Claude Code GitHub Workflows: 2026 Case Studies & Setup

Edition of November 13, 2026 — facts verified November 13, 2026. Current edition always at https://tygartmedia.com/claude-code-github-workflows/.

Direct Answer (November 13, 2026): To automate GitHub PR reviews and issue fixes, drop `anthropics/claude-code-action@v1` into `.github/workflows/claude.yml`, commit a concrete `CLAUDE.md` at the repo root, and pass bounded flags like `--model claude-opus-5-5 --max-turns 10` (or `claude-sonnet-5-5` for routine triage; current models are Fable 5.2, Opus 5.5, Sonnet 5.5, and Haiku 5.5, all 1M context).

Last refreshed: November 13, 2026 · Sources: Claude Code docs & Anthropic models overview

Seven hours. That's how long it took Claude Code to autonomously navigate a 12.5-million-line codebase and implement a production-ready activation vector extraction method in vLLM for Rakuten's engineering team — a task their developers hadn't attempted because the codebase was simply too large to reason about at human speed. The result: 99.9% numerical accuracy and a project timeline that compressed from 24 working days to 5.

That's not a demo. That's a production case study. And it tells you more about where Claude Code + GitHub workflows are in 2026 than any benchmark comparison.

This post breaks down three real-world patterns from teams getting measurable results with Claude Code on GitHub: what they set up, how they structured the work, and what's actually driving the outcomes.

The Setup That Enables Everything: CLAUDE.md First

CLAUDE.md first — the setup that enables everything.

Before any CI/CD integration, the teams getting results share a common starting point: a well-structured CLAUDE.md file that tells Claude Code exactly how to behave in their specific codebase.

Andrej Karpathy's lean 65-line CLAUDE.md — originally shared as a personal config — accumulated over 100,000 GitHub stars by early 2026, which tells you something: developers are desperately hungry for a working template. What made it valuable wasn't length. It was specificity. Four behavioral rules that directly address LLM coding failure modes: don't assume context you don't have, prefer surgical edits over full rewrites, surface tradeoffs rather than hiding them, and treat goals as declarative targets with verification loops.

That last principle is the most important for GitHub integration. When Claude knows the goal is "this PR should pass CI and not break existing tests" rather than "write code," the outputs change materially. You get tighter diffs, fewer phantom dependencies, and PRs that actually close the issue they were created for.

Your CLAUDE.md lives in the repo root and commits alongside your code. It travels with the codebase. Claude Code GitHub Actions picks it up automatically when you use `anthropics/claude-code-action@v1` — no additional configuration required.

The GitHub Actions Setup

The GitHub Actions setup.

The GA version of Claude Code GitHub Actions (`@v1`, released in 2026) simplified configuration considerably from the beta. Here's the minimum viable setup:

```
name: Claude Code
on:
  issue_comment:
    types: [created]
  pull_request_review_comment:
    types: [created]
jobs:
  claude:
    runs-on: ubuntu-latest
    steps:
      - uses: anthropics/claude-code-action@v1
        with:
          anthropic_api_key: ${{ secrets.ANTHROPIC_API_KEY }}
```

Drop this in `.github/workflows/claude.yml`, install the GitHub app at `https://github.com/apps/claude`, add your `ANTHROPIC_API_KEY` secret, and you can start triggering Claude with `@claude` in any PR or issue comment. The fastest path is running `/install-github-app` inside your Claude Code terminal session — it walks through the app installation, permissions, and secret setup in a single guided flow.

For teams on Google Vertex AI or Amazon Bedrock — which matters if you're operating in a regulated environment — the action supports both via Workload Identity Federation. Bedrock uses region-prefixed model strings (e.g., `us.anthropic.claude-sonnet-5-5` or legacy `claude-sonnet-4-6` depending on region availability); Vertex pulls the project ID from the auth step automatically.

