



# Graphite

# The Modern Software Developer: Code Review

CS146S · Stanford University, Fall 2025

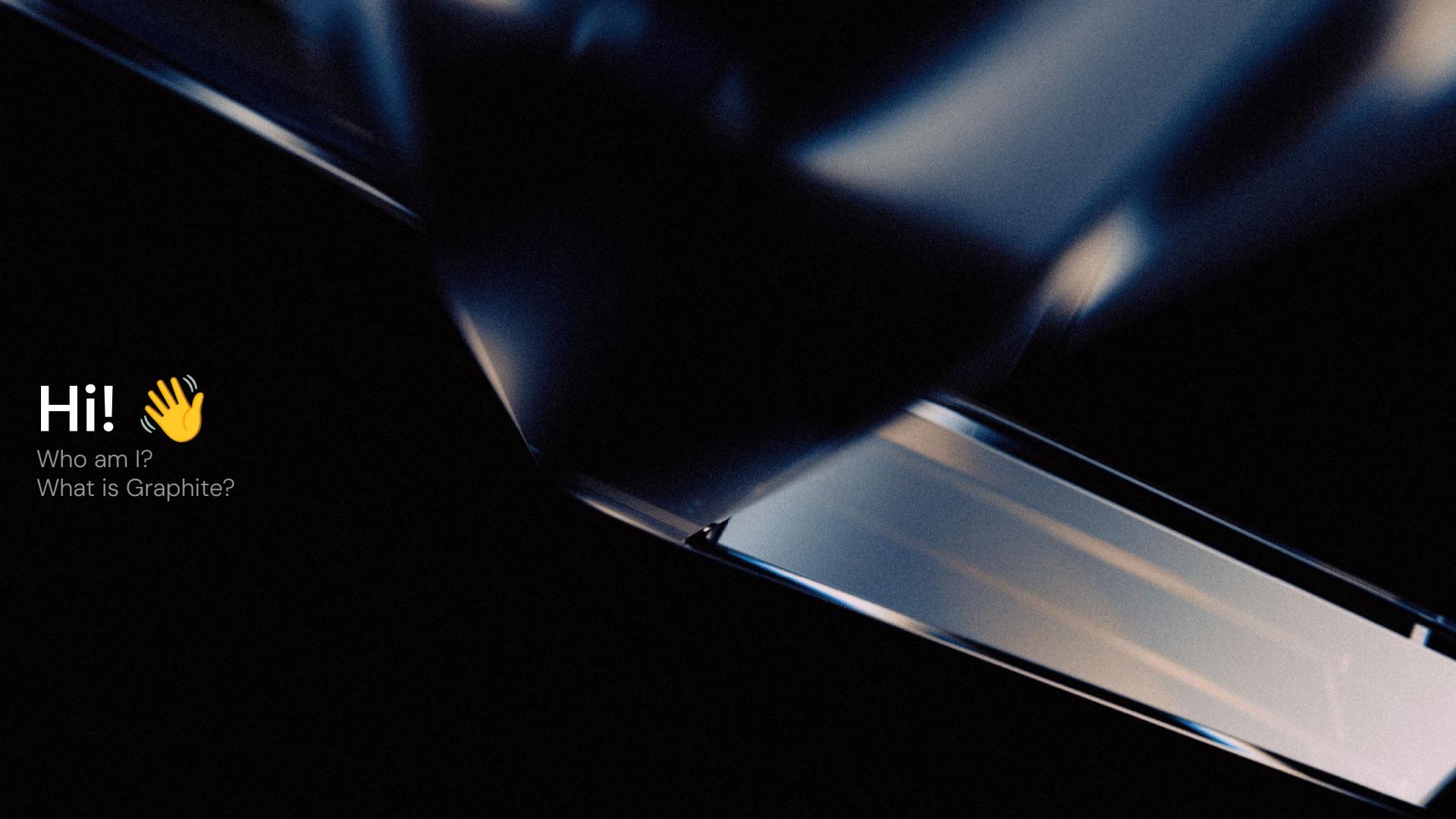
# Agenda

00 Hi! 

01 Collaborating with humans

02 Collaborating with AI

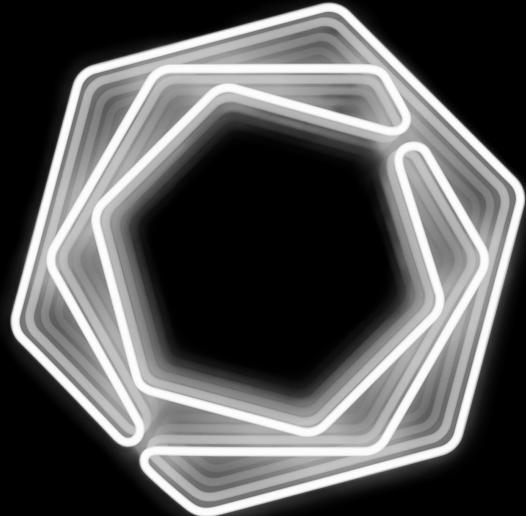
03 Software development  
in the limit



Hi! 🙌

Who am I?  
What is Graphite?

Graphite is the AI-powered code review platform helping developers create, review, and merge code changes.



# The AI-native code review platform

A screenshot of a GitHub pull request page for a PR titled "feat(pr-redesign): persist Info section expanded state to local storage". The interface includes a sidebar with navigation, a main content area with sections like "What changed?", "Why?", and "Discussion", and a detailed code diff view at the bottom. Overlaid on the right side is a "Graphite Chat Beta" window. This window has a sidebar with "Explaining the Pull Request" and "PR v1 [latest]". It contains a "Thinking" section with text about persisting expanded states and a "Key changes:" list. Below this is a "Why is this component?" section with a "Thinking" part about the Disclosure component. At the bottom, there's a text input field asking "Ask Graphite about this PR...".

```

monologue #60925
feat(pr-redesign): persist Info section expanded state to local storage
Eli Howey 09-08-feat_pr-redesign_persist_info_section... → main 6 files +45/-5 Updated yesterday
Your review is requested on EliHowey's PR
Stack 1 of 1
Share stack ⌂ ...
What changed?
PR info sections in the redesigned Info content will now persist their most recent expanded state to local storage.
This allows users to collapse sections permanently, which is useful for sections that they rarely never use.
Why?
Fixes GT-17695
Discussion 1
Show files Compare Base → v1 Latest version ▾
.../ic/graphite-app/src/router-pages/pull-request/partials/code-diff/info-column/PrAssigneesSection.tsx 4+/1 TypeScript Viewed P ...
19 lines hidden
28 import { PopoverMenu } from ".../lib/first-draft/popover-container";
29 import { AssigneeSelector } from ".../lib/selectors/AssigneeSelector";
30 import type { PrPageContext } from ".../models";
31
32 export const PrAssigneesSection = observer(function()
33
34 import { PopoverMenu } from ".../lib/first-draft/popover-container";
35 import { AssigneeSelector } from ".../lib/selectors/AssigneeSelector";
36 import type { PrPageContext } from ".../models";
37
38 + import { useInfoSectionOpenState } from ".../useInfoSectionOpenState";
39
40 export const PrAssigneesSection = observer(function()

```

Our AI agent can answer questions about the PR.

It is fully codebase aware.

It can also propose updates, helping reviewers quickly resolve comments and CI.

# Graphite at a glance

Graphite helps engineers ship more.

## A best-in-class code review platform built on-top of GitHub.

Graphite provides a streamlined interface to create, review, and merge PRs, instantly and bi-directionally synced with GitHub.

## Trusted by the developers and companies you already know.

We're fortunate enough to collaborate with some of the biggest names in technology, and improve developer velocity across a wide swath of metrics.

**+33%**

Increase in PRs shipped per developer after adopting Graphite at Shopify

**+21%**

LoC/eng at Asana

**-74%**

Time between PRs merged at Ramp

**100,000s+**

Trusted by 100,000s of developers at 1,000s of organizations

# Collaborating with humans

How developers work together

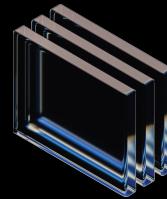
# How developers collaborate

CREATE



A developer proposes a set of code changes, this atomic unit is called a **pull request**.  
(a.k.a. diff, patch, changelist, or merge request)

REVIEW

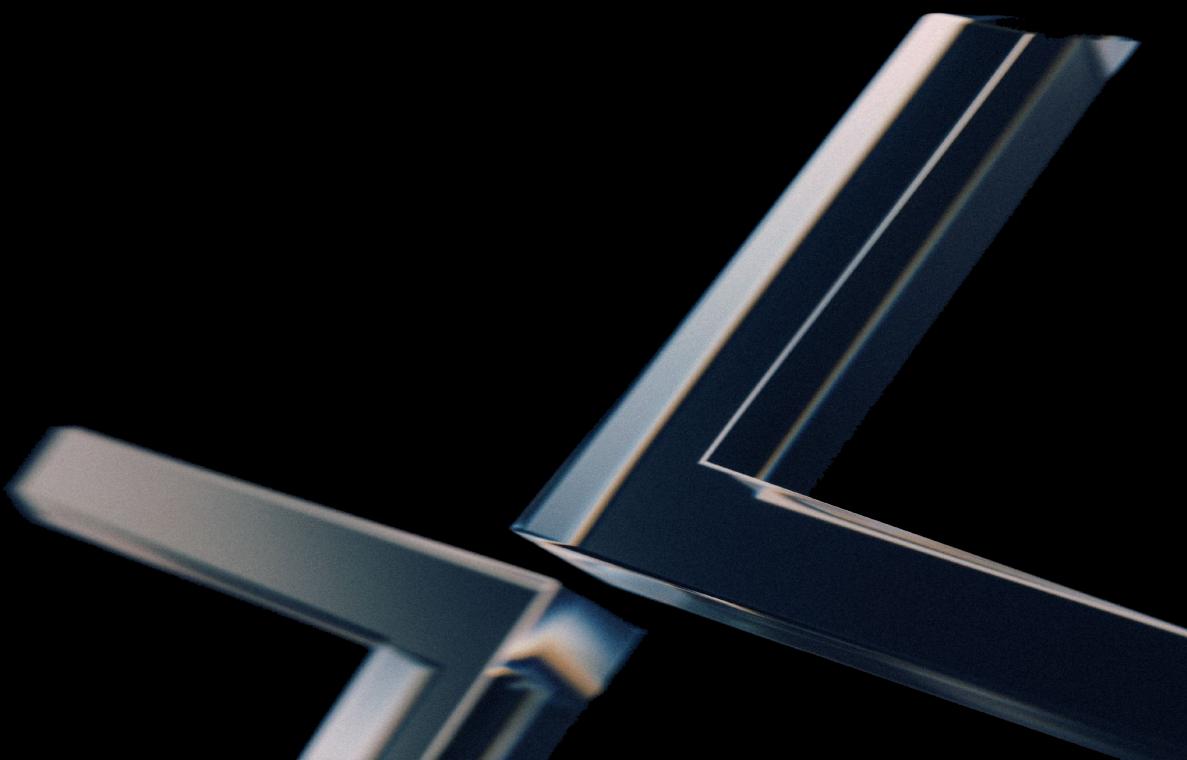


Another developer refines (suggests improvements) through **comments** and ultimately **approves** the pull request.

MERGE



The original developer **merges** their pull request back into the codebase.



---

AI means more code than ever is being created, but that also means more code needs to be reviewed and merged.

# The birth of code review: Fagan Inspections

## Abstract:

We can summarize the discussion of design and code inspections and process control in developing programs as follows: 1. Describe the program development process in terms of operations, and define exit criteria which must be satisfied for completion of each operation. 2. Separate the objectives of the inspection process operations to keep the inspection team focused on one objective at a time: Operation Overview Preparation Inspection Rework Follow-up Objective Communications/education Education Find errors Fix errors Ensure all fixes are applied correctly 3. Classify errors by type, and rank frequency of occurrence of types. Identify which types to spend most time looking for in the inspection. 4. Describe how to look for presence of error types. 5. Analyze inspection results and use for constant process improvement (until process averages are reached and then use for process control).

