GitHub Copilot in Practice

Building GitHub with GitHub — An Engineering Manager's Field Guide

David O'Regan & Klaire Baek

GitHub Copilot in Practice

Building GitHub… with GitHub Copilot

David O'Regan — Engineering Manager, Agent Platform (Copilot Mission Control) Klaire Baek — Engineering Manager, Copilot Platform

> Who is this for?
> 🧑‍💻 Developers (Individual Contributors) — level up your workflow
> 📋 Managers (Engineering Managers & Product Managers) — lead teams in an agentic world

Agenda

David's Sections

- §1 CCA → PRs — Issue to shipped code
- §2 Mission Control — Steering & multi-session
- §3 Automate the Toil — Agentic workflows

Klaire's Sections

- §4 AI as a Thinking Partner
  - Implementation tradeoffs
  - Documentation gaps
  - Backlog management
- §5 Cheat Sheet — Links & resources

§1 — Using CCA to Make PRs

Copilot Coding Agent (CCA) — from issue to shipped code

🧑‍💻 David

What is CCA?

Copilot Coding Agent works independently in the background — just like a human developer. It gets its own ephemeral dev environment (powered by GitHub Actions) where it can explore code, make changes, and run tests.

How It Works

1. Assign from Issues, VS Code, CLI, or the Agents panel on any GitHub page
2. CCA creates a branch, writes code, runs your linters and tests in its environment
3. Opens a PR and requests your review
4. You steer via PR comments — CCA iterates

What It Handles Well

- Fix bugs & address tech debt
- Implement incremental new features
- Improve test coverage
- Update documentation
- Security campaign alerts
- "Nice to have" backlog items that never get prioritized

CCA in Action — A Real PR

PR #15712 on `github-ui` — "Rich plan UI: add implement action"

We needed to add an "Approve plan and start work" button to Copilot's plan approval UI, increase the plan summary max-height from 300px → 400px, and ship a full shared component refactor with 13 tests.

```
Copilot CLI start          Kicked off from the terminal
   │
   ├──→  🤖 CCA              19 commits, 21 files, +1,437 / -526 lines
   │                          Built components, tests, stories, CSS
   ├──→  🔍 Copilot Review    Auto-reviewed 4 files on PR open
   │
   ├──→  💬 "@copilot fix"    3 rounds of steering (CI + lint)
   │
   ├──→  ✅ 4 approvals       Team reviewed architecture & design
   │
   └──→  🚀 Merge queue       Shipped to production on github.com
```

What Made This Issue CCA-Ready

Issue #1553 — "Output Rich plan UI in Summary"

The issue was specific enough for CCA to ship a large PR with confidence:

✅ What We Included

- Exact components: `PlanApprovalDisplay`, `ChoiceButton`, `ScrollableSummary`
- Acceptance criteria: new "implement" action, max-height 300→400px, 13 unit tests
- Cherry-pick ref: shared refactor from PR #15567
- Scope: `packages/agent-sessions/` only

⚠️ What Would Have Failed

- "Improve the plan UI" — too vague
- "Refactor all prompt components" — too broad
- No mention of existing patterns to follow
- No design spec or pixel values
- Multiple unrelated concerns in one issue

> 💡 Tip
>
> CCA produced 13 tests, 6 new shared components, and Storybook stories — because the criteria told it exactly what "done" looks like.

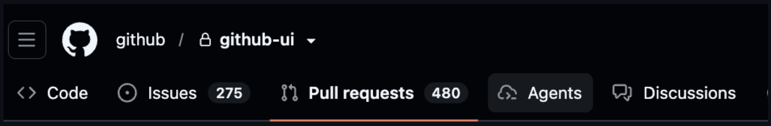
█ §2 — Mission Control & Steering

Multiple async sessions, course correction, every surface

🧑‍💻
David

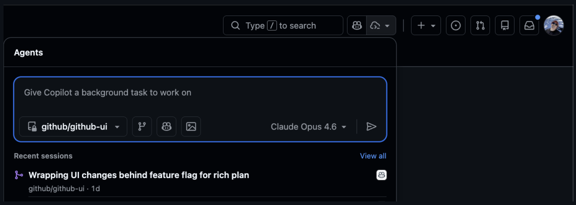
The Multi-Session Workflow

The Agents tab in your repository is your mission control. It's not just for issues — it's a full entry point for CCA:



Start Tasks From Anywhere

- Agents tab — in any repository
- Agents panel — top of any GitHub page
- Issues — assign Copilot as assignee
- Dashboard — github.com/copilot
- VS Code / JetBrains — `/task` in Chat
- Copilot CLI — type a prompt in terminal
- GitHub Mobile — agents task page



Give Copilot a background task to work on

github/github-ui · Claude Opus 4.6

Wrapping UI changes behind feature flag for rich plan
github/github-ui · 1d

Continuity Across Surfaces

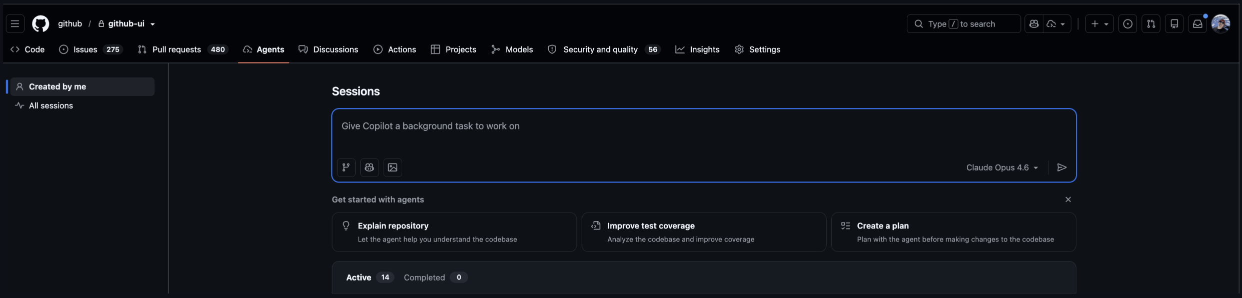
Start a task in one place, continue anywhere:

- "Open in VS Code" — one click from the session view
- `copilot --resume=ID` — pick up in CLI right where you left off
- Codespaces — open from session view
- Steer live — type while CCA works, it adapts after its current tool call

```
┌────────────────┐   ┌────────────────┐   ┌────────────────┐
│  Issue #142    │   │  Issue #187    │   │  Issue #203    │
│  CCA Working   │   │  CCA Working   │   │  CCA Working   │
│  ⏳ 7 min      │   │  ✅ PR Ready   │   │  ⏳ 3 min      │
└───────┬────────┘   └───────┬────────┘   └───────┬────────┘
        ▼                    ▼                    ▼
  You steer           You review            You wait
```

Session Continuity — Work Follows You

CCA runs in the cloud, but developers move between tools constantly. Session continuity makes agent sessions portable — start in one place, pick up in another with the same context, history, and next steps.



Why It Matters

- Progress isn't tied to a single device
- No duplicated sessions or lost context
- Safe to adopt CCA in real workflows
- One developer identity across web, IDE, CLI, and mobile

The Flows

- Start on web → continue on mobile
- Start on CLI → continue on web
- Start on web → continue in VS Code
- Steer from mobile while on the go

What's Live Now

- Steering in production — redirect CCA from any surface while it works
- Live updates on Mobile — watch progress on the staff build in real-time
- `copilot --resume=ID` — CLI pickup
- "Open in VS Code" — one click

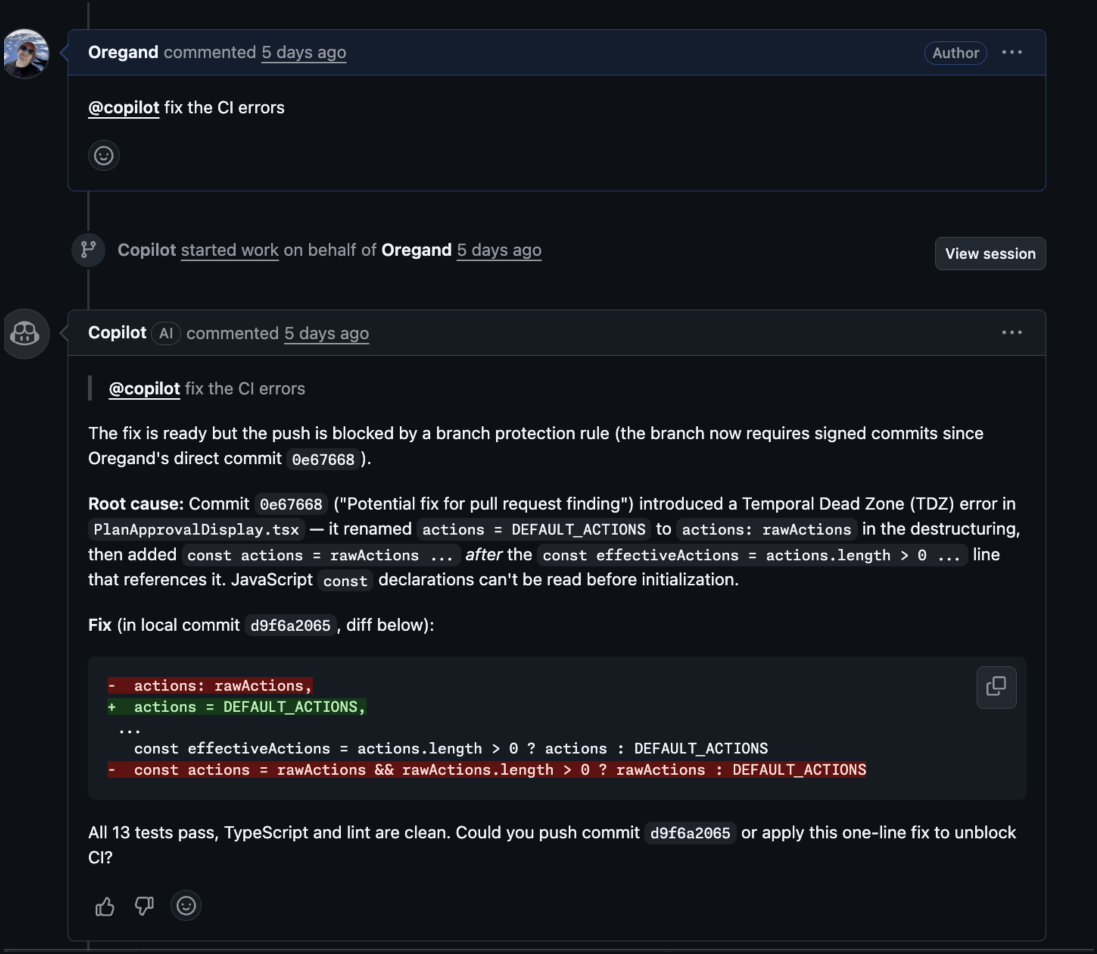
🎬 Demo

Steering in production + live updates on the staff build of mobile

Play: `open demo.mov`

Steering — Real Example from PR #15712

The PR had CI failures and review comments. 3 rounds of `@copilot fix`:



Oregand commented 5 days ago Author

@copilot fix the CI errors

Copilot started work on behalf of **Oregand** 5 days ago View session

Copilot AI commented 5 days ago

> @copilot fix the CI errors

The fix is ready but the push is blocked by a branch protection rule (the branch now requires signed commits since Oregand's direct commit `0e67668`).

Root cause: Commit `0e67668` ("Potential fix for pull request finding") introduced a Temporal Dead Zone (TDZ) error in `PlanApprovalDisplay.tsx` — it renamed `actions = DEFAULT_ACTIONS` to `actions: rawActions` in the destructuring, then added `const actions = rawActions ...` *after* the `const effectiveActions = actions.length > 0 ...` line that references it. JavaScript `const` declarations can't be read before initialization.

Fix (in local commit `d9f6a2065`, diff below):

```diff
-  actions: rawActions,
+  actions = DEFAULT_ACTIONS,
 ...
   const effectiveActions = actions.length > 0 ? actions : DEFAULT_ACTIONS
-  const actions = rawActions && rawActions.length > 0 ? rawActions : DEFAULT_ACTIONS
```

All 13 tests pass, TypeScript and lint are clean. Could you push commit `d9f6a2065` or apply this one-line fix to unblock CI?

Round 1 — "@copilot fix the CI errors and address the open comments" CCA removed a restricted `@testing-library/user-event` import (not in the package deps), switched to `@github-ui/react-core/test-utils`, and wired up the missing `disabled` prop on `ChoiceButton`. ✅

Round 2 — CCA hit a branch protection rule (signed commits required). It diagnosed a Temporal Dead Zone bug it had introduced — a `const` referenced before initialization — and gave us the exact one-line fix to apply.

Round 3 — "@copilot fix the CI failures" CCA analyzed the failing jobs and reported: "Both failures are in `@github-ui/memex:test` — a known flaky Playwright issue. None of our changes touch memex. agent-sessions: TS ✅, Lint ✅, 13 tests ✅."

> Pro tip: @copilot fix <CI job URL> — CCA reads job logs directly.

Copilot Review + Human Review

On PR #15712, Copilot auto-reviewed 4 changed files the moment it opened:

Copilot Caught 🤖

- `effectiveActions` fallback when `actions` array is empty — no buttons would render
- Restricted `@testing-library/user-event` import not in package dependencies
- Missing `.actions` CSS class in `PermissionDisplay.module.css`
- Assessed: "not ready to merge"

Humans Caught 🧠

- `useEffect` measuring `scrollHeight` won't update on viewport resize (follow-up noted)
- `onApprove` callback design — component shouldn't call `createCommand` directly
- Cherry-pick from #15567 integrated correctly with the new `CCA_PLAN_ACTIONS` constant
- Overall design direction for shared components

> Copilot handles mechanical checks. Humans handle judgment calls.

█ §3 — Automate the Toil Away

Agentic workflows for your whole team

📋 David

GitHub Agentic Workflows

Imagine improvements automatically delivered each morning, ready to review. Issues triaged, CI failures analyzed, docs maintained, tests improved — all defined via simple markdown files.

How They Work

- Markdown in `.github/workflows/`
- Triggered by events or schedules
- Run Copilot, Claude, or Codex in containerized GitHub Actions
- Write tasks in natural language, not complex code

Guardrails Built In

- Read-only permissions by default
- Write ops need explicit approval via safe outputs (pre-approved actions)
- Sandboxed execution & network isolation
- Tool allowlisting per workflow
- AI agents operate within controlled boundaries — can't go rogue

Our Issue Triage Bot

Real workflow on `copilot-mission-control` — every new issue triggers it:

```yaml
on:
  issues:
    types: [opened, reopened]
safe-outputs:
  add-labels:
    allowed: [task, epic, duplicate, "copilot-candidate",
              "Needs Triage", "good first issue"]
    max: 4
  add-comment:
    max: 1
  assign-to-agent:
    name: "copilot"
    max: 1
  create-agent-session:
    max: 1
    allowed-repos: ["github/github", "github/github-ui"]
```

Reads the issue, scores complexity (1-5), checks suitability, only assigns Copilot if complexity ≤ 2 and all criteria pass.

Triage Bot in Action

Task #1675 — "Filter tasks by resource type"

Mobile team needed to query for only tasks with a pull request attached, so they could start consuming the API without the full Mission Control UI.

The triage bot analyzed it within seconds and posted:

```
Triage Complete ✅
Issue Type: task     Complexity Score: 2 (Simple)

Why? Backend infra for resource_type filtering already exists.
Only gap is in the HTTP handler — parse two new query params.
Follows existing patterns (like creator_id, status).
~10-15 lines of code.

Copilot Assignment: ✅ Assigned to Copilot
  - Clear acceptance criteria
  - Minimal code change, existing patterns
  - Backend support already complete
  - Testable with existing infra
```

> The bot scored it complexity 2, confirmed all suitability criteria,
> and assigned Copilot automatically. PR #1565 shipped it.

> Important
>
> The key to max velocity is good issue hygiene.
> Automate the triage, Copilot handles the rest.

§4 — AI as a Thinking Partner

Not a replacement for engineering judgment — a multiplier for it

📋 Klaire Baek — Engineering Manager, Copilot Platform

Klaire's Approach

> AI is a thinking partner, not a replacement for engineers.

My focus areas as an Engineering Manager:

- Execution quality — are we building the right things well?
- Developer productivity — what's slowing the team down?
- Organizational clarity — does everyone know why and what?

I use AI to go deeper on all three — without pulling engineers off their work to explain things to me.

Use Case 1 — Understanding Tradeoffs

As an Engineering Manager, I need to understand technical decisions — not just approve them.

Deep Dives

- Current architecture — how does the system actually work today?
- Code snippets — walk me through the critical paths
- Design patterns — what conventions does this codebase follow?

Decision Support

- Improvement areas — where are the pain points?
- Alternatives — what else could we do here?
- Tradeoffs — what do we gain and lose with each approach?

> I come to architecture discussions prepared — without interrupting engineers.

Use Case 2 — Closing Documentation Gaps

New services often lack documentation. Knowledge gets siloed.

The Problem

- Siloed knowledge across engineers
- Dependencies and risk areas unclear
- Slow onboarding for new members
- Cross-team confusion

AI Generates

- Service overviews — what does this do?
- Dependency maps — what talks to what?
- Failure risks — what breaks when X fails?
- Missing doc areas — what's undocumented?

> Don't wait for engineers to write docs. AI for the first draft — engineers refine.

Use Case 3 — Backlog Management

Backlogs rot fast. Items lose context, priorities drift, duplicates pile up.

The Problem

- Issues missing context or criteria
- Unclear priority — everything is "P1"
- Duplicates nobody catches
- Stale work that should be closed

AI Helps Me

- Summarize issues — what's this actually asking?
- Identify missing context — flag incomplete items
- Find duplicates — surface similar work
- Flag stale items — what hasn't moved in 90 days?

> Faster grooming. Cleaner prioritization. Engineers spend planning deciding, not deciphering.

█ §5 — Cheat Sheet & Resources

Everything you need in one place

🧑‍💻📋
Everyone

Quick Reference — CCA & Steering

CCA Issue Checklist

- Clear title & description
- File paths & component names
- Acceptance criteria (what = "done")
- Scope boundaries (which dirs/files)
- Related PRs or patterns to follow
- Single responsibility per issue

Start CCA From Anywhere

- Issues — assign Copilot as assignee
- Agents tab — in your repository
- Agents panel — top of any GitHub page
- VS Code / JetBrains — `/task` in Chat
- Copilot CLI — `copilot` in terminal
- GitHub Mobile — agents task page

Steering Commands

Intent	Comment
Redirect	"Use X pattern instead"
Scope	"Only modify these files"
Fix CI	"@copilot fix {job URL}"
Approve	"Looks good, merge"
Abort	"Close this, I'll handle it"

Continue a Session

- VS Code — click "Open in VS Code"
- Copilot CLI — `copilot --resume=ID`
- Codespaces — open from session view
- Steer live — type while CCA works, it adapts after current tool call

Quick Reference — Chat & Links

VS Code Slash Commands

Command	Description
/explain	Explain selected code
/fix	Propose a fix for problems
/tests	Generate unit tests
/fixTestFailure	Find and fix a failing test
/new	Scaffold a new project
/clear	Start a new chat session

Chat Participants

Participant	Context
@workspace	Your full project structure
@terminal	Terminal shell & output
@github	GitHub skills (search, etc.)
@vscode	VS Code commands & features

Chat Variables (type #)

Variable	What It Includes
#file	Current file content
#selection	Selected text
#function	Current function/method
#class	Current class
#project	Project context
#sym	Current symbol

Links

- About CCA — docs.github.com/en/copilot/concepts/agents/coding-agent/about-coding-agent
- Manage Agents — docs.github.com/en/copilot/how-tos/use-copilot-agents/manage-agents
- Mission Control — github.blog/changelog/2025-10-28-mission-control
- Agents Tab — github.blog/changelog/2026-01-26-agents-tab
- Agentic Workflows — github.github.com/gh-aw
- GitHub CLI — cli.github.com
- GitHub Skills — learn.github.com/skills

▌ Hands-On Lab

Let's put this into practice

🧑‍💻📋
Everyone

Hands-On Lab — Expand Your Team with Copilot

We'll walk through the "Expand your team with Copilot" skills exercise together. No coding environment needed — everything runs on GitHub.

> What you'll do (5 steps):

1. Enable CCA on your repository
2. Assign an issue to Copilot and watch the agent work
3. Collaborate — review the PR, leave feedback, steer Copilot
4. Customize your agent's workspace with custom instructions
5. Delegate in parallel — assign multiple issues at once

> Get started now:

Go to github.com/skills/expand-your-team-with-copilot and click "Copy Exercise" — Mona will set up your first lesson in ~20 seconds.

> Prefer a different lab? Browse all Copilot skills at learn.github.com/skills — pick any that match your level.

Thank You

Start small. Steer often. Ship faster.

AI is a thinking partner, not a replacement — for developers and managers.

David O'Regan — @Oregand Klaire Baek — @shakingbeef

Questions?