The action defaults to Sonnet. For heavy refactoring tasks on large codebases, bump it explicitly:

```
claude_args: "--model claude-opus-5-5 --max-turns 10"
```

`claude-opus-5-5` is Anthropic's current daily-driver Opus ($4/$20 per MTok, with `claude-fable-5-2` at $8/$40 for flagship runs, superseding legacy Opus 4.7). For routine PR review and issue triage, `claude-sonnet-5-5` ($2/$10) or `claude-haiku-5-5` ($0.10/$0.50 ≤100K) is faster and more cost-efficient. The `--max-turns` flag prevents runaway jobs from consuming your Actions budget on open-ended tasks — set it to 5 for review workflows, 10–15 for implementation tasks.

Rakuten: Autonomous Work at Codebase Scale

Rakuten's engineering team used Claude Code to tackle vLLM — a 12.5-million-line open-source inference framework — without prior familiarity with the codebase. Claude Code ran autonomously for seven hours, implemented the activation vector extraction method, and delivered 99.9% numerical accuracy.

The workflow wasn't magic. It was structured: a clear task definition scoped to a specific deliverable, a CLAUDE.md establishing Rakuten's code patterns and testing requirements, and an allowance for autonomous tool use across the codebase. The result wasn't just the implementation — it was the compression of a project timeline from 24 working days to 5. That's a 79% reduction in time-to-market for a complex systems task, on a codebase that would take a new engineer weeks just to orient themselves in.

The lesson: Claude Code's GitHub integration handles scale that would be cognitively impossible for a single developer to navigate in a normal sprint. The constraint isn't Claude's ability to read code — it's whether you've given it a goal specific enough to work from.

TELUS: 500,000 Hours at the Portfolio Level

TELUS is a different kind of case. Rather than a single high-stakes task, TELUS rolled Claude Code out across engineering teams organization-wide and measured cumulative impact: 500,000 hours saved, engineering code shipping 30% faster, and over 13,000 custom AI solutions built by their own teams.

The 13,000 solutions number is the most telling. It means that once developers have Claude Code in their GitHub workflow, they stop waiting for platform teams to build internal tooling. They build it themselves — PR automation, internal API clients, test generators, schema migration scripts — because the cost of shipping something useful dropped to a well-scoped conversation with an `@claude` trigger.

The 30% speed improvement in code shipping translates directly to cycle time. Fewer context switches between writing code and writing tests. Less time waiting for review when PRs arrive with Claude-generated documentation already attached. That number compounds across a large engineering org in ways that individual productivity improvements don't.

The Pattern Across All Three

The pattern across all three case studies.

Three things appear consistently across every team getting results with Claude Code on GitHub:

A real CLAUDE.md — not a placeholder. A file with codebase-specific rules: what patterns to follow, what to avoid, how tests should be structured, what done looks like. Karpathy's version works because it encodes failure modes. Yours should encode your team's standards.

Goal-oriented triggers, not open-ended requests. `@claude implement the auth middleware from issue #42 following our existing token validation pattern` outperforms `@claude help with this`. The action inherits your CLAUDE.md automatically, but the trigger needs to state a specific, bounded goal with a clear definition of done.

Autonomous mode for the right task class. Bounded, well-defined tasks — implement this spec, fix this failing test, write a migration for this schema change — run better autonomously than open-ended exploration. Use `--max-turns 10` and let it run. Reserve manual review for the output, not the process.

Where to Start

Run `/install-github-app` in your Claude Code terminal. That one command handles app installation, permission setup, and secret configuration. Add a CLAUDE.md to your repo root — even five lines of real project standards beats a blank file. Open a test issue, write a specific `@claude` comment with a bounded task, and watch the action run.

Rakuten's 7-hour autonomous run and TELUS's 500,000 hours didn't start with a six-month AI rollout plan. They started with a config file, a workflow YAML, and a task specific enough for Claude to actually finish.

■ Topic Cluster Index: Deep-Dive Guides Explore our comprehensive, step-by-step guides in the **Claude Code Insider** series: ■Claude Code — Every Question Answered (Complete FAQ 2026) ■We Published Hundreds of Articles About Claude — And Some of Them Were Wrong. Here's Everything We're Doing About It. ■Claude Code for Teams: What to Commit, What to.gitignore, and What Actually Survives a Pull Request ■Claude Code Hooks: The Workflow Control Layer That Actually Enforces Your Rules ■Claude MCP in 2026: What Actually Changed and How to Configure It Without Wasting Tokens ■Claude Code Pricing in May 2026: What $20, $100, and $200 a Month Actually Buy You ■Claude Code Case Studies: What the Numbers Actually Say in 2026 ■Claude Code managed-settings.json: The Org-Wide Policy File Most Teams Skip ■The Plan-Mode-Plus-Hooks Pattern: How to Actually Trust Claude Code in a Production Repo ■Claude Code vs Cursor in May 2026: A Practitioner's Honest Take After Agent View and Composer 2.5 ■Why Sentry Is the Second MCP Server You Should Install in Claude Code (Not GitHub) ■10 Notion Workers in 3 Hours: What Happens When Claude Code Does the Typing ■What I Actually Did Last Night ■The Operator's Stack ■You Don't Need a Developer. You Need a Better Workflow. ■The Bus Factor Problem ■The Trust Gap ■When I Stopped Being the Bottleneck ■Installing Claude Code on Windows in 2026: The Native Installer Walkthrough That Actually Works ■What Actually Drives Claude Code Adoption: Inside a 30-Engineer Rollout That Held 35% at Month Four ■Claude Code Server-Managed Settings: The Admin Console Push That Replaces Your MDM Pipeline ■The Worktree Workflow: How Many Parallel Claude Code Sessions Are Actually Worth Running ■Claude Code vs Cursor in 2026: Token Efficiency, Agent Teams, and What I Actually Run ■MCP Scopes in Claude Code: Why –scope Is the Flag That Saves Your Team

Primary & Reference Links Cited on This Page

- Claude Code docs — `https://code.claude.com/docs/en/overview`
- Anthropic models overview — `https://platform.claude.com/docs/en/about-claude/models/overview`
- Tygart Media: Claude Code — Every Question Answered (Complete FAQ 2026)
- Tygart Media: We Published Hundreds of Articles About Claude — And Some of Them Were Wrong. Here's Everything We're Doing About It.

- Tygart Media: Claude Code for Teams: What to Commit, What to .gitignore, and What Actually Survives a Pull Request
- Tygart Media: Claude Code Hooks: The Workflow Control Layer That Actually Enforces Your Rules
- Tygart Media: Claude MCP in 2026: What Actually Changed and How to Configure It Without Wasting Tokens
- Tygart Media: Claude Code Pricing in May 2026: What $20, $100, and $200 a Month Actually Buy You
- Tygart Media: Claude Code Case Studies: What the Numbers Actually Say in 2026
- Tygart Media: Claude Code managed-settings.json: The Org-Wide Policy File Most Teams Skip
- Tygart Media: The Plan-Mode-Plus-Hooks Pattern: How to Actually Trust Claude Code in a Production Repo
- Tygart Media: Claude Code vs Cursor in May 2026: A Practitioner's Honest Take After Agent View and Composer 2.5
- Tygart Media: Why Sentry Is the Second MCP Server You Should Install in Claude Code (Not GitHub)
- Tygart Media: 10 Notion Workers in 3 Hours: What Happens When Claude Code Does the Typing
- Tygart Media: What I Actually Did Last Night
- Tygart Media: The Operator's Stack
- Tygart Media: You Don't Need a Developer. You Need a Better Workflow.
- Tygart Media: The Bus Factor Problem
- Tygart Media: The Trust Gap
- Tygart Media: When I Stopped Being the Bottleneck
- Tygart Media: Installing Claude Code on Windows in 2026: The Native Installer Walkthrough That Actually Works
- Tygart Media: What Actually Drives Claude Code Adoption: Inside a 30-Engineer Rollout That Held 35% at Month Four
- Tygart Media: Claude Code Server-Managed Settings: The Admin Console Push That Replaces Your MDM Pipeline
- Tygart Media: The Worktree Workflow: How Many Parallel Claude Code Sessions Are Actually Worth Running
- Tygart Media: Claude Code vs Cursor in 2026: Token Efficiency, Agent Teams, and What I Actually Run
- Tygart Media: MCP Scopes in Claude Code: Why –scope Is the Flag That Saves Your Team