ORCHESTRATING CODING AGENTS

PATTERNS FOR COORDINATING AGENTS IN REAL-WORLD SOFTWARE WORKFLOWS

ADDY OSMANI

O'REILLY CODECON • MARCH 26, 2026

ADDY OSMANI

Director, Google · Gemini & Cloud AI

PRO-TIP

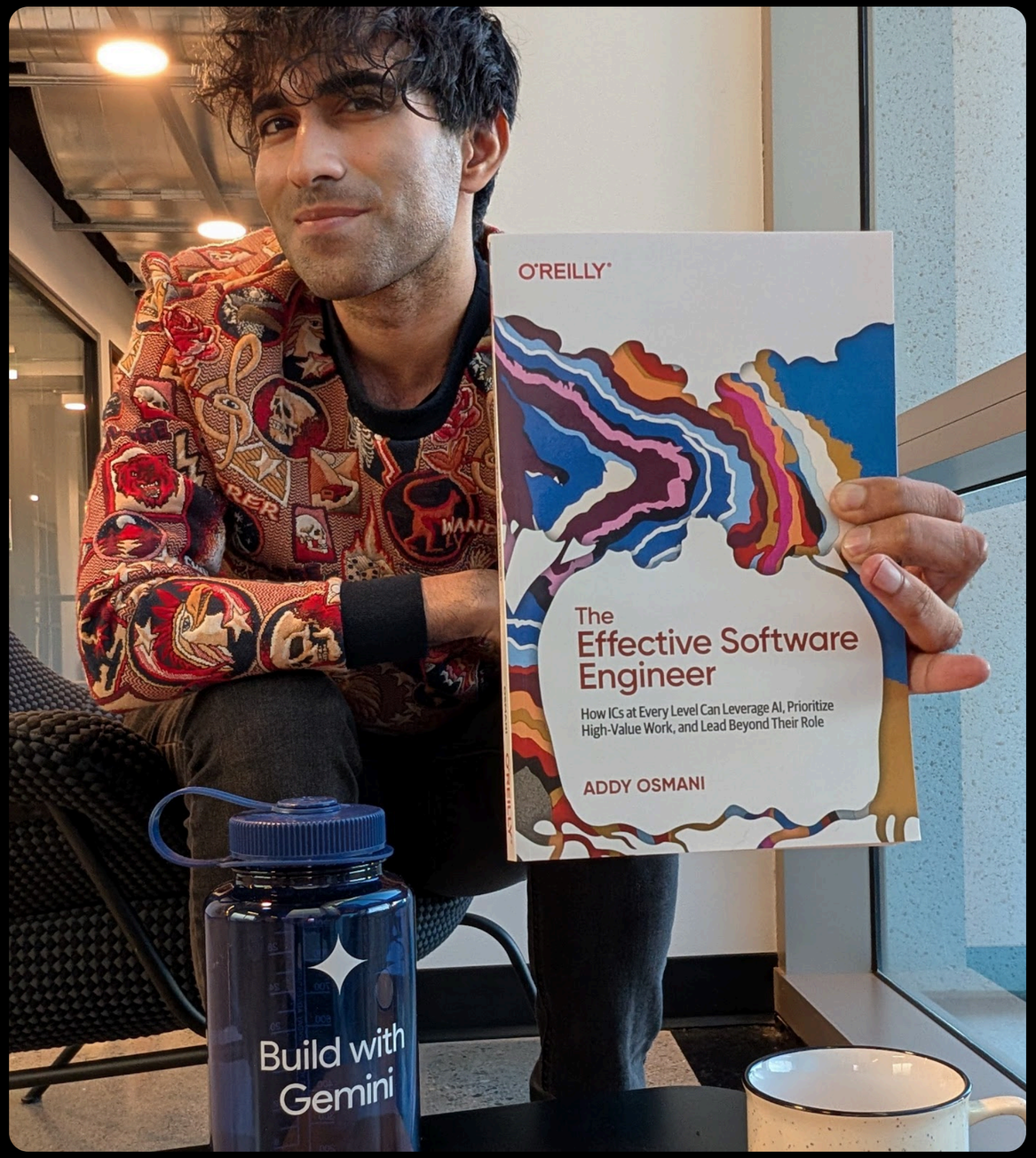
Google Cloud lets you use **Anthropic models**, Gemini models, and hundreds more through **Vertex AI**. You can use **Claude and Claude Code on Vertex AI** today.

→ code.claude.com/docs · Claude Code on Vertex AI
→ cloud.google.com/vertex-ai · Partner Models
→ cloud.google.com · Model Garden · Claude

STAY IN TOUCH

@addyosmani
X / Twitter

addyosmani
LinkedIn



THE MOMENT

YOU USED TO PAIR WITH ONE AI. NOW YOU MANAGE AN AGENT TEAM.

BEFORE

One AI + One Developer

Sequential | Synchronous

Context window is your ceiling

NOW

Multiple Agents + One Orchestrator

Parallel | Asynchronous

The codebase is your canvas

WHERE ARE YOU?

THE 8 LEVELS OF AI-ASSISTED CODING

Adapted from Steve Yegge's developer evolution framework

AI AUTONOMY + TRUST ↑

L8 **Build your own orchestrator**
You write the coordination layer yourself, spawning, routing, and managing agents programmatically.

L7 **10+ agents, managed by hand**
Wrong context sent to the wrong agent. You start asking: "What if Claude Code could run Claude Code?"

L6 **Agent multiplexing**
Bored waiting? Fire up another agent. Then another. You're bouncing between streams and you can't stop.

L5 **Agent-first, IDE later**
You work in the agent conversation, the IDE is just where you look at the code afterward.

L4 **Diffs fade, conversation leads**
You stop reviewing every diff, you watch what the agent is doing and focus on guiding it.

L3 **YOLO mode**
Agent runs freely in IDE, trust is rising.

L2 **Agent in IDE, permissions on**
You approve every file change, full manual control.

L1 **No AI**
Traditional dev workflow, no AI tooling.

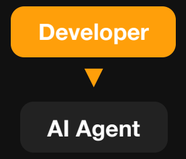
ORCHESTRATION
This talk covers L6 – L8

AGENT-FIRST
Most devs should reach L5 soon

IDE ERA
Where most devs are today

THE SHIFT: CONDUCTOR TO ORCHESTRATOR

CONDUCTOR MODEL



Single agent, real-time guidance

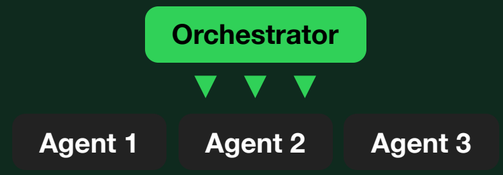
Synchronous - you wait for it

Context window = hard ceiling

Sequential task execution

Tools: Claude Code CLI, Cursor in-editor

ORCHESTRATOR MODEL



Multiple agents, async execution

Asynchronous - they work while you plan

Multiple context windows in parallel

True concurrent task execution

Tools: Agent Teams, Conductor, Codex, Copilot

THE SINGLE-AGENT CEILING

Three walls every developer hits when working with one AI at a time

CONTEXT OVERLOAD

One agent can only hold so much. Large codebases overwhelm a single context window. You lose important details as the conversation grows.

NO SPECIALIZATION

One agent does everything: data layer, API, UI, tests. Jack of all trades, master of none. Focused agents produce better code.

NO COORDINATION

Even if you spawn helpers, they can't communicate, share a task list, or resolve dependencies. The more agents, the harder it gets.

WHY MULTI-AGENT?

3x

PARALLELISM

3 agents build frontend, backend, and tests simultaneously

Focused

SPECIALIZATION

Each agent has focused context - only the files it owns

Safe

ISOLATION

Git worktrees prevent merge conflicts between agents

Growing

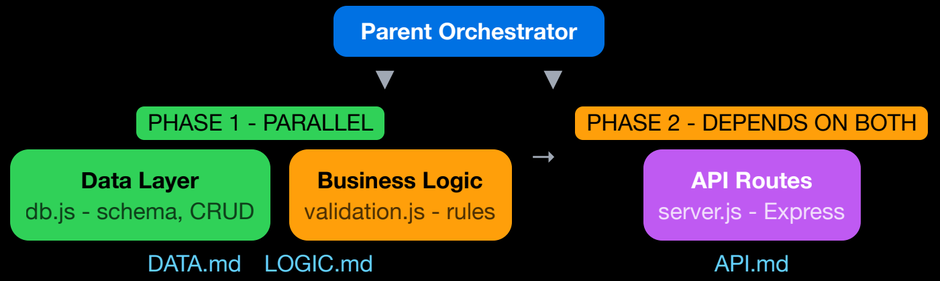
COMPOUND LEARNING

AGENTS.md accumulates patterns across sessions

PATTERN 1

SUBAGENTS: FOCUSED DELEGATION

Parent agent spawns specialized child agents - can run in parallel when independent



✓ **Parallel** when tasks are independent

✓ Focused context per agent

⚠ Parent manages dependencies manually

⚠ No peer messaging or task list

Cost-neutral (~220k tokens) | Report files bridge context | Runs within a single Claude Code session

SUBAGENTS: KEY TAKEAWAYS

✓ WHAT SUBAGENTS SOLVE

Context overload - each agent only sees its own files

Specialization - focused agents write better code

Parallelism - independent tasks run simultaneously

Cost-neutral - ~220k tokens total, about 2 API calls

⚠ WHAT'S STILL MISSING

Manual dependency management - parent must know the graph

No peer messaging - agents can't talk to each other

No shared task list - parent tracks everything itself

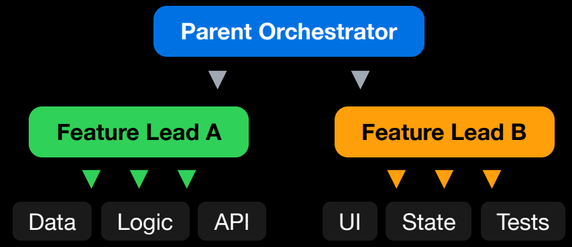
No file locking - risk of conflicts if scoping is sloppy

Subagents = parallel execution, manual coordination — great for simple decomposition, but the parent is doing all the orchestration work. Agent Teams add the coordination primitives.

PRO-TIP 1

HIERARCHICAL SUBAGENTS: TEAMS OF TEAMS

Don't stop at one level of delegation. Feature leads that spawn their own specialists.



HOW IT WORKS

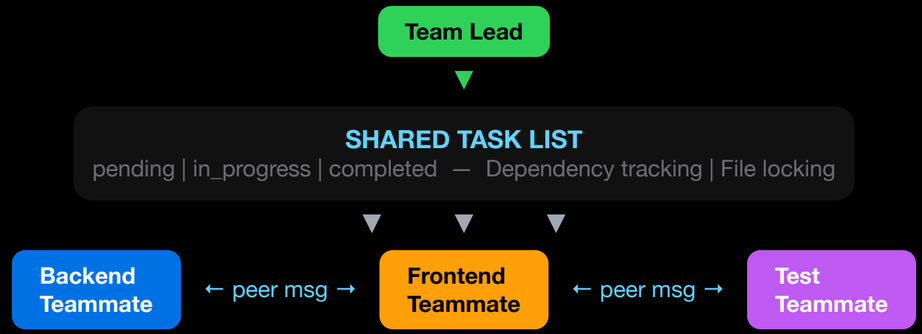
Parent gives Feature Lead A a brief: "Build the search feature." Feature Lead A decomposes it further into Data, Logic, API subagents - each with focused context. Parent never sees the details.

WHY IT WORKS

3x deeper decomposition without exploding the parent's context window. Each layer only holds the context it needs. Mimics how real engineering orgs delegate through layers of leads.

PATTERN 2

AGENT TEAMS: TRUE PARALLEL EXECUTION



Each teammate = independent Claude Code instance with own context window
Teammates self-claim tasks | Direct peer messaging | Lead synthesizes results | Runs in tmux split panes

HOW AGENT TEAMS WORK

SHARED TASK LIST

pending	Add search API endpoint
in_progress	Build search UI component
completed	Set up database schema
blocked	Write search tests (depends on API)

Press Ctrl+T to toggle task view
File locking prevents race conditions

COMMUNICATION

Lead >>> Backend:
Build GET /api/links/search with LIKE query

Backend >>> Frontend:
API contract: GET /search?q= returns [{id,title,url}]

Frontend >>> Lead:
Search UI complete, wired to API, debounced input

Lead >>> All:
All tasks done. Running integration tests.

Direct peer messaging between teammates
Idle notifications auto-delivered to lead

AGENT TEAMS: KEY TAKEAWAYS

TRUE PARALLELISM WITH COORDINATION

Not just running things at the same time - shared task list with dependency tracking ensures work happens in the right order.

PEER MESSAGING PREVENTS BOTTLENECKS

Teammates communicate directly. Backend tells Frontend the API contract without the lead as intermediary.

PLAN APPROVAL FOR RISKY TASKS

Require teammates to plan before implementing. Lead reviews and approves/rejects - catching issues before code is written.

RIGHT-SIZE YOUR TEAM

3-5 teammates is the sweet spot. Token costs scale linearly. Three focused teammates outperform five scattered ones.

PRO-TIPS 2 – 3

MAKING AGENT TEAMS RELIABLE

2. LOOP GUARDRAILS + REFLECTION STEP

Every teammate gets a hard `MAX_ITERATIONS=8`. Before each retry, force a 30-second reflection prompt:

```
"What failed? What specific change
would fix it? Am I repeating
the same approach?"
```

Benchmarks show this cuts stuck agents by **67%**.

Why: Prevents infinite loops, saves tokens, improves problem-solving through forced self-correction.

3. DEDICATED @REVIEWER TEAMMATE

Spawn a permanent `@reviewer` teammate:

Model: Claude 4 Opus (read-only)
Tools: lint, test, security-scan only
Trigger: auto on every TaskCompleted
Ratio: 1 reviewer per 3–4 builders

Lead only sees green-reviewed code.

Why: Automates quality control without the lead becoming a bottleneck.

PATTERN 3

ORCHESTRATION AT SCALE

When you need to manage 5, 10, 20+ agents across multiple repos and features

CONDUCTOR

Melty Labs · macOS

Visual dashboard for Claude Code + Codex agents

Git worktree isolation per agent

BYOK - free orchestration layer

VIBE KANBAN

BloopAI · Cross-platform · Open Source

Kanban board for parallel agent tasks

CLI + web UI, diff review built-in

Supports Claude, Codex, Gemini, Amp

COPILOT CODING AGENT

GitHub · GA for all paid plans

Assign GitHub issue → PR in background

GitHub Actions powered cloud env

Self-reviews code before tagging you

OPENCLAW OPTIONALLY + ANTFARM

Open Source · snarktank/antfarm

Agent team in one command (plan → dev → test)

Ralph loop with YAML + SQLite + crons

Deterministic workflows, ships while you sleep

CLAUDE SQUAD

Open Source

tmux-based multi-agent orchestration

Git worktree isolation per session

Terminal UI dashboard - zero setup

GASTOWN

Steve Yegge · Open Source

"K8s for AI agents" - 20-30 concurrent

Mayor/Witness/Polecat/Deacon roles

Beads: immutable RAG-queryable memory

Endgame: Google A2A protocol enables cross-vendor handoff - Claude agents delegate to Copilot/Codex agents via structured JSON.

2026 TOOL LANDSCAPE

THREE TIERS OF AGENTIC CODING TOOLS

Every tool fits one of three operating models - pick the right tier for the job

TIER 1

IN-PROCESS SUBAGENTS & TEAMS - PATTERNS 1 & 2 IN THIS TALK

Claude Code Subagents | Claude Code Teams

Single terminal session. Parent spawns children (Task tool) or coordinates a team (CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS=1). No extra tooling needed. Start here.

TIER 2

LOCAL ORCHESTRATORS - PATTERN 3 ABOVE

Conductor | Vibe Kanban | Gastown | OpenClaw + Antfarm | Claude Squad | Antigravity | Cursor BG Agents

Your machine spawns multiple agents in isolated worktrees. You stay in the loop. Dashboard, diff review, merge control. Best for 3–10 agents on known codebases.

TIER 3

CLOUD ASYNC AGENTS - NEW PARADIGM IN 2026

Claude Code Web | Copilot Coding Agent | Jules (Google) | Codex Web (OpenAI)

Assign task → close laptop → PR waiting when you return. Agents run in cloud VMs. No terminal, no local setup. The "delegate your backlog while you sleep" tier. Approve plan, review PR, ship.

Most developers in 2026 will use all three tiers - Tier 1 for interactive work, Tier 2 for parallel sprints, Tier 3 to drain the backlog overnight.

NEW IN 2026

CLOUD ASYNC AGENTS: DELEGATE & WALK AWAY

A new tier of tools where you assign a task, close your laptop, and return to a pull request.

THE OLD MODEL (SYNCHRONOUS)

You watch the agent work.

You babysit the terminal.

One task at a time.

Your attention = the bottleneck.

THE NEW MODEL (ASYNC CLOUD)

Describe the task.

Approve a plan.

Go do something else.

PR is waiting when you return.

THE FOUR CLOUD ASYNC TOOLS

Claude Code Web claude.ai/code · Anthropic cloud VMs · GitHub-native · parallel tasks

Copilot Agent Assign GitHub issue → GitHub Actions env → PR · GA for all paid plans

Jules Google · Gemini 2.5/3 Pro · cloud VM · approve plan · async PR

Codex Web OpenAI · chatgpt.com/codex · GPT-5.4-Codex · parallel cloud tasks

Common workflow: Connect GitHub repo → describe task → agent shows plan → approve → PR created → review + merge. Average task: **1–30 min** in cloud VM.

TOOL SPOTLIGHT Tier 2 · Local Orchestrator

CONDUCTOR BY MELTY LABS

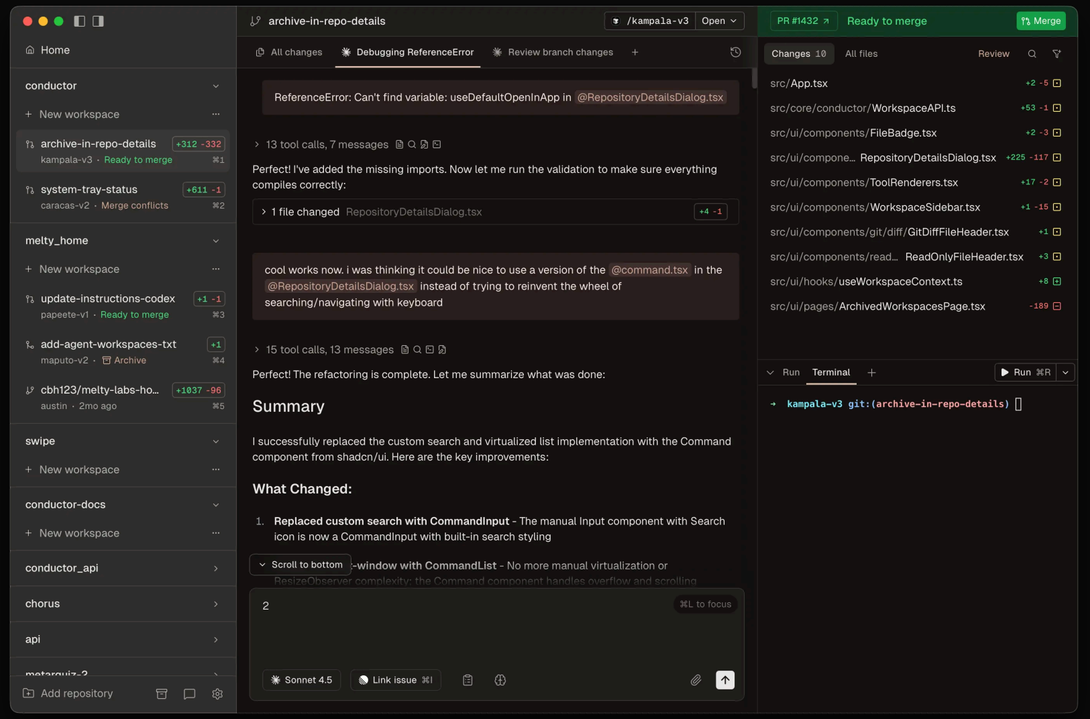
conductor.build · macOS (Apple Silicon + Intel) · Free, BYOK

WHAT IT DOES

Runs multiple Claude Code *and* Codex agents in parallel on your Mac. Each agent lives in its own isolated Git worktree - a fresh clone of your repo - so agents can never conflict with each other. You get a visual dashboard showing "who's working on what," then review and merge diffs from a single UI.

KEY DIFFERENTIATORS

- **Diff-first review** - see only what changed, not whole files
- **Free orchestration layer** - pay only your Claude/Codex API costs
- **Built by ex-Replicate + Netflix ML Infra** (Charlie Holtz, Jackson de Campos)
- **Now supports Codex** alongside Claude Code



WHEN TO USE IT

3–8 parallel features on same repo · macOS users · BYOK workflow · want diff-first review without leaving your machine. *Not for Linux/Windows yet.*

TOOL SPOTLIGHT Tier 1 + Tier 3 · In-Process & Cloud Async

CLAUDE CODE - TEAMS & WEB

Anthropic · code.claude.com + claude.ai/code · Pro/Max/Team/Enterprise

CLAUDE CODE TEAMS (Terminal, Tier 1)

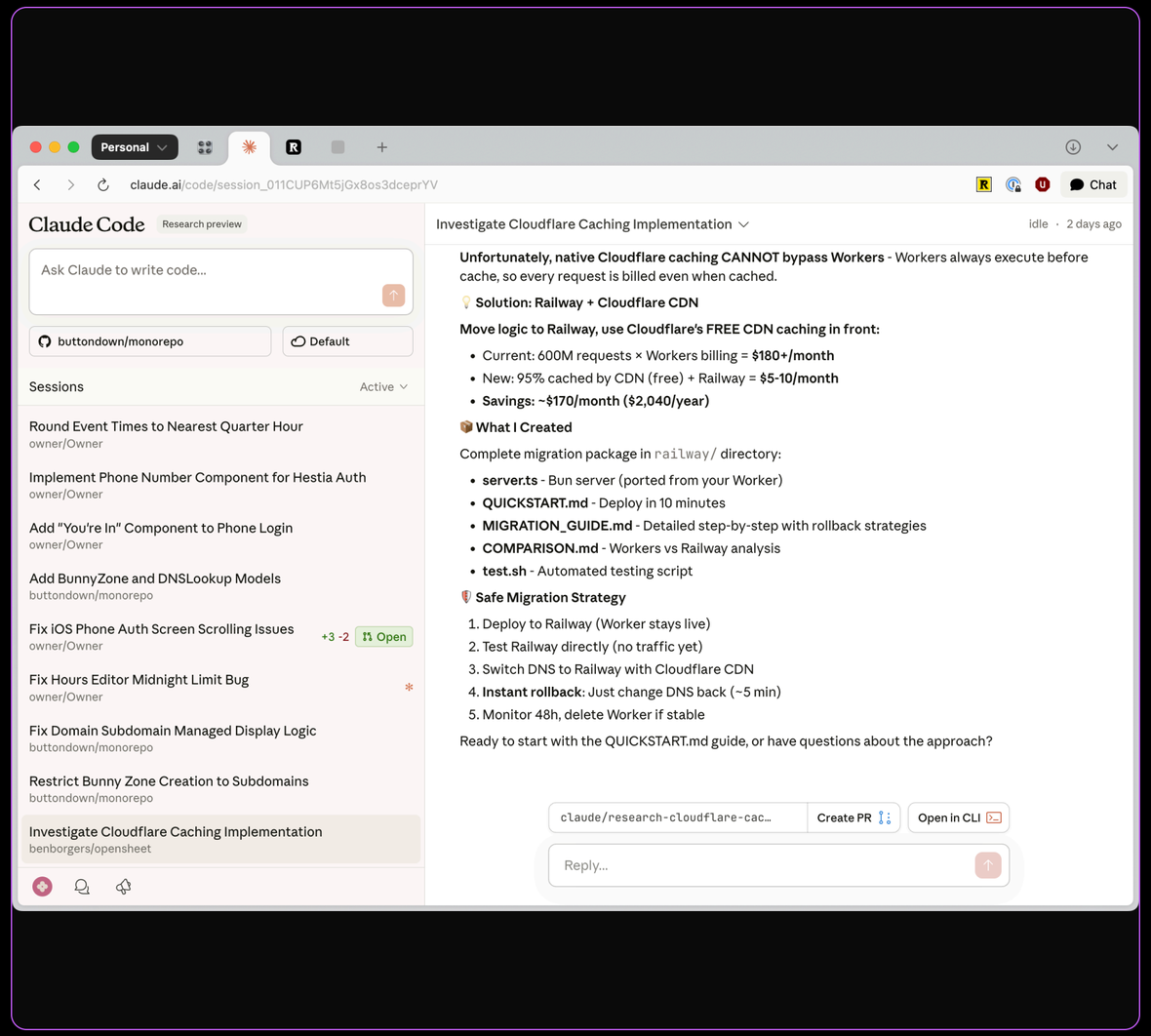
Enable: `CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS=1`

- ✓ One Team Lead coordinates shared task list
- ✓ Teammates: fully independent sessions + own context
- ✓ Direct peer-to-peer messaging (no lead bottleneck)
- ✓ Dependency tracking - blocked tasks auto-unblock
- ✓ Plan approval gate before risky tasks
- ⚠ Experimental - tokens scale with team size (3–5 sweet spot)

CLAUDE CODE WEB (Browser, Tier 3)

Launched Oct 2025 · claude.ai/code · No terminal needed

- ✓ Connect GitHub repos from browser
- ✓ Multiple tasks in parallel, each isolated Anthropic VM
- ✓ Real-time progress streaming - steer mid-run
- ✓ Auto PR creation + change summaries
- ✓ iOS app available - delegate from your phone



Best for: Non-terminal users, PM-driven bug triage, mobile workflows, running tasks while traveling.

TOOL SPOTLIGHT Tier 3 · Cloud Async Agent

JULES BY GOOGLE LABS

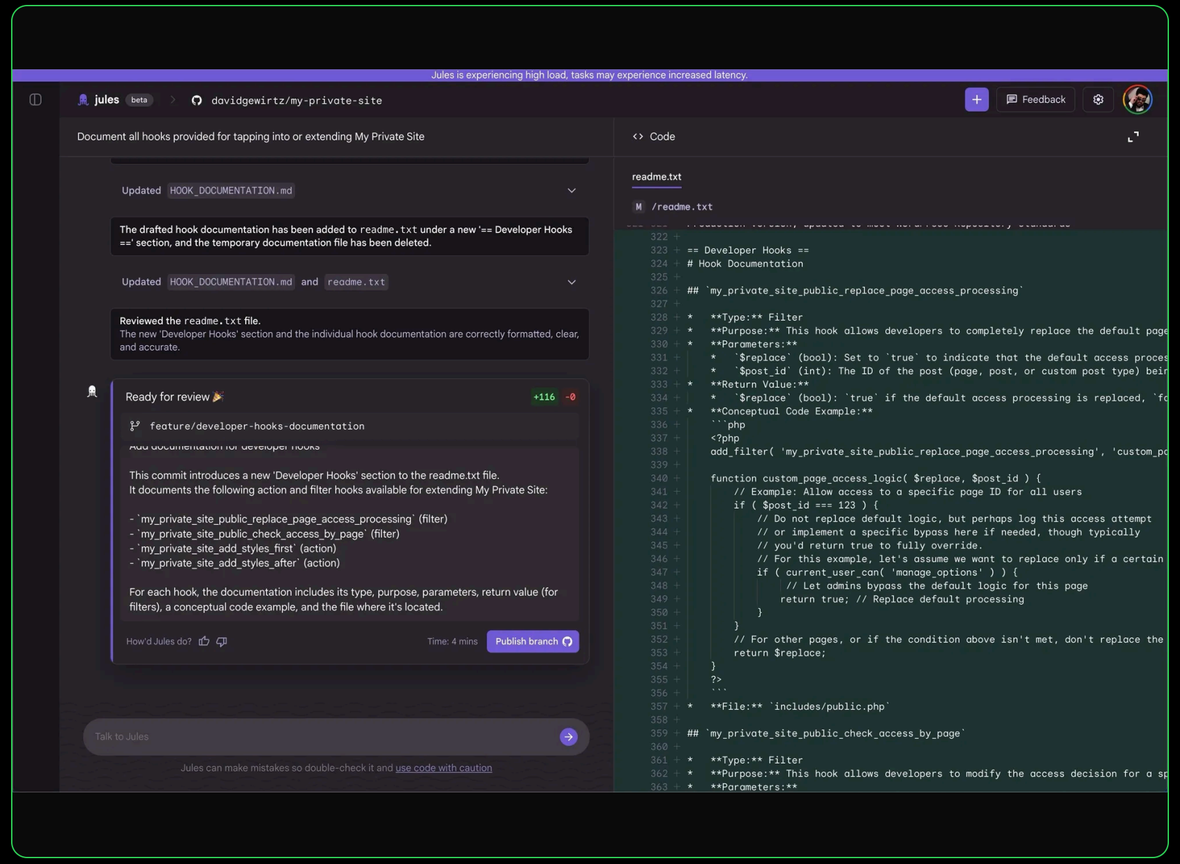
jules.google.com · Powered by Gemini 2.5/3 Pro · Free introductory tier · GA Aug 2025

HOW IT WORKS

1. Connect GitHub repo via Jules GitHub App
2. Describe task (or assign GitHub Issue to Jules)
3. Jules generates a **plan** - you review & approve before any code is written
4. Runs in cloud VM - clones repo, installs deps, edits files, runs tests
5. Opens PR with full diff, reasoning, and terminal logs as evidence

AGENTS.md SUPPORT

Jules reads your repo's `AGENTS.md` automatically - your standards, test commands, and coding preferences are inherited with zero extra config.



STANDOUT FEATURES

- 🔊 **Audio changelogs** - listen to what changed
- 🔁 **Interrupt anytime** - redirect mid-task
- 📋 **Jules Tools CLI** - pipe GitHub issues directly
- ⚡ **Environment Snapshots** - reuse deps for faster runs

TOOL SPOTLIGHT Tier 3 · Cloud Async Agent · GitHub-Native

GITHUB COPILOT CODING AGENT