Published in: IBM Systems Journal ( Volume: 15 , Issue: 3, 1976)

Page(s): 182 - 211

DOI: [10.1147/sj.153.0182](https://doi.org/10.1147/sj.153.0182)

Date of Publication: 31 December 1976 

Publisher: IBM

Print ISSN: 0018-8670

IBM, 1976: an engineer named Michael Fagan introduces Fagan Inspections.

([Link to paper](#))

# The birth of code review: Email patches

**From** Icenowy Zheng <REDACTED>  
**Subject** [PATCH v3 5/7] clk: sunxi-ng: add support  
for the Allwinner H6 CCU  
**Date** Fri, 23 Feb 2018 20:35:53 +0800

The Allwinner H6 SoC has a CCU which has been largely rearranged.

Add support for it in the sunxi-ng CCU framework.

```
diff --git a/.../bindings/clock/sunxi-ccu.txt
b/.../bindings/clock/sunxi-ccu.txt
index 4ca21c3a6fc9..9ae27881c924 100644
--- a/.../bindings/clock/sunxi-ccu.txt
+++ b/.../bindings/clock/sunxi-ccu.txt
@@ -20,6 +20,7 @@ Required properties :
      - "allwinner,sun50i-a64-ccu"
      - "allwinner,sun50i-a64-r-ccu"
      - "allwinner,sun50i-h5-ccu"
+
      - "allwinner,sun50i-h6-ccu"
      - "nextthing,gr8-ccu"
```

Emailed patches replace printed code.

The Linux kernel still does this.

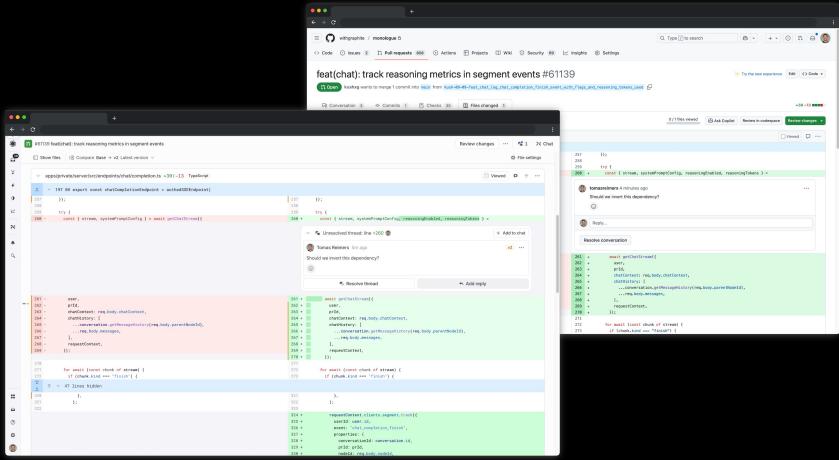
# The birth of code review: Mondrian



Google, early 2000s: Guido van Rossum\* introduces **Mondrian**: a web UI for review.

\*yes, Python's Benevolent Dictator

# The birth of code review: Online tools



Tools such as:

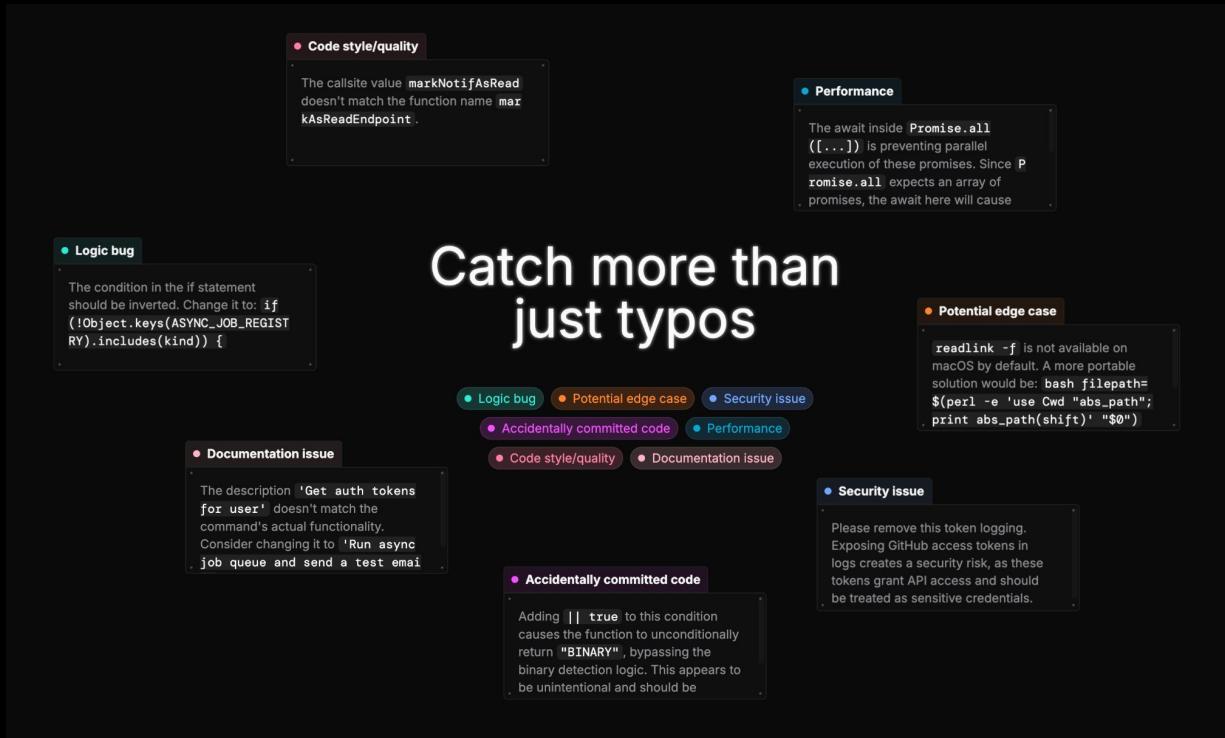
- Review Board
- Gerrit / Critique (Google)
- Phabricator (Facebook)
- GitHub

Make their way into the public tool chain, popularizing the practice of code review.

# Collaborating with AI

Keeping up with super-human authors.

# AI can already catch more than typos



Graphite will proactively scan PRs for bugs.

It posts potential issues to both Graphite and GitHub.

It works great out of the box and is fully customizable.

With AI, do humans even need to  
do code review?

# The purpose(s) of code review (in order)

- 01 Alignment confirmation
- 02 Knowledge diffusion
- 03 Proofreading

# How will humans collaborate with AI?

PLATFORM



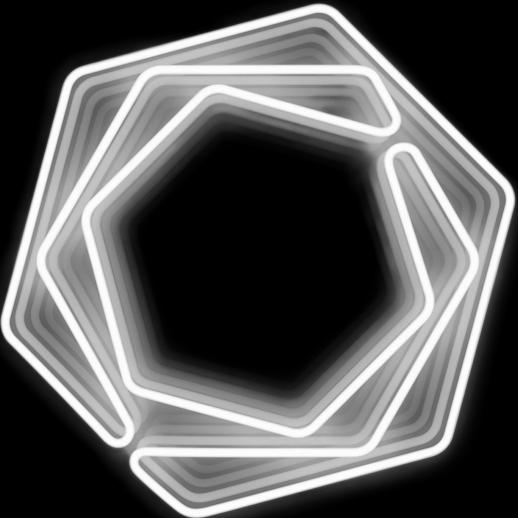
Better for context gathering and  
**augmenting** human reviewers.

PARTICIPANT



Better for **replacing** human  
reviewers entirely.

# Live demo



# The limits of AI (today)

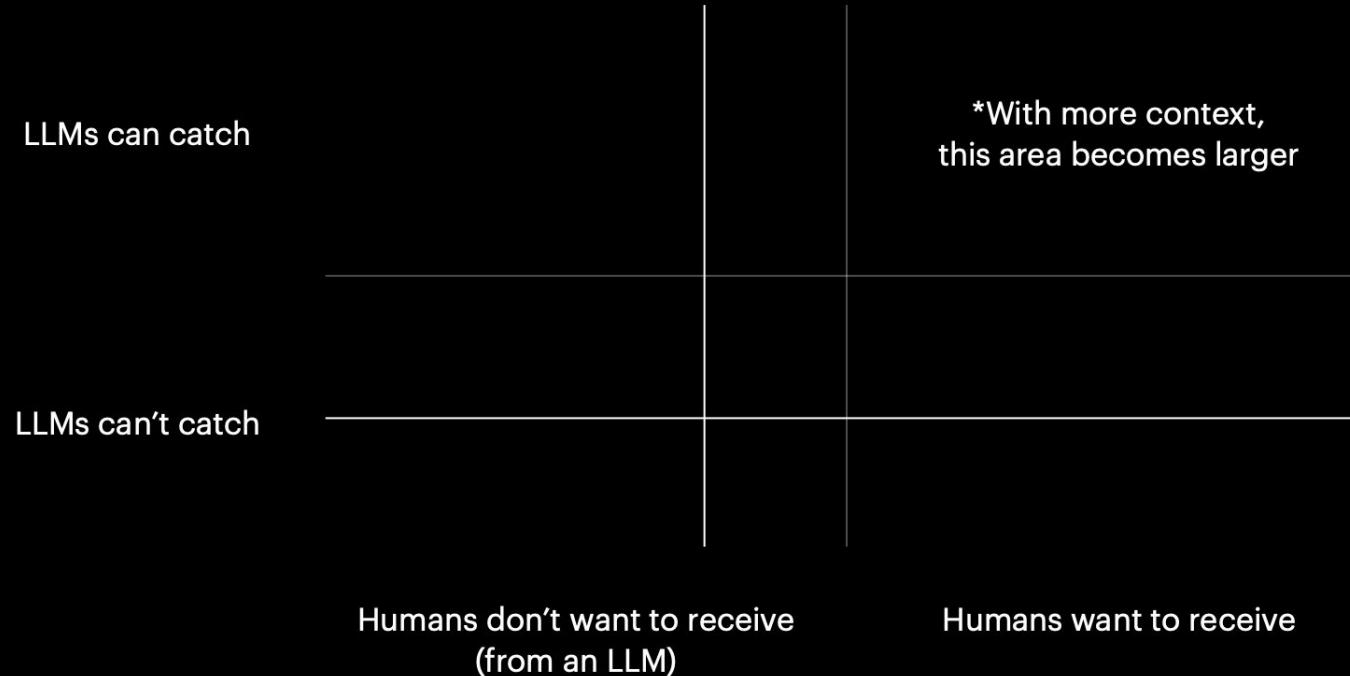
LLMs can catch

LLMs can't catch

Humans don't want to receive  
(from an LLM)

Humans want to receive

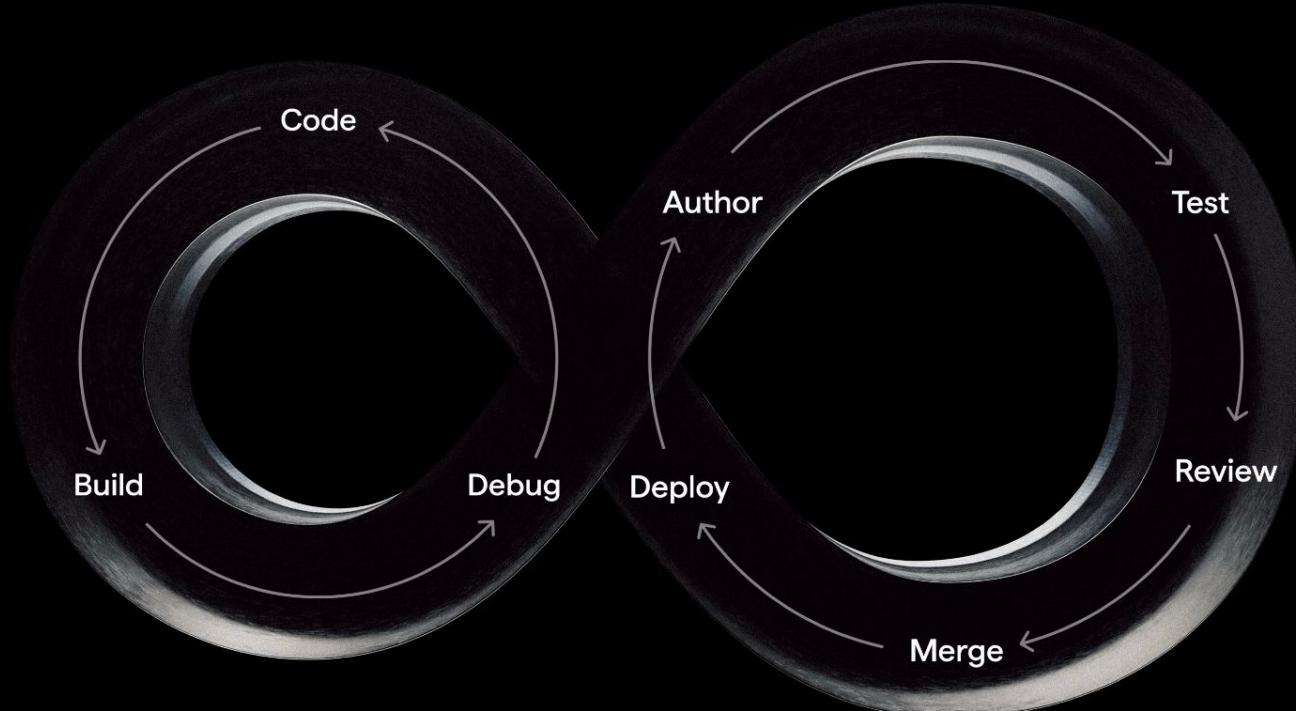
# The limits of AI (tomorrow)



# Software development in the limit

Where do we go from here.

# Software development has always had two parts



# Software development has always had two parts



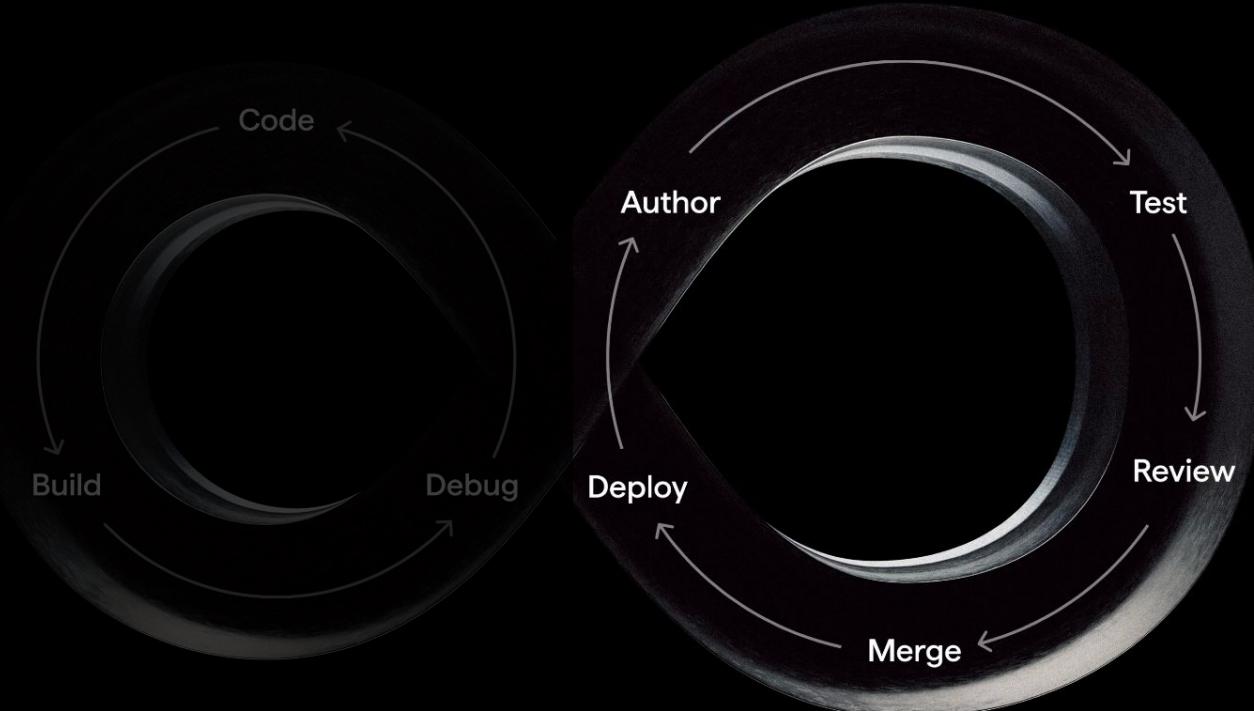
# Software development has always had two parts



# AI is already making the inner loop 10x faster

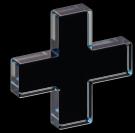


# What happens to the outer loop?



# Three doors for the next generation software dev

CYBORG



Developers directly **review** changes, augmented by AI.

If they're not **authoring** the code, does this still make sense?

EM



Developers **manage** AI directly.

Underwriting architecture, but maybe not technical specifics.

AGENCY



Developers treat AI as a third-party contractor.

Underwriting **product requirements** but **not code**.

## AI Review Agent

Graphite Agent is your AI-powered companion to catch bugs, enforce your organization's conventions, and make sure errors and security vulnerabilities stay out of your codebase.

```
142  
143  
144  });
145 const durationMS = startTime
146   .diff(DateTime.now())
147   .as('milliseconds');

148
149 serverContext.slog.verbose({
150   message: `Rebase completed in ${rebaseDurationMs}ms`,
151
152
153
154})
```

Graphite Reviewer just now ...

The calculation of durationMs is incorrect. It's currently calculating the duration from the current time to the start time, which will result in a negative duration.

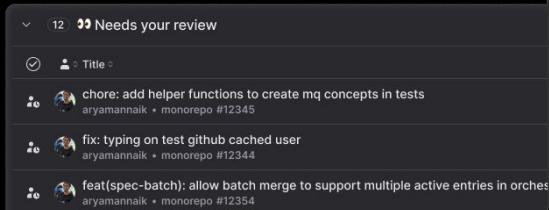
To fix this, swap the order of the arguments in the diff()  
method: DateTime.now().diff(startTime)

Spotted by Graphite reviewer

Is this helpful?

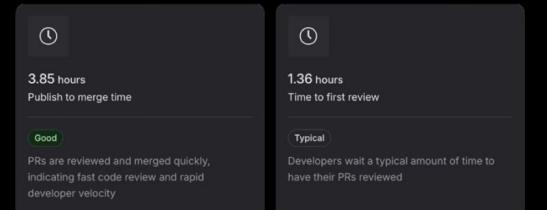
## Stacking

A git workflow optimized for authors who want to keep developing without being blocked on their reviewers.



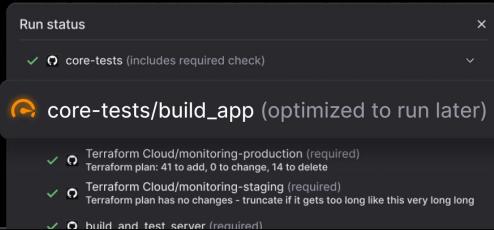
## Measure developer productivity

See exactly how much more efficient AI-generated coding tools & Graphite's code review are making your developers with actionable insights.



## Smarter CI

Predictive CI that only runs when you need it. Saving your team time and money.



## A review experience built for teams

Supercharge your team with reviewer assignment, merge queues, automations, and insights.



# Thank you! Any questions?

 Graphite