` on Mac or `Ctrl-Shift-P` on PC.
2. In the Command Palette, start typing "rename" and select the Git: Rename Branch command.
3. In the box, provide the new name of the branch and press `Enter`.

At this point, you can see that the name of the branch in the bottom-left corner has been updated to the new name. You can rename it back to `main` by following the same steps.

Another change you'll see after initializing your repository is that each of the files in the Source Control Panel have a "U" next to them. "U" stands for untracked, meaning that these files are new or changed, but have not been added to the repository. To track a file, you just need to click the plus sign adjacent to the file name. If you want to track all of the files, you can click the top plus next to the word **CHANGES**.

When you do this, the file(s) that you select will be staged, and the letter next to them will change to "A". This means the file is staged, but not yet uploaded to GitHub. In order to upload the changes, you'll need to commit the files.

1. Enter a message in the text box at the top of the Source Control window describing the commit. Alternatively, you could click the **Copilot** icon in the text box to have Copilot generate a commit message for you.
2. Select the **Commit** button underneath the text box to commit your changes.

/ Blog

Creating and changing branches

Right now, you're likely on the `main` branch. Remember that you can check the branch by looking in the bottom-left corner of your window. If this were a major app and you were adding new code or features, you'd want to create a new branch and use that for your work.

1. Open the Command Palette by pressing `Shift-Command-P` on Mac or `Ctrl-Shift-P` on PC.
2. Enter "create branch" in the text box.
3. Select **Git: Create Branch...** from the list of options.
4. Enter the name of the new branch in the box. For example, "new-features".
5. Press `Enter`.

Once you do this, VS Code will create the new branch and automatically transfer you to this branch. You can verify this by looking at the branch name in the bottom-left corner.

Tracking changes you make

Now that you're in your working branch, go ahead and enter a line of code in a file. When you do this, you'll notice that a thin green line appears on the right side of your editor next to the code you added. This section of the editor is called the gutter, and this green line reflects a new line of code that you added.

Move to a different line and make some changes in the line of code that already exists. When you do this, you'll see a blue line with a pattern across it in the gutter. This line indicates that you've made changes to an existing line of code.

Finally, move to an unchanged line in the file and delete it. Notice that the gutter adds a red arrow. This indicates that a line of code was removed from the file.

When you modify this file, you can see that the file appeared in your Source Control window under the **CHANGES** header. If you hover over the filename, you'll see several buttons

/ Blog

all the changes, and stage all the changes.

Viewing diffs

Sometimes you want to see the changes that you made in a file. VS Code lets you do this by performing side-by-side diffs without needing another tool. To see the changes on a file, click on the name of the file you want to see in the Source Control window. From here you can see the changes in the file and compare the differences.

Depending on your preferences, you can also view your diffs in what is called an inline view.

1. Click the three dots (...) in the top-right of the diff view.
2. Select **Inline View** from the drop-down menu.

This lets you see all of the changes in a single window without splitting it up over two separate views. From this view, you can even make edits inside of the diff view.

Once you've made any changes you want to make to the file, it's time to upload them to GitHub. Following the steps we went over before, go ahead and stage your changes, and then commit your staged changes. Once you finish these steps, you shouldn't have any files displayed in the Source Control window.

Merging branches

Note that changes you've uploaded will still be in your working branch. If you navigate back to the `main` branch, you'll see the original code before the changes you made.

1. Click the branch name in the bottom-left of the window.
2. Branches names appear in the drop-down at the top of the program. Select the `main` branch.

In order to get these changes into your `main` branch, you'll need to merge branches.

/ Blog

- The box at the top will prompt you with branches that you want to merge from. Select the branch with the changes you want to merge into `main`.

Congratulations! Now your `main` branch has incorporated the changes!

Publishing to GitHub

Let's say you want to take your project and publish it up to GitHub. To do so, click the **Publish Branch** button in the Source Control window. VS Code will prompt you with whether you want to publish it as a private or as a public repository. Select the option you want, and then VS Code handles the rest.

Once VS Code finishes the publishing process, it will notify you that the project has been published to a repository on GitHub. You can click the **Open on GitHub** button to visit your project on GitHub and see it online.

Cloning a repository

Now let's say you want to clone a repository so that you can work on it on your machine. This creates a local copy that you can use and sync the changes between the two locations. There are multiple ways that you can clone a repository, and this is an easy way to do it in VS Code.

- Navigate to the home page of the repository you want to clone.
- Click the green `<>` **Code** button at the top of the repository file list.
- In the drop-down menu, select the **Copy URL to clipboard** button next to the box containing the repository's URL.
- Open VS Code.
- Open the Command Palette by pressing `Shift-Command-P` on Mac or `Ctrl-Shift-P` on PC.
- Type in "clone".

/ Blog

9. Select **Clone from URL** with the URL you pasted after it.
10. A pop-up window will ask you for a location. Choose the folder where you want to store the project files.
11. Click the **Select as Repository Destination** button.
12. A pop-up menu will appear asking if you want to open the repository. Select **Open**.

Congratulations! You've just cloned the repository to your machine and can start to work on it in your local environment!

Model Context Protocol

Did you know that you don't have to do everything manually in VS Code? You could leverage Model Context Protocol (MCP) to safely let AI tools access external tools and data. The first step is to add the GitHub MCP extension.

1. In the left navigation bar, click the **Extensions** icon.
2. In the search box, enter "@mcp github".
3. Select the GitHub extension from the list.
4. In the description for the extension, click **Install**.
5. A pop-up appears, asking you to allow the MCP server to authenticate. Select **Allow**.
6. Select your GitHub username from the list.

At this point, the GitHub MCP server is installed. You can verify this by looking at the bottom of the Extensions view and seeing the section for installed MCP servers. **With the MCP server installed, you can use Copilot chat to create some code for you, and it will do so by leveraging external tools where necessary.**

1. Open the chat window by selecting the **Chat** icon next to the Command Palette window.
2. Enter a prompt asking Copilot to add some features to your project.

/ Blog

Next steps

And that's it! We covered some of the most common ways developers use VS Code to interact with Git. We've gone over everything from creating repositories, to publishing on GitHub, and even threw in a little bit of using AI at the end. There are more advanced tips, but these elements are what you'll use most frequently.

Happy coding!

Tags:

Git

Github

GitHub for beginners

VS Code

Written by



Kedasha Kerr

[@ladykerr](#)

Kedasha is a Developer Advocate at GitHub where she enjoys sharing the lessons she's learned with the wider developer community. She finds joy in helping others learn about the tech industry and loves sharing her experience as a software developer. Find her online [@itsthatladydev](#).

More on Git

Highlights from Git 2.54

The open source Git project just released Git 2.54. Here is GitHub's look at some of the most interesting features and changes introduced since last time.

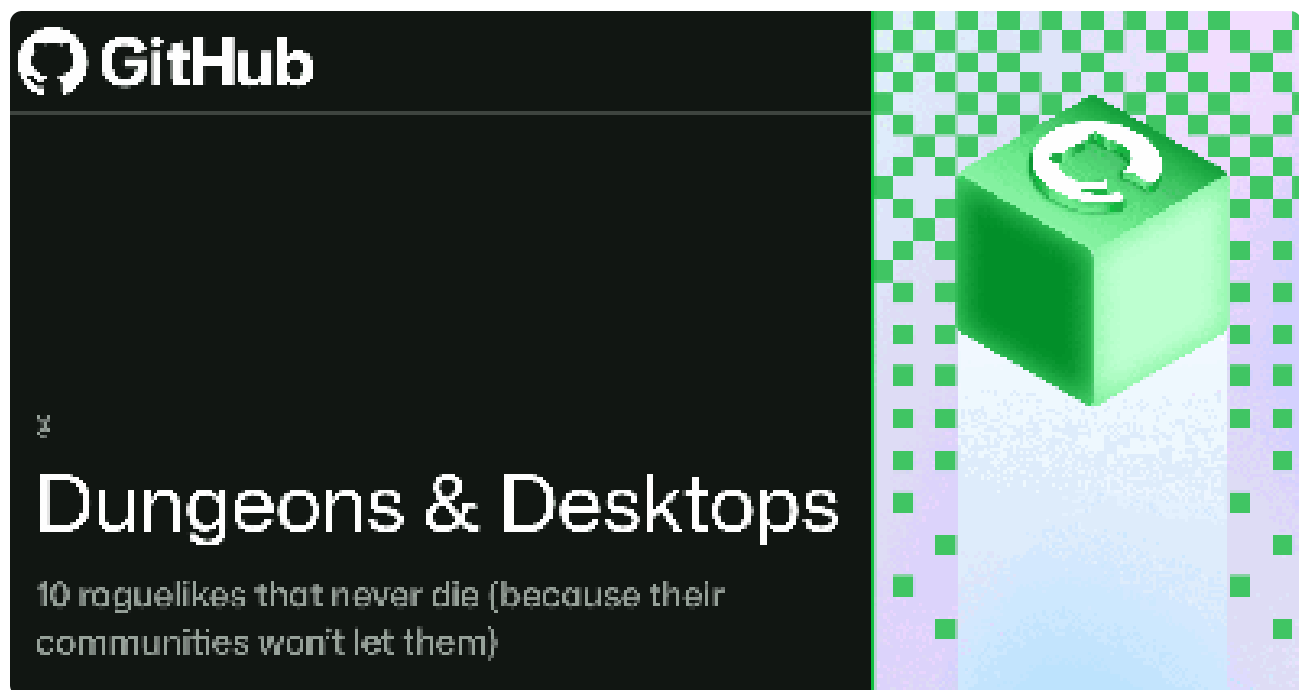
/ Blog

Highlights from Git 2.52

The open source Git project just released Git 2.52. Here is GitHub's look at some of the most interesting features and changes introduced since last time.

Taylor Blau

Related posts



🔗

Dungeons & Desktops

10 roguelikes that never die (because their communities won't let them)

Application development

Dungeons & Desktops: 10 roguelikes that never die (because their communities won't let them)

Roguelikes don't die. They fork, mutate, get argued over, rewritten, abandoned, and revived again. Sometimes all at once.

/ **Blog**



Developer skills

GitHub for Beginners: Getting started with OSS contributions

Learn how to find opportunities to contribute to the open source community.

Kedasha Kerr



/ Blog

Kedasha Kerr

Explore more from GitHub



Docs

Everything you need to master GitHub, all in one place.

Go to Docs [↗](#)



GitHub

Build what's next on GitHub, the place for anyone from anywhere to build anything.

Start building [↗](#)



/ Blog

Learn more [↗](#)



The GitHub Podcast

Catch up on the GitHub podcast, a show dedicated to the topics, trends, stories and culture in and around the open source developer community on GitHub.

Listen now [↗](#)

We do newsletters, too

Discover tips, technical guides, and best practices in our biweekly newsletter just for devs.

Your email address

* Your email address

Yes please, I'd like GitHub and affiliates to use my information for personalized communications, targeted advertising and campaign effectiveness. See the [GitHub Privacy Statement](#) for more details.

/ Blog

GitHub

Product

[Features](#)

[Security](#)

[Enterprise](#)

[Customer Stories](#)

[Pricing](#)

[Resources](#)

Platform

[Developer API](#)

[Partners](#)

[Atom](#)

[Electron](#)

[GitHub Desktop](#)

Support

[Docs](#)

[Community Forum](#)

[Training](#)

[Status](#)

[Contact](#)

Company

[About](#)

[Blog](#)

[Careers](#)

[Press](#)

[Shop](#)

© 2026 GitHub, Inc. [Terms](#) [Privacy](#) [Manage Cookies](#)
Do not share my personal information



← [rss-offline](#) · [abrir original](#) · **Barely treading water [rendered]** · Leadership in Tech · 2026-05-24 15:05



Leadership in Tech

Barely treading water

#314 – May 24, 2026

how leaders can quietly fall into overload while still appearing successful

If you are a human, ignore this field

[Barely treading water](#)

11 minutes by Michael Lopp

Michael explores how leaders can quietly fall into overload while still appearing successful. Through a conversation with his chief of staff, Michael realizes he is failing at prioritization, delegation, and setting limits. He points out common warning signs of burnout and offers practical solutions: admitting failure, prioritizing honestly with trusted support, delegating important work, and learning to say no. It highlights that effective leadership requires self-awareness, honesty, and difficult decisions.

[Unblocked: The context layer for modern engineering teams](#)

sponsored by Unblocked

Whether it's written by an agent or an engineer, the difference between compilable and mergeable code has always been context. Unblocked turns code, docs, tickets, and conversations into actionable context, so engineers move faster and agents stay on track.

[Fix delivery first](#)

10 minutes by Ant Murphy

Ant explains that many companies fail with OKRs, discovery, and AI because their real problem is poor delivery systems. Long release cycles, technical debt, and weak engineering foundations slow progress and reduce trust. Using the Theory of Constraints, Ant argues companies should first fix delivery bottlenecks before focusing on strategy, discovery, or AI. Strong delivery creates faster feedback, builds trust, and makes innovation and improvement possible.

[On blunt feedback](#)

2 minutes by Shreyas Doshi

Blunt feedback only helps when the giver has good intentions and real expertise in the area. In toxic workplaces, some feedback is designed to attach damaging labels to you through backchannels, not to help you grow. And even well-meaning feedback can miss the mark when the giver is less skilled than the recipient. For high performers, knowing when to ignore or adapt feedback is a rare but valuable skill.

[You taught the company to overload you](#)

4 minutes by Aviv Ben-Yosef

Tech leaders who always say yes create the same problems as those who always say no. Making the cost of every agreement visible is the real solution. Be honest about workload limits, avoid sudden pushback after long silence, and always present a range of options rather than a flat refusal. Boundaries need constant reinforcement, not just a one-time announcement.

[The quiet power of presence](#)

3 minutes by Andi Roberts

Most of us listen only to find a gap where we can speak. Real listening means paying attention to the values and hopes behind someone's words, not just the words themselves. Sitting with silence a little longer than feels comfortable often brings out what matters most. When you truly hear someone without judging or fixing them, you build the trust that holds relationships together.

industry

- [A new generation of ads for the AI era of Search](#)

- [SpaceX files for IPO](#)
- [The last six months in LLMs in five minutes](#)

security

- [GitHub confirms breach of 3,800 repos via malicious VSCode extension](#)
- [CISA Admin Leaked AWS GovCloud Keys on Github](#)
- [Mini Shai-Hulud Strikes Again: 317 npm Packages Compromised](#)

And the most popular article from the last issue was:

- [Recognize your management wins](#)

If you are a human, ignore this field

newsletters

- [Programming Digest](#)
- [React Digest](#)
- [C# Digest](#)

© 2013-2026 [Bonobo Press](#)
[Newsletters](#) · [Privacy](#) · [Advertise](#)

[Armin Ronacher's Thoughts and Writings](#)[blog](#) [archive](#) [projects](#) [travel](#) [talks](#) [about](#)

Building Pi With Pi

written on May 24, 2026

Pi is now part of Earendil, but in the important sense it is still [Mario's](#) project. He has been living with its issue tracker longer than I have, and he has been exposed to the weirdness of the new form of agent traffic in Open Source projects for longer too. This post is mostly a reflection of my own experience after spending more time in the tracker, using Pi to work on Pi, and watching what I have learned about it so far.

Slop Issues

Unsurprisingly, we are using Pi to build Pi. That sounds like a cute dogfooding thing but it really helps understand what we do. An interesting effect of building with agents is that it changes the role of the issue tracker a tiny bit. The issue descriptions are not just messages from a user to a maintainer because we also use them as inputs for prompts in Pi sessions. It is something I might hand to my clanker¹ and say: “understand this, reproduce it, inspect the code, and propose a fix.”

That means the shape of the issue matters in a new way. A bad issue was always annoying, but at least a lot of issues were vague. Now we are also dealing with a class of issues that are 5% human and 95% clanker-generated and largely inaccurate shit. A bad issue that contains a plausible but wrong diagnosis creates extra work.

The most frustrating failure mode right now is that people submit issues that are not in their own voice. They contain an observed problem somewhere, but it has been thrown into a clanker and the clanker reworded it and made a huge mess of it. Typically, it was prompted so badly that the conclusions produced are more often than not inaccurate but always full of confidence. The result is complete guesswork on root causes, fake-minimal repros, suggested implementation strategies, analogies to adjacent but often the wrong code, and long lists of error classes that might or might not matter.

That is worse than no diagnosis.

I don't want to point to specific issues because I really do not want to bad mouth anyone, but it is frustrating. It is also frustrating because when I give that issue to Pi, Pi sees the wrong diagnosis too. It does not treat the issue body as a rumor. It treats it as evidence. It will happily go down the path that the issue already prepared for it, because the prose is confident and the code references look plausible. We use a custom slash command called `/is`, which specifically has this instruction in it:

Do not trust analysis written in the issue. Independently verify behavior and derive your own analysis from the code and execution path.

Unfortunately, it does not fully work, because when humans first throw their issue through the clanker wringer, their clanker expands scope almost immediately. What was once a very narrow and fact based bug observation, turns into a much expanded surface area full of hypotheses. So at least personally, I increasingly want issue reports to be condensed to what the human actually observed:

1. I ran this command.
2. I expected this to happen.
3. This happened instead.
4. Here is the exact error or log.

That is enough. If you used an LLM to understand the problem, great, maybe leave it as a follow-up comment. But the issue and the issue text should be something you own. If you do not know the root cause, say that. I too can operate a clanker, and I

would rather do this myself than use your slop. If your repro is a guess, say that. If the only hard fact is one stack trace, give me the stack trace and stop there.

Slop Begets Slop

That we're seeing issues full of slop is just a result of the present day quality of these machines. Sadly, their failures in creating good issues extend to a lot of code that is generated. Not all of it, but a lot of code. Over and over I keep running into them over-engineering the hell out of issues and implementations.

If you tell them that "this malformed session log crashes the reader," the clanker will often add a tolerant reader. Then it will add a fallback, then maybe a migration, then more debug output, then a test for all of this. None of this is necessarily wrong in isolation, but it can be the wrong move for the system.

At Pi's core is a rather well-designed session log with invariants that must be upheld. The clanker's present-day behavior is to just assume that no such invariants exist, and instead to make the system work with all kinds of malformedness, blowing up the complexity in the process.

Almost always, the correct fix is not to handle the bad state, but to make the bad state impossible. This matters a lot for persisted data such as Pi session logs. They are opened, branched, compacted, exported, shared, and analyzed. The goal here is to never write bad session data. Yet if you just let the clanker roam freely, it will attempt to handle every case of bad data in the session log with a more permissive reader.

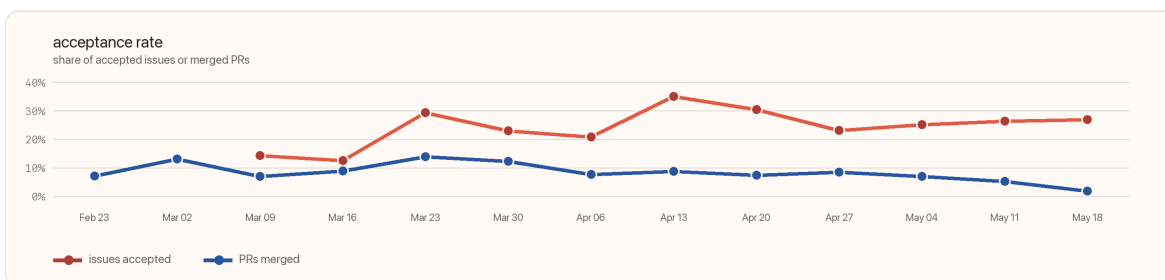
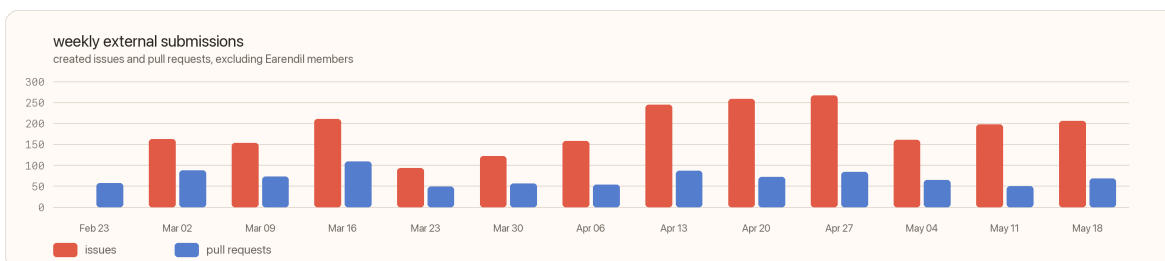
I have complained about this plenty, but working on Pi's code base continues to reinforce the point. This is one of the ways LLM authored code grows so much needless complexity. All these models see a local failure and try to locally defend against it. As maintainers we have to keep pulling the conversation back to the global invariant, which is harder than it should be, and it's laborious.

Volume Is The Problem

Then there is the issue of volume. The tracker is receiving a lot of issues and PRs, and a significant fraction of them are clearly LLM-assisted. Some are good, none are excellent, and most are just bad. The total throughput is a maintenance problem by itself.

As you might know, Pi's issue tracker is automated to close all issues and pull requests from new contributors, and there is a manual process by which we might reopen some of them or approve individuals. So auto-close -> reopen -> close again is an interesting statistic for us to look at.

I pulled the public GitHub tracker data while writing this over the last 90 days. Excluding Earendil members, that leaves 3,145 external issues and pull requests. Of those, 2,504 were auto-closed because they were from non-approved individuals. 17% were re-opened but that somewhat undercounts issues, because some remain closed while we still fix them. If we also count issues referenced by a main-branch commit or merged pull request that number rises to 26%. For pull requests the number is worse: 60 of 714 auto-closed PRs were ultimately merged, or about 8%.



Many of the issues and PRs are complete slop and in some cases the humans did not even realize that they created them. Sources of low-quality spam include OpenClaw instances, as well as some skills that people put into their context that seemingly encourage issue creation.

GitHub clearly is not built to deal with this new form of Open Source, but I'm increasingly feeling the need to put the blame less on GitHub than on all the people

involved who make that experience painful. If your clanker shits on someone else's issue tracker then it's not the fault of GitHub, it's yours alone.

Careful Parallelism

Pi might be built with Pi, but we're quite far off today from where Bun and OpenClaw already are: fully detached, automated software engineering. Maybe we will reach that point, I don't know. Today it does not seem like we know how to pull off a dark factory and we also don't yet have the desire. That said, there is quite a bit of parallelism going on, and it is mostly for reproducing issues.

The small setup we use for this is three tiny pieces in Pi's own committed `.pi` folder. `/is` (for analyze **i**ssue) is a prompt for analyzing GitHub issues: it labels and assigns the issue, reads the full thread and links, then explicitly tells the agent not to trust the analysis in the issue and to derive its own diagnosis from the code. Then an extension adds a `prompt-url-widget` which watches the prompt before the agent starts, recognizes the GitHub issue or PR URL that `/is` (or the PR equivalent) put into the prompt, fetches the title and author with `gh`, renders that in a little UI widget, and renames the session. It also rebuilds that state on session start or session switch, so if we reopen an older investigation the window still tells the developer which issue it belongs to.

In practice this means it's possible to have several Pi windows open, each running `/is` against a different issue, and the UI keeps the investigations visually distinct while the agents do their independent reproduction and code reading. Once the investigations are done, one can work through them sequentially. To finish off everything, `/wr` (**w**rap it up) is the matching wrap-up prompt: it infers the GitHub context from the session, updates the changelog, drafts or posts the final issue comment with a disclaimer, commits only the files changed in that session, adds the appropriate `closes #...` when there is exactly one issue, and pushes from `main`.

```
Next validation before fixing:
```

1. Add the repro as a failing unit test.
2. Add a companion SSE idle test if we want timeout semantics there too.
3. Define exact semantics:
 - `timeoutMs` bounds WS connect wait.
 - `timeoutMs` bounds time to first event.
 - optionally also bounds idle gap between events.
4. For auto, only fallback to SSE if timeout occurs before any provider event. After first event, return an error.

```
That keeps the fix scoped and avoids unsupported claims about Codex.
```

```
TPS 42.0 tok/s. out 862, in 187, cache r/w 209,408/0, total 210,457, 20.5s
```

```
openai-codex can hang on Working... with zero-usage aborted turns
```

```
@liushuaiiu
```

```
https://github.com/earendil-works/pi/issues/4945
```

```
— super-precise
```

```
~/Development/pi-mono (fix/rpc-backpressure-retry-abort) • Issue: openai-codex ca...
↑444k ↓22k R11M $8.260 (sub) 77.4%/272k (auto) (openai-codex) gpt-5.5 • xhigh
```

Open Source Is About Hard Problems Worth Fixing

You will have noticed this already but Open Source in a post-AI world is under a strange new pressure. We are getting more code, more projects, and more issues. Projects appear with no real users, or a temporary audience of one, and even projects with thousands of stars can have a shelf life of weeks.

For us, Pi's harness layer is worth maintaining carefully because it solves hard coordination problems and creates a platform we and others can build on. We also know that coordination and cooperation lifts us all up. Many times the right answer is not to work around a problem locally, but to make the upstream behavior correct. Mario has been very good at refusing to make Pi paper over every misconfigured gateway, and we're trying to preserve that discipline. When a gateway behaves correctly, everybody benefits.

Sadly that type of thinking is quickly disappearing because these machines make local workarounds cheap, so code accumulates local defenses against every misbehavior. Instead of humans talking to humans about where a fix belongs, one human and one machine work around the problem in isolation.

Keep in mind that AI has not increased the number of people who need software, or the number of maintainers who can review it. It has mostly increased the amount of code and the number of projects competing for attention. Some of that is healthy, but a lot of it fragments effort that should be shared.

We need stronger foundations, not weaker ones. Open Source needs more collaboration, not more isolated work with a machine. Human communication is hard, and it is tempting to avoid it when you can sit alone with your clanker. But isolation is not where Open Source derives its value. The value is in the community and the structure that lets projects outlive their original creators.

1. To me, [clanker](#) is a much preferable term for agent. Agency lies with humans, not with machines. Calling these things agents I still believe is a mistake, but alas.↩

This entry was tagged [ai](#), [open-source](#) and [pi](#)
[copy as](#) / [view](#) markdown

© Copyright 2026 by Armin Ronacher.

Content licensed under the Creative Commons Attribution-NonCommercial 4.0 International
[License](#).

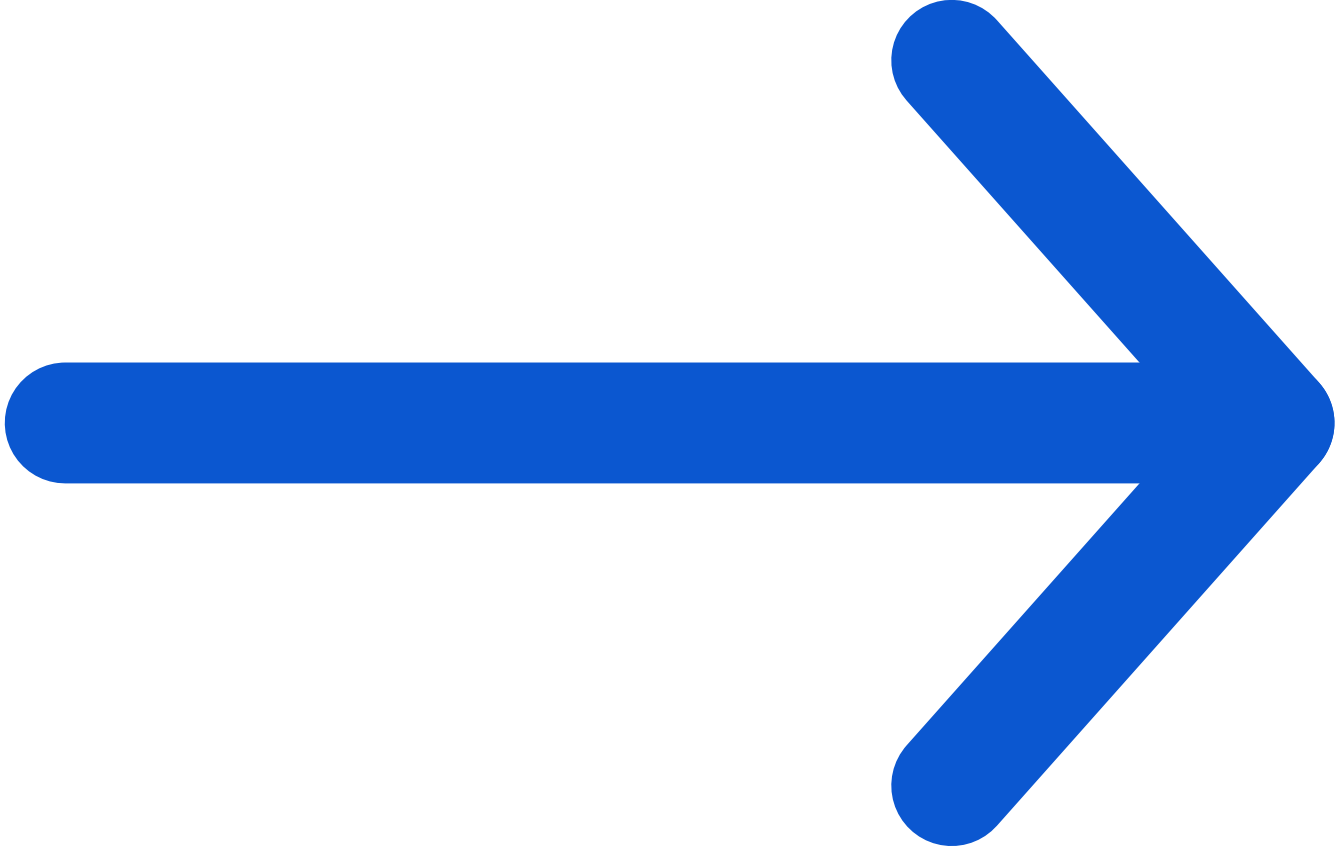
Contact me via [mail](#), [bluesky](#), [x](#), or [github](#).

You can [sponsor me on github](#).

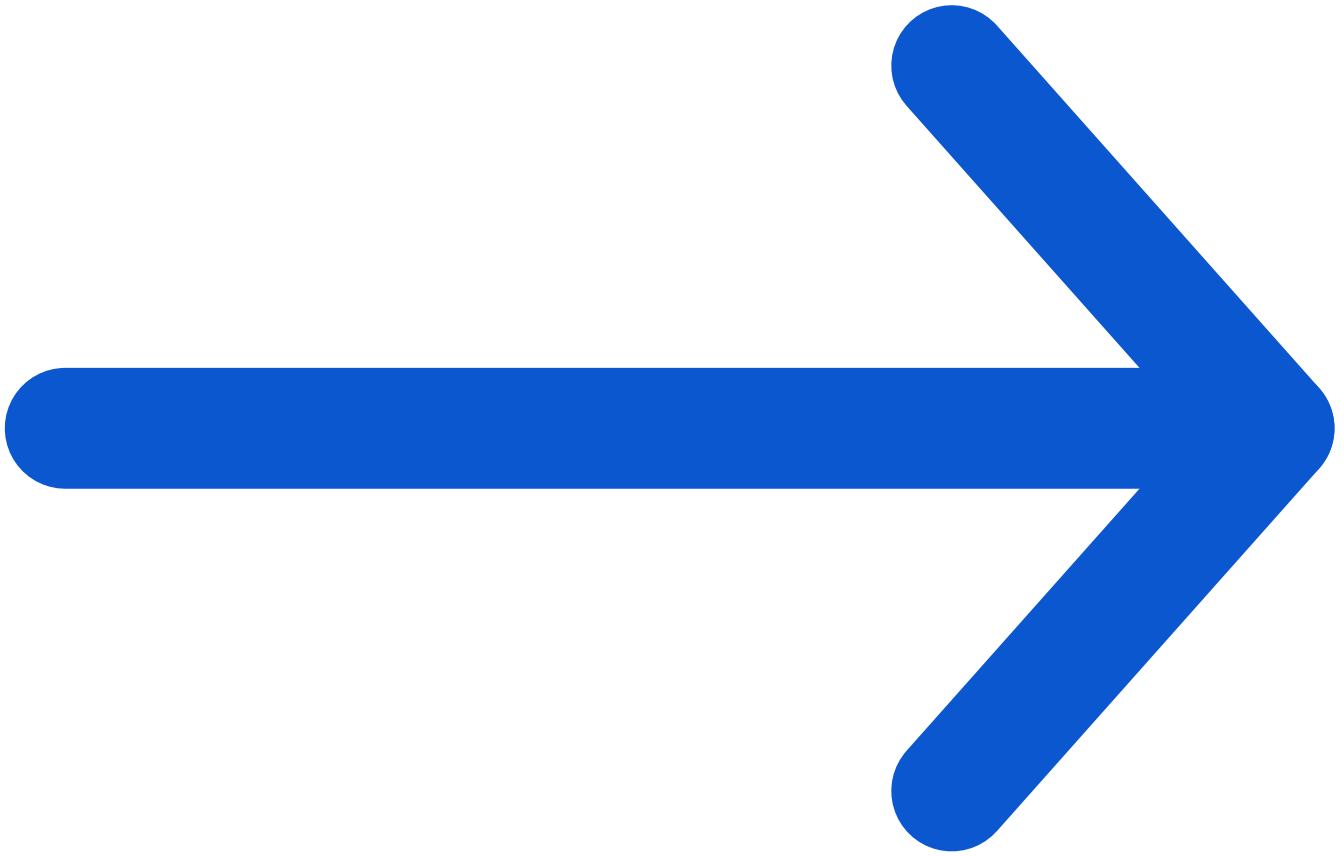
More info: [imprint](#) & [AI transparency](#). Subscribe via [atom](#) / [RSS](#).

Color scheme: auto, light, dark.

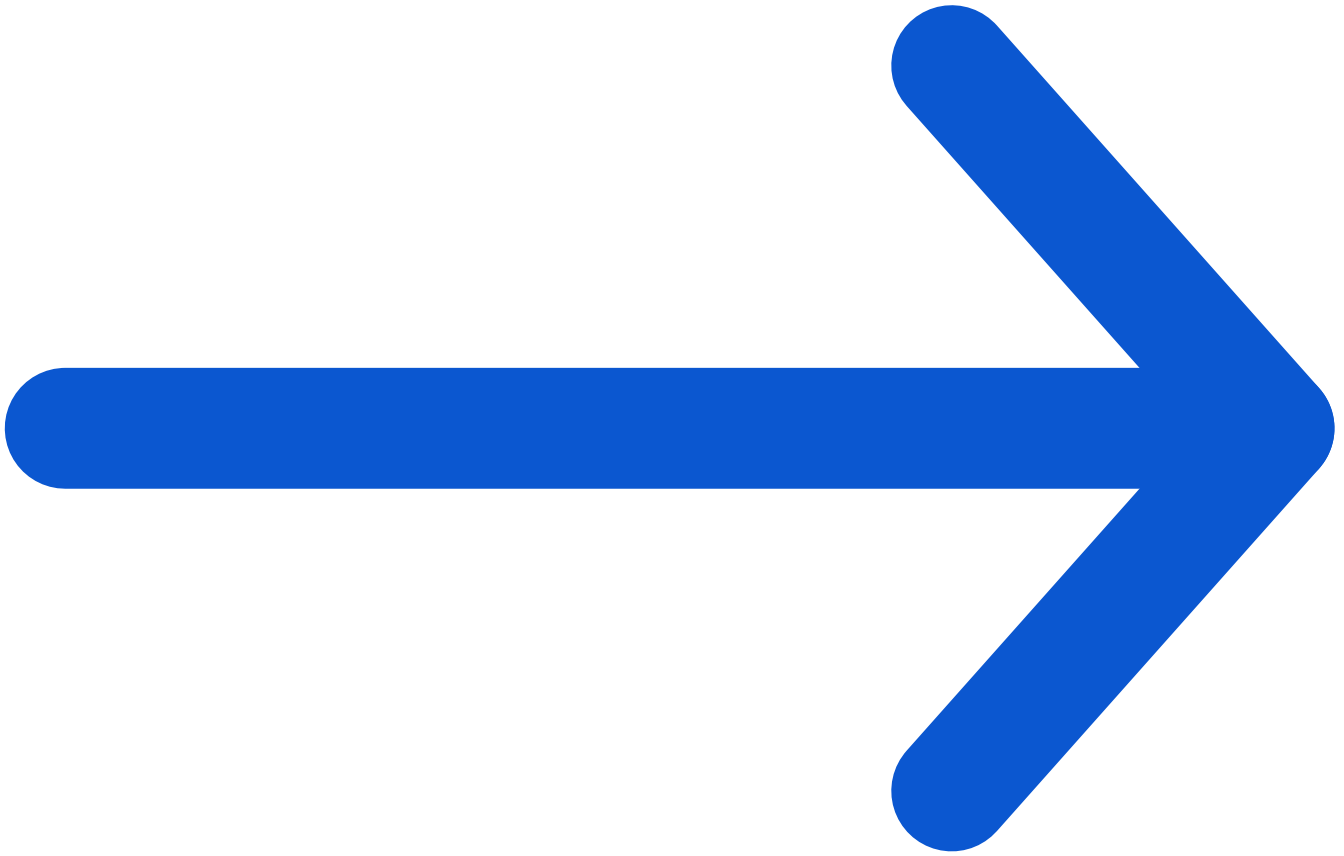
[Inside the Architecture of DigitalOcean's Inference Router](#)



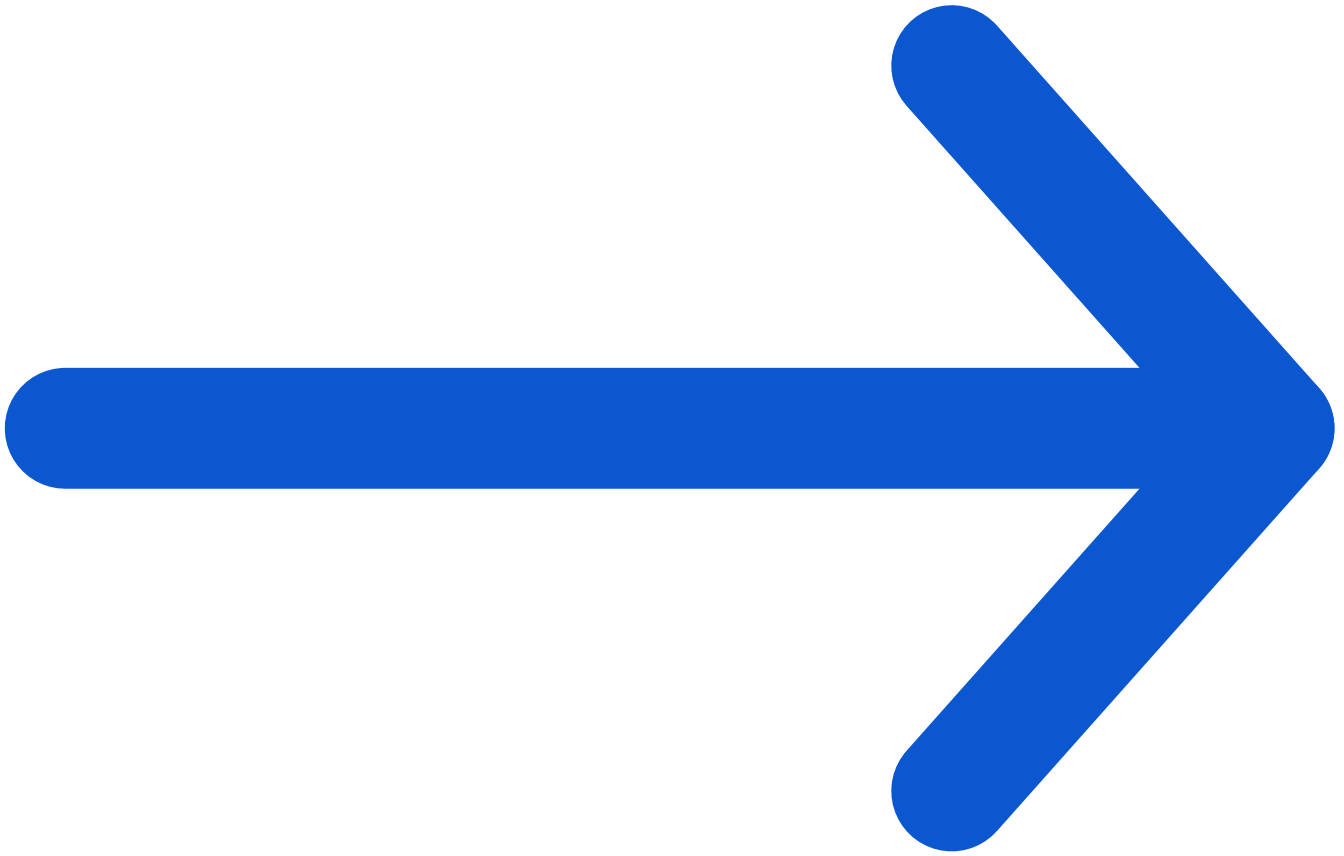
[Introducing DigitalOcean AI-Native Cloud](#)



[Now Available: Kimi K2.6](#)



[Missed the Deploy 2026 Keynote live? Watch it now and catch everything that was announced](#)



- [Blog](#)
- [Docs](#)
- [Careers](#)
- [Get Support](#)
- [Contact Sales](#)



-

Featured AI Products

Compute

Build, deploy, and scale cloud compute resources

Containers and Images

Safely store and manage containers and backups

Managed Databases

Fully managed resources running popular database engines

Management and Dev Tools

Control infrastructure and gather insights

Networking

Secure and control traffic to apps

Security

Help protect your account and resources with these security features

Storage

Store and access any amount of data reliably in the cloud

[Browse all products](#)

- Solutions ▾

AI/ML

CMS

Data and IoT

Developer Tools

Gaming and Media

GPU

Hosting

Security and Networking

Startups and SMBs

Web and App Platforms

[See all solutions](#)

- Developers ▾

Community

Documentation

Developer Tools

Get Involved

Utilities and Help

- Partners ▾

Become a Partner

Marketplace

- Pricing
- Log in
- Sign up
- Log in
- Sign up



Product updates

Request-Based Autoscaling Is Now Generally Available on App Platform



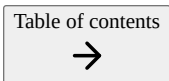
By Bikram Gupta and Greeshma Pillai

- Published: May 22, 2026
- 4 min read

[<- Back to blog home](#)

Table of contents

- Now Available for Shared and Dedicated CPU Instances
- Know Your Baseline Before You Set Thresholds
- How to Configure Request-Based Autoscaling
- Get Started With Request-Based Autoscaling



[Now Available for Shared and Dedicated CPU Instances](#) [Know Your Baseline Before You Set Thresholds](#) [How to Configure Request-Based Autoscaling](#) [Get Started With Request-Based Autoscaling](#)

Traffic doesn't spike on a schedule. A product launch, a viral moment, or a flash sale can send request volume through the roof in seconds, long before your CPU metrics catch up. That gap is where performance suffers.

Today, we're excited to announce that request-based autoscaling on DigitalOcean App Platform is now generally available. Your apps can now automatically scale based on live HTTP traffic signals (requests per second and P95 response latency) so your infrastructure reacts to what's actually happening, not what happened minutes ago.

Now Available for Shared and Dedicated CPU Instances

Until now, autoscaling on App Platform required a **dedicated CPU plan**. That meant a good portion of App Platform users (anyone running on shared CPU instances) had no path to automatic horizontal scaling at all.

That changes today. Request-based autoscaling works on **both shared and dedicated CPU instances**. Whether you're running an early-stage project on a shared plan or a high-throughput production service on dedicated resources, you can now configure autoscaling to match your traffic—no plan upgrade required.

Faster, More Responsive Scaling

CPU-based autoscaling is reactive by nature. CPU is a lagging indicator: your containers have to be visibly struggling before the scaler knows there's a problem, and by then, your users are already waiting.

Request-based autoscaling acts on the signals that actually reflect user experience:

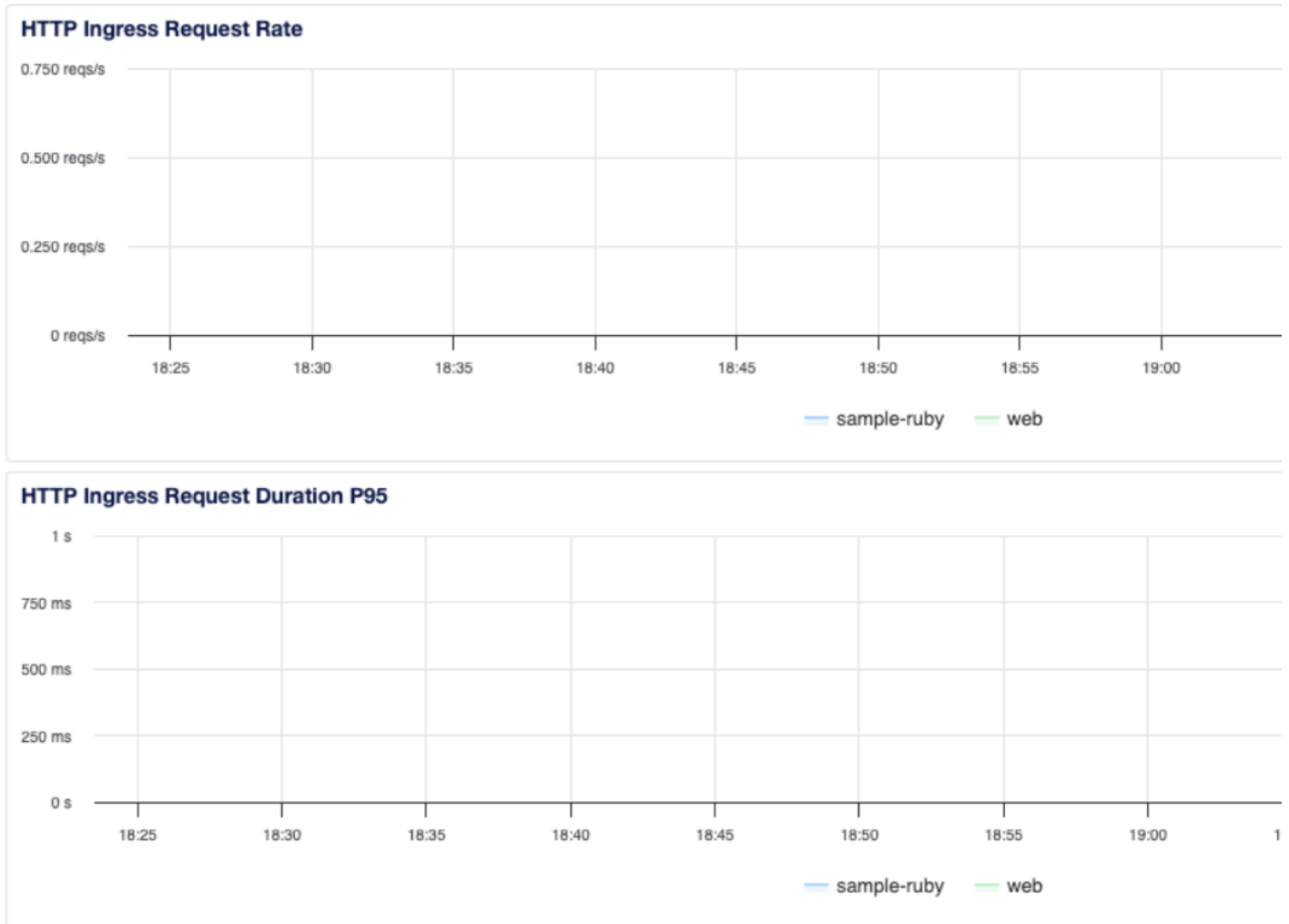
- **Requests per second per instance:** how many requests each container is handling right now
- **P95 request latency:** the response time that 95% of your users are seeing

When traffic rises and either threshold is exceeded, new containers spin up immediately. When load drops and all metrics fall back below their targets, the scaler brings containers back down. You get the capacity headroom you need, faster, and pay only for what you use.

You can also combine request-based and CPU-based metrics on dedicated plans. The autoscaler scales up when *any* configured threshold is crossed, and scales down only when *all* metrics are back in range.

Know Your Baseline Before You Set Thresholds

Configuring good autoscaling thresholds starts with understanding your normal traffic patterns. The **Insights** tab in the App Platform console gives you exactly that.



The Insights tab shows you **HTTP Ingress Request Rate** (requests per second) and **HTTP Ingress Request Duration P95** (your 95th-percentile latency) over time. Use this to understand how your service behaves under normal load before dialing in your autoscaling rules.

How to Configure Request-Based Autoscaling

Using the Control Panel

Go to the [Apps](#) page, select your app, open the **Settings** tab, and select your web service component. In the **Resource Size** section, click **Edit**.

Select the **Shared CPU** or **Dedicated CPU** tab. Under **Scaling**, toggle **Autoscale** on. Set your **Minimum Containers** and **Maximum Containers**, then configure at least one autoscaling rule:

Scaling

Autoscale
 Automatically scale based on requests per second, response time, and/or CPU usage. On

Minimum Containers*

^
v

Maximum Containers*

^
v

Autoscaling rules *

Scaling triggers when any added rule is met. [View insights](#) to help set thresholds based on usage.

Scale on number of requests per second

Requests per second*

rps

Scale on response time and speed (p95)

Scale on CPU usage threshold

- **Scale on number of requests per second** set a target RPS per instance
- **Scale on response time and speed (P95)** set a target P95 latency in milliseconds
- **Scale on CPU usage threshold** available on dedicated CPU plans

Click **Save**. A redeployment kicks off automatically and your app starts autoscaling.

Using the App Spec

Add an autoscaling block to your service component in your app spec. The example below scales between 1 and 10 containers, targeting 100 requests per second per instance and a P95 latency of 500 ms:

None

```
name: my-app
services:
- name: web
  github:
    repo: your-org/your-repo
    branch: main
```

```
autoscaling:  
  min_instance_count: 1  
  max_instance_count: 10  
  metrics:  
    requests_per_second:  
      per_instance: 100  
    request_duration:  
      p95_milliseconds: 500
```

[Copy](#)

Submit your updated spec via `doctl apps update` or the [Apps API](#). You can tune these values at any time—if your service is scaling earlier than you'd like, raise the target; if you're seeing latency before new containers arrive, lower it.

A few things to keep in mind:

- Request-based autoscaling applies to **web service components** that receive external HTTP traffic. Worker and function components are not eligible.
- It cannot be used alongside **Scale to Zero** (Inactivity Sleep) on the same service.
- Scaling decisions are based on a 5-minute rate window, so the autoscaler responds to sustained load rather than momentary spikes.

Get Started With Request-Based Autoscaling

Your traffic doesn't follow a schedule. Your scaling shouldn't either. Request-based autoscaling is available now on every DigitalOcean account. Head to the Insights tab to understand your traffic patterns, then configure your autoscaling rules directly in the console or via the app spec.

→ [Read the documentation to get started](#)

About the author(s)



Bikram Gupta
Author



Greeshma Pillai
Author

Share

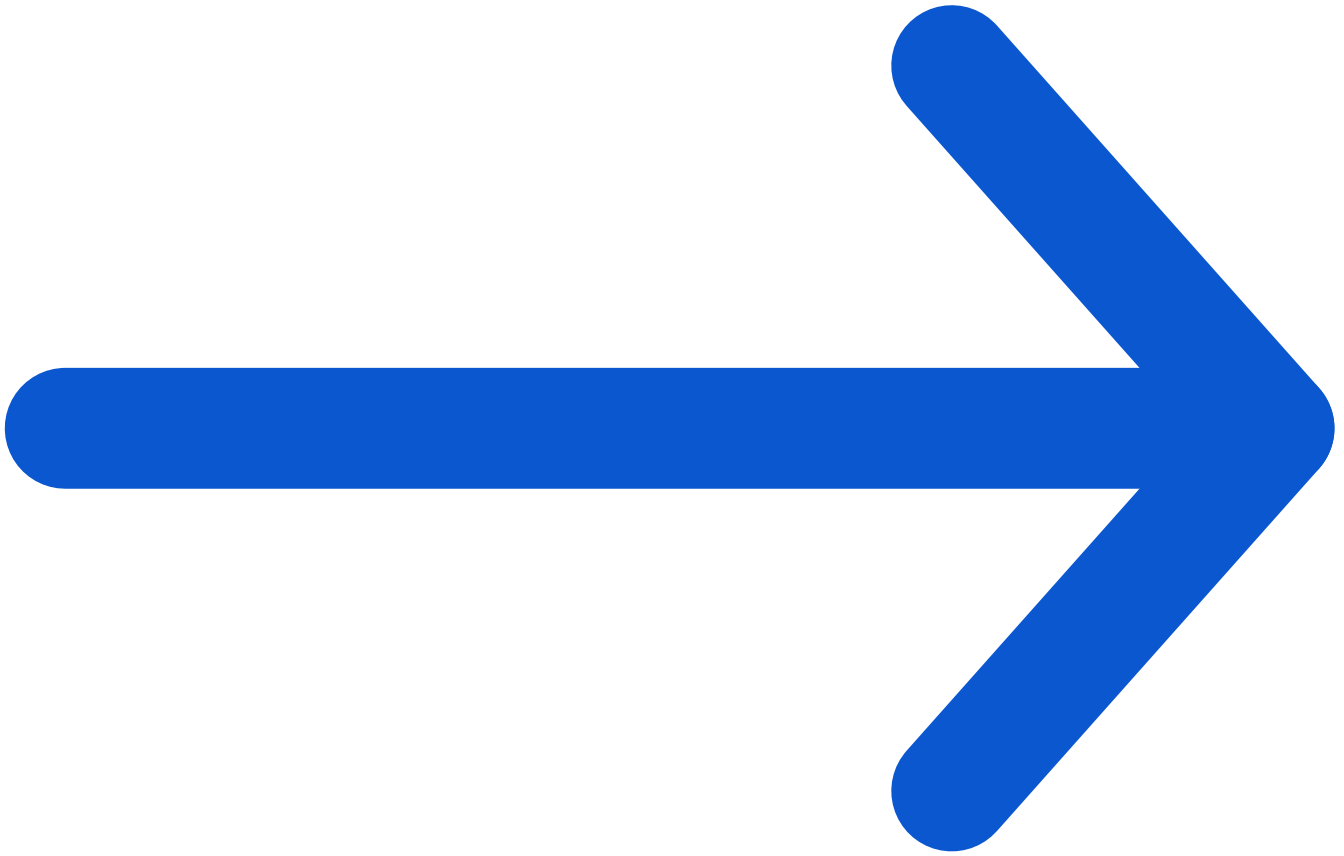


- [Product Updates](#)

Start building today

From GPU-powered inference and Kubernetes to managed databases and storage, get everything you need to build, scale, and deploy intelligent applications.

[Sign up](#)



Related Articles



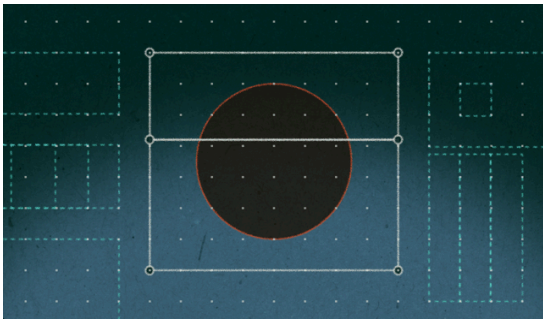
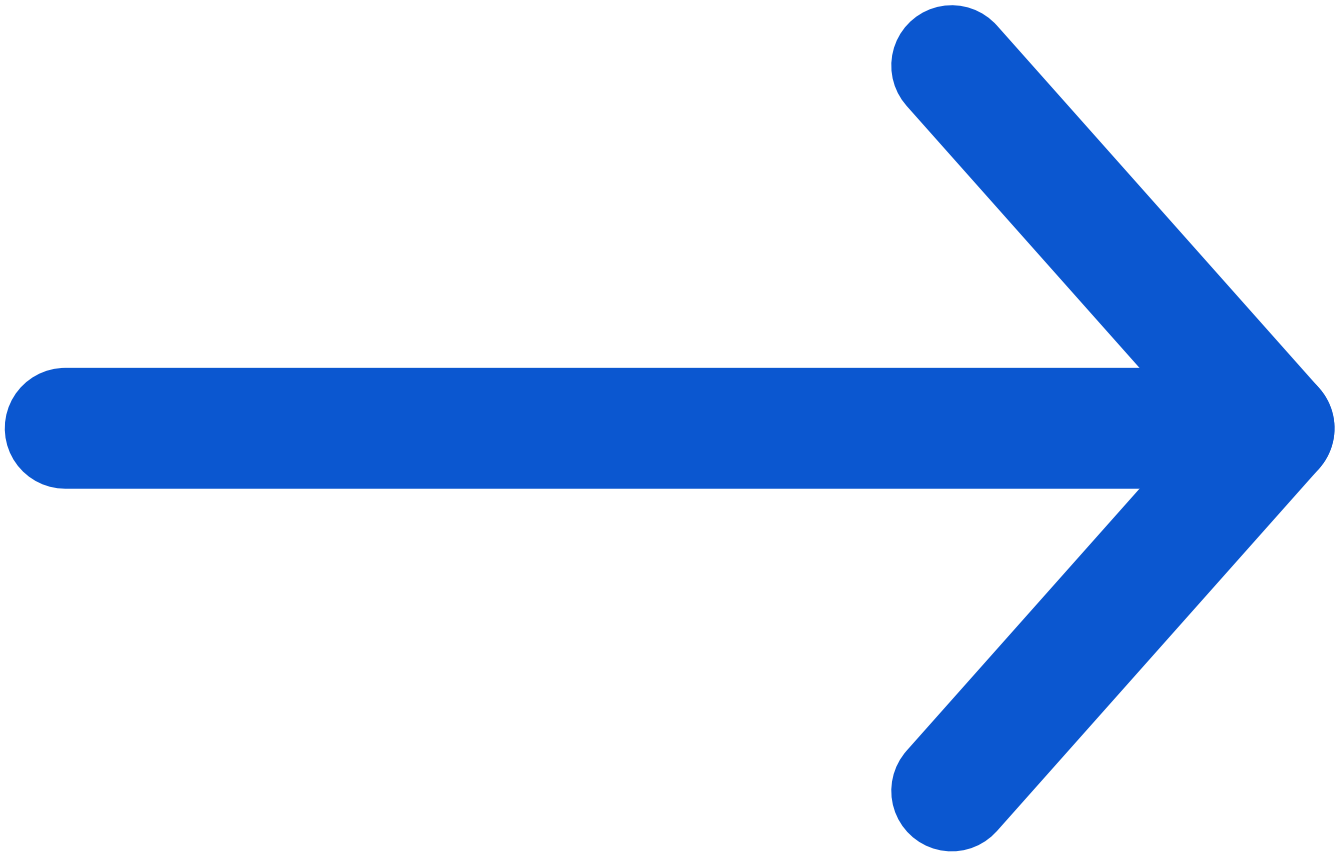
Product updates

Powering the Inference Era: Inside the DigitalOcean AI-Native Cloud

Vinay Kumar, Chief Product & Technology Officer

- May 4, 2026
- 7 min read

[Read more](#)



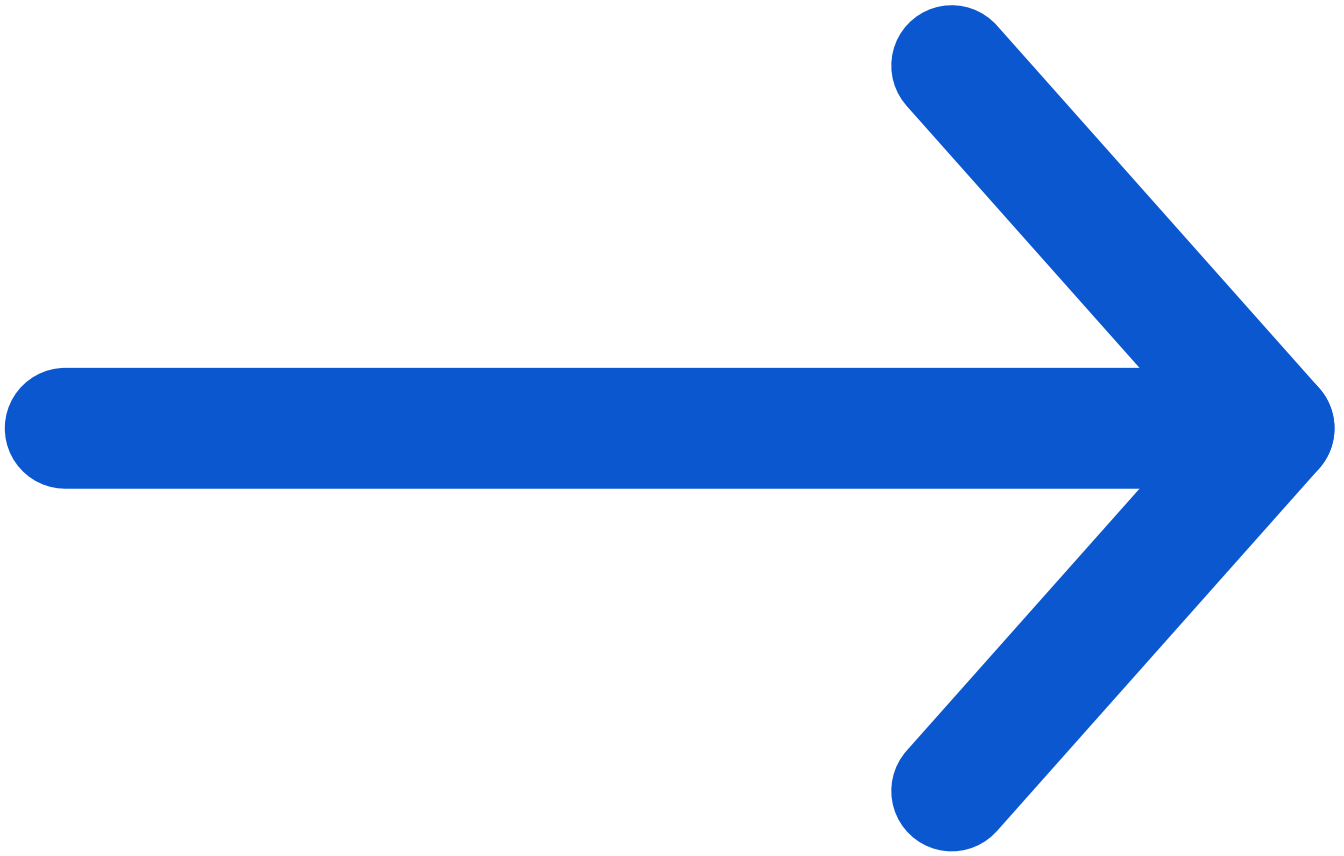
Product updates

Introducing DigitalOcean AI-Native Cloud for Production AI Workloads

[Paddy Srinivasan](#)

- April 28, 2026
- 4 min read

[Read more](#)



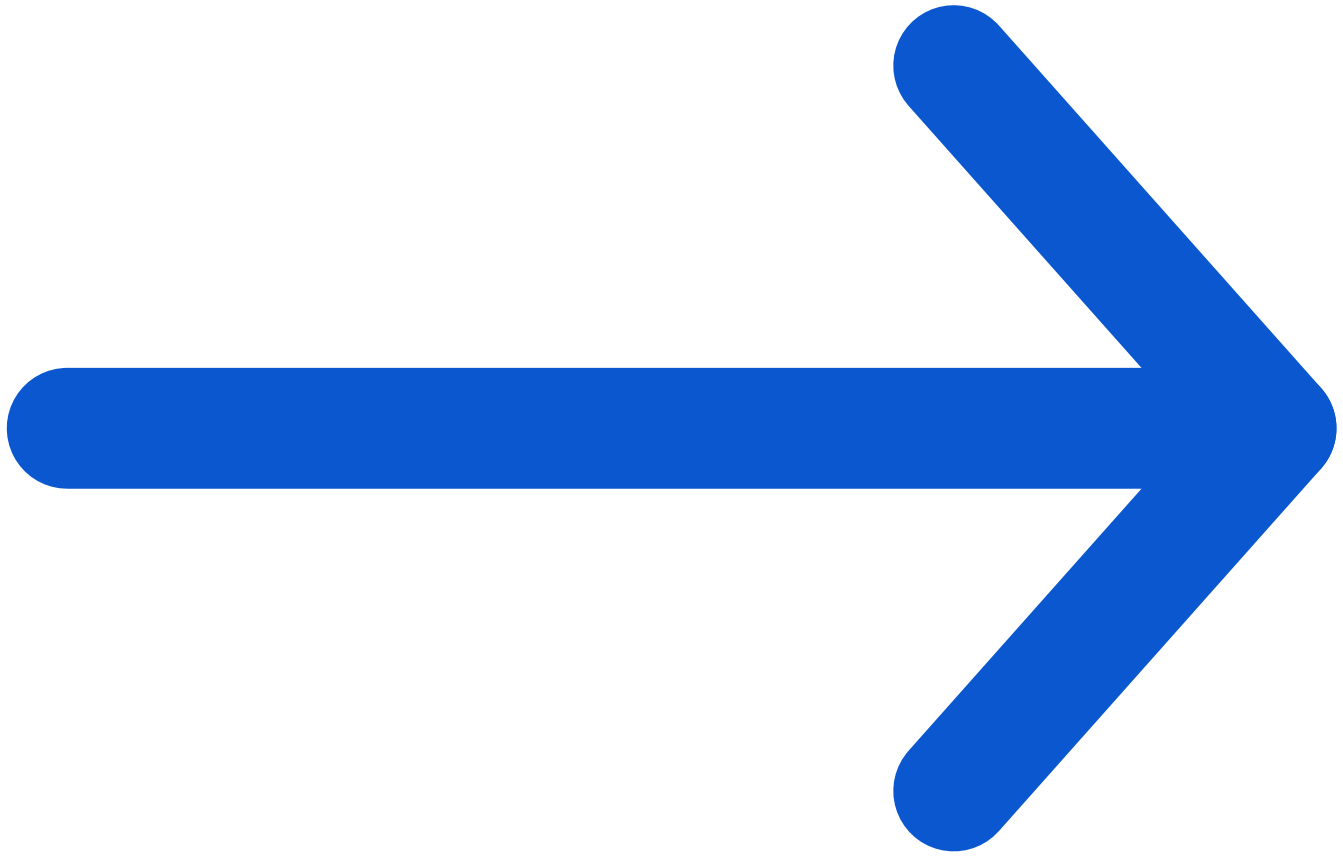
Product updates

The Agentic Era Demands a New Class of Infrastructure: DigitalOcean Acquires Katanemo Labs

Vinay Kumar, DigitalOcean Chief Product & Technology Officer

- April 2, 2026
- 3 min read

[Read more](#)



Company

- [About](#)
- [Leadership](#)
- [Blog](#)
- [Careers](#)
- [Customers](#)
- [Partners](#)
- [Referral Program](#)
- [Affiliate Program](#)
- [Press](#)
- [Legal](#)
- [Privacy Policy](#)
- [Security](#)
- [Investor Relations](#)

Products

- [GPU Droplets](#)
- [Bare Metal GPUs](#)
- [Inference Engine](#)
- [Data & Learning](#)
- [Droplets](#)
- [Kubernetes](#)
- [Functions](#)
- [App Platform](#)
- [Load Balancers](#)
- [Managed Databases](#)
- [Spaces](#)
- [Block Storage](#)
- [Network File Storage](#)
- [API](#)
- [Uptime](#)
- [Cloud Security Posture Management \(CSPM\)](#)
- [Identity and Access Management \(IAM\)](#)

- [Cloudways](#)
- [View all Products](#)

Resources

- [Community Tutorials](#)
- [Community Q&A](#)
- [CSS-Tricks](#)
- [Write for DOnations](#)
- [Currents Research](#)
- [DigitalOcean Startups](#)
- [Wavemakers Program](#)
- [Compass Council](#)
- [Open Source](#)
- [Newsletter Signup](#)
- [Marketplace](#)
- [Pricing](#)
- [Pricing Calculator](#)
- [Documentation](#)
- [Release Notes](#)
- [Code of Conduct](#)
- [Shop Swag](#)

Solutions

- [AI GPU Hosting](#)
- [H100 Cloud GPU](#)
- [AI Training GPU](#)
- [GPU Inference](#)
- [VPS Hosting](#)
- [Website Hosting](#)
- [VPN](#)
- [Docker Hosting](#)
- [Node.js Hosting](#)
- [Web Mobile Apps](#)
- [WordPress Hosting](#)
- [Virtual Machines](#)
- [View all Solutions](#)

Contact

- [Support](#)
- [Sales](#)
- [Report Abuse](#)
- [System Status](#)
- [Share your ideas](#)



Company



Products



Resources



Solutions



Contact



 © 2026 DigitalOcean, LLC. [Sitemap](#).

- 
- 
- 
- 
- 
- 
- 
- 

Dark mode is coming soon.

This site uses cookies and related technologies, as described in our [privacy policy](#), for purposes that may include site operation, analytics, enhanced user experience, or advertising. You may choose to consent to our use of these technologies, or manage your own preferences. Please visit our [cookie policy](#) for more information.

AGREE & PROCEED

DECLINE ALL

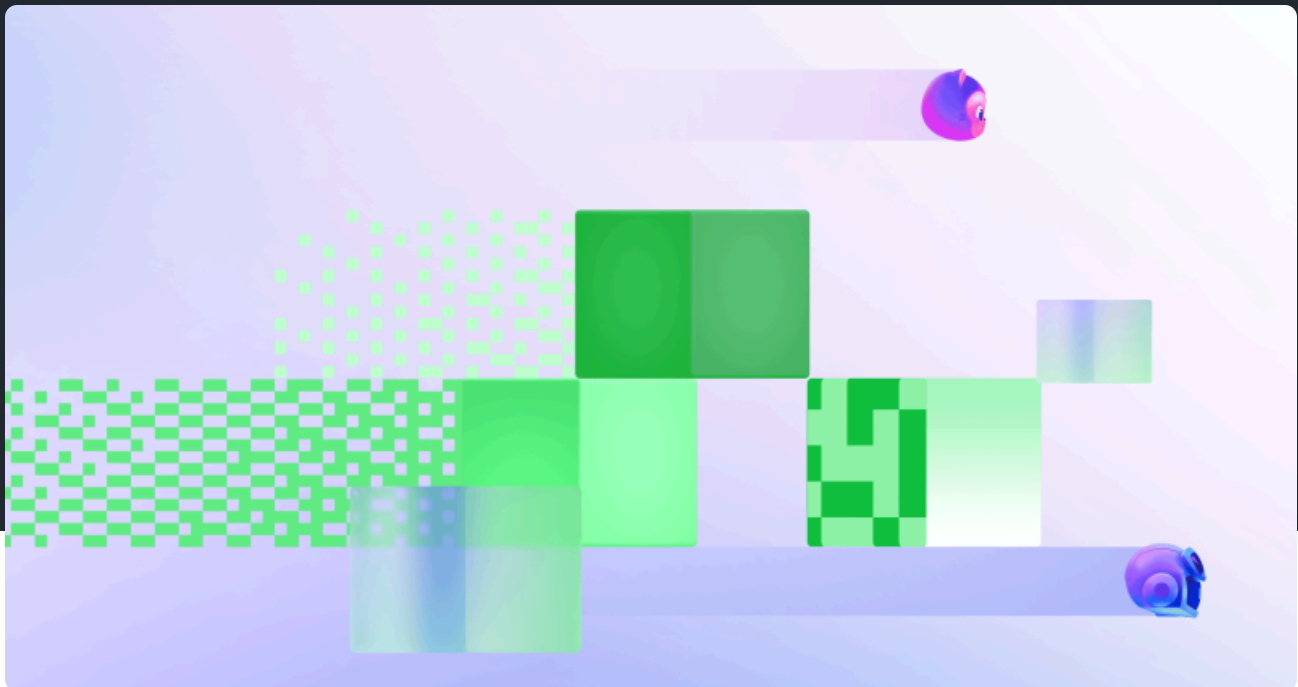
MANAGE CHOICES

[Cookie Preferences](#)

Home / AI & ML / GitHub Copilot

GitHub recognized as a Leader in the Gartner® Magic Quadrant™ for Enterprise AI Coding Agents for the third year in a row

We are committed to empowering every developer by building an open, secure, and AI-powered platform that defines the future of software development.



Mario Rodriguez · @mariorod

May 22, 2026

 4 minutes

Share:



/ Blog

asynchronous AI coding agent workflows will improve software engineering team productivity by 30% to 50%, surpassing the 0% to 20% gains from AI code assistants in 2025.” We believe realizing those gains requires agentic capabilities across every stage of the SDLC—not just code generation, but the review, security, and governance layers where work actually gets stuck. GitHub Copilot covers that full surface. Today, developers don’t just ask Copilot to write a function—they assign an agent to an issue and walk away. The agent handles the rest. The developer returns to review, steer, and approve. That’s the shift: from writing code to orchestrating outcomes. The result isn’t just faster code. It’s faster software, shipped with confidence.

That shift is playing out at enterprise scale. GitHub Copilot now serves 140,000 organizations—nearly triple the number from a year ago—with overall growth topping 100% year over year and most users leveraging multiple AI models. GitHub Copilot CLI is also seeing rapid adoption, with usage nearly doubling month over month. Together, these signals point to a platform being used with growing sophistication. As the market expands and new entrants emerge, we believe the depth of GitHub’s native integrations, security controls, and agentic workflows is unmatched for enterprises governing AI-assisted development at scale. Against that backdrop, we’re pleased to announce that Gartner has positioned GitHub as a Leader in the 2026 Magic Quadrant™ for Enterprise AI Coding Agents **for the third consecutive year.**

/ Blog



As part of the report, Gartner evaluated 12 vendors based on their ability to execute and completeness of vision. GitHub placed as the highest in ability to execute.

According to Gartner, “Leaders in this Magic Quadrant combine strong execution with a clear ability to shape the direction of the market. These vendors stand out for differentiated product experiences, rapid innovation and broad relevance across modern software engineering workflows, including agentic execution that extends beyond in-editor assistance into planning,

/ Blog

While Leaders are not identical in approach, they consistently show that they can translate technical advances into durable market influence and remain central to how organizations adopt agentic software engineering at scale.”

We believe our continued Leader placement in the 2026 Gartner Magic Quadrant for Enterprise AI Coding Agents underscores the strength in our execution, consistently delivering innovations in agentic development, uniquely:

- Honoring developer choice and flexibility by including multiple models from multiple providers and integrating Copilot into multiple surfaces, including code editors, CLIs, IDEs, and GitHub’s web, desktop, and mobile apps.
- Integrating Copilot not just at the beginning, but throughout the software lifecycle, including issues, code reviews, pull requests, and actions.
- Providing engineering teams with governance controls to observe, audit, and secure their use of AI.

And since the evaluation, we’ve kept building, sharpening our strengths and putting more power and capability in developers’ hands across the AI-native software lifecycle.

What’s next

We’re building on the strengths that make Copilot a leading tool for developers and enterprises, expanding its core capabilities and deepening its integrations. GitHub is uniquely positioned for the agentic era, and we’re continuing to invest across the full software lifecycle: deeper agentic workflows across every surface where developers work, expanded model choice with intelligent routing, and Copilot performance improvements grounded in understanding not just how code is generated, but how software on GitHub is actually built.

Let’s build.

[Read the full Gartner report](#) to discover how vendors were evaluated and why GitHub was recognized as a Leader.

/ Blog

publications, and does not advise technology users to select only those vendors with the

highest ratings or other designation. Gartner research publications consist of the opinions of Gartner's research organization and should not be construed as statements of fact.

Gartner disclaims all warranties, expressed or implied, with respect to this research, including any warranties of merchantability or fitness for a particular purpose.

Gartner and Magic Quadrant are trademarks of Gartner, Inc., and/or its affiliates.

Gartner, Magic Quadrant for Enterprise AI Coding Agents, Philip Walsh, Keith Holloway, Matt Brasier, Nitish Tyagi, Neha Agarwal, 20 May 2026.

This graphic was published by Gartner, Inc. as part of a larger research document and should be evaluated in the context of the entire document. The Gartner document is available upon request from GitHub.

Tags :

AI coding tools

Gartner Magic Quadrant

GitHub Copilot

Written by



Mario Rodriguez

[@mariorod](#)

Mario Rodriguez leads the GitHub Product team as Chief Product Officer. His core identity is being a learner and his passion is creating developer tools—so much so that he has spent the last 20 years living that mission in leadership roles across Microsoft and GitHub. Mario most recently oversaw GitHub's AI strategy and the GitHub Copilot product line, launching and growing Copilot across thousands of organizations and millions of users. Mario spends time outside of GitHub with his wife and two daughters. He also co-chairs and founded a charter school in an effort to progress education in rural regions of the United States.

/ Blog

Scaling AI opportunity across the globe: Learnings from GitHub and Andela

Developers connected to Andela share how they're learning AI tools inside real production workflows.

Admas Kanyagia & Ali Condah

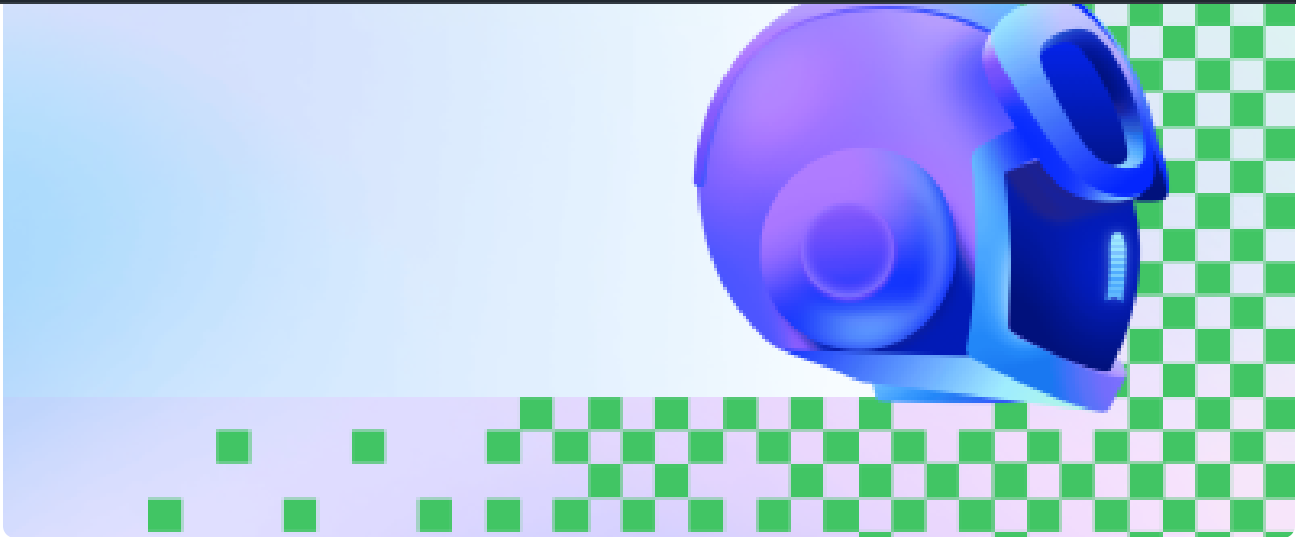
What AI is actually good for, according to developers

AI is designed to help you do what you love most, not replace your expertise. Discover how developer feedback and real-world experience are shaping AI coding tools that keep you in control.

Cassidy Williams

Related posts

/ Blog

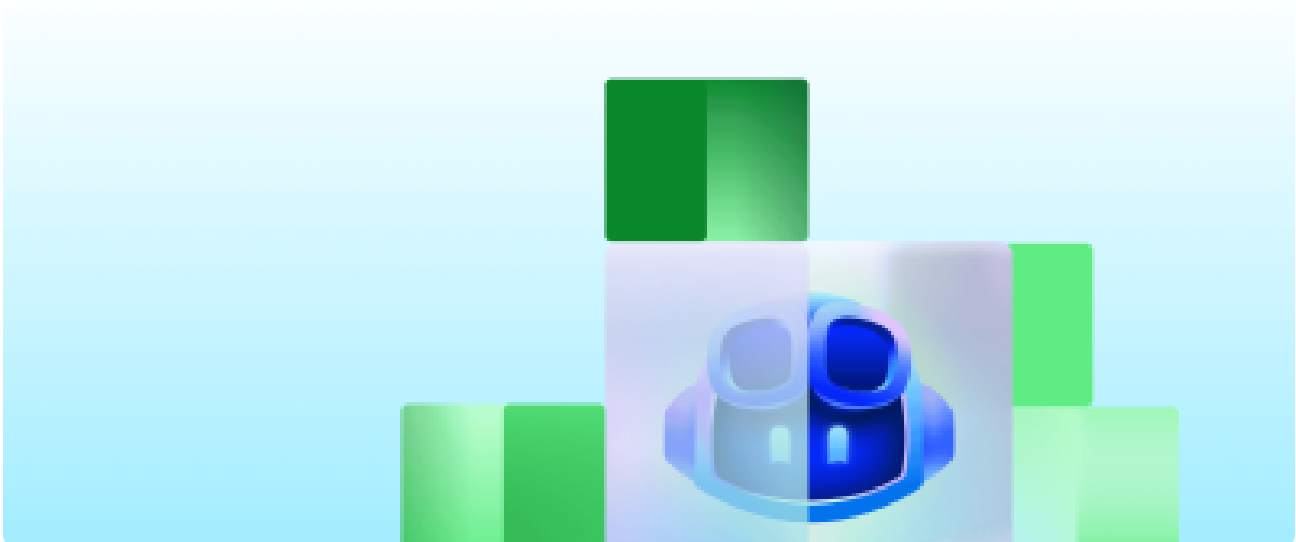


AI & ML

Take your local GitHub sessions anywhere

Kick off work in VS Code or the CLI, finish it from your phone. Remote control for GitHub Copilot sessions is now generally available on github.com and GitHub Mobile.

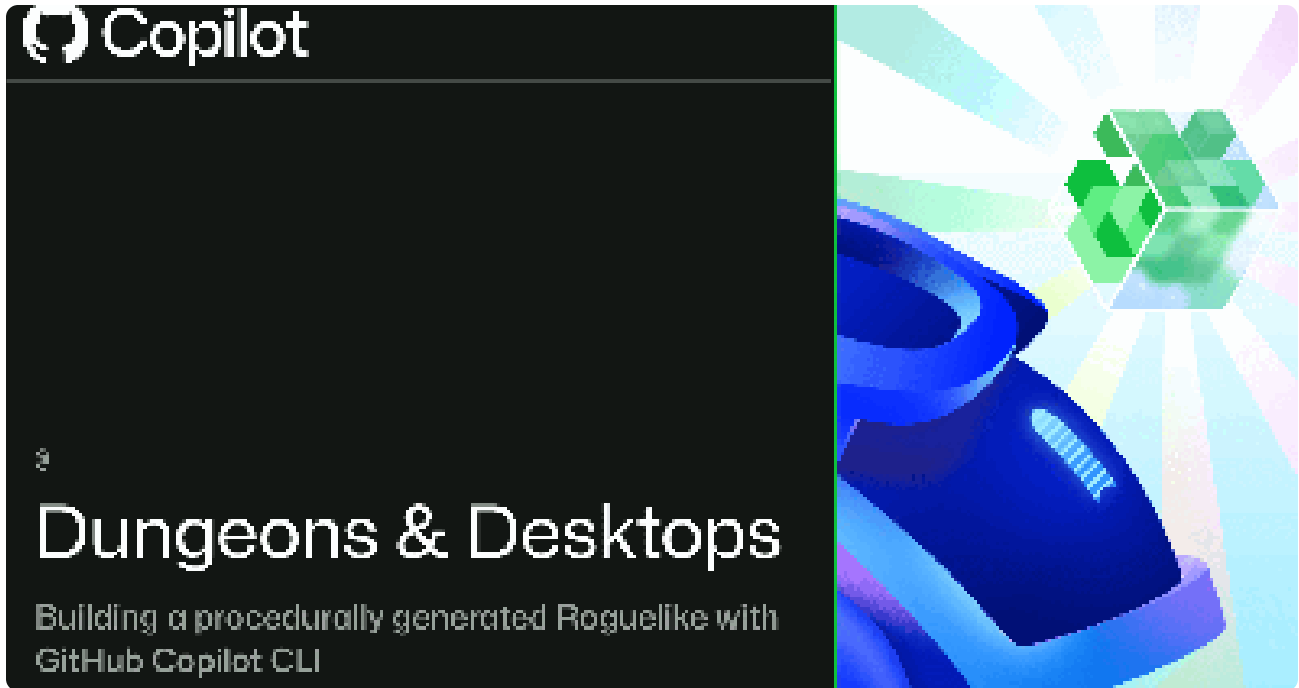
Candy Ho & Ellie Bennett



AI & ML

/ **Blog**

Eric Bailey



AI & ML

Dungeons & Desktops: Building a procedurally generated roguelike with GitHub Copilot CLI

Learn how one Hubber used GitHub Copilot CLI to build an extension that turns any codebase into a unique, roguelike dungeon.


Lee Reilly

Explore more from GitHub



Docs


/ Blog

[Go to Docs](#) 



GitHub


Build what's next on GitHub, the place for anyone from anywhere to build anything.

[Start building](#) 



Customer stories

Meet the companies and engineering teams that build with GitHub.

[Learn more](#) 



The GitHub Podcast

Catch up on the GitHub podcast, a show dedicated to the topics, trends, stories and culture in and around the open source developer community on GitHub.

/ Blog

We do newsletters, too

Discover tips, technical guides, and best practices in our biweekly newsletter just for devs.

Your email address

* Your email address

Yes please, I'd like GitHub and affiliates to use my information for personalized communications, targeted advertising and campaign effectiveness. See the [GitHub Privacy Statement](#) for more details.

Subscribe >

GitHub

Product

Features

Security

Enterprise

Platform

Developer API

Partners

Atom

Support

Docs

Community Forum

Training

Company

About

Blog

Careers

/ Blog

RESOURCES

© 2026 GitHub, Inc. [Terms](#) [Privacy](#) [Manage Cookies](#)
Do not share my personal information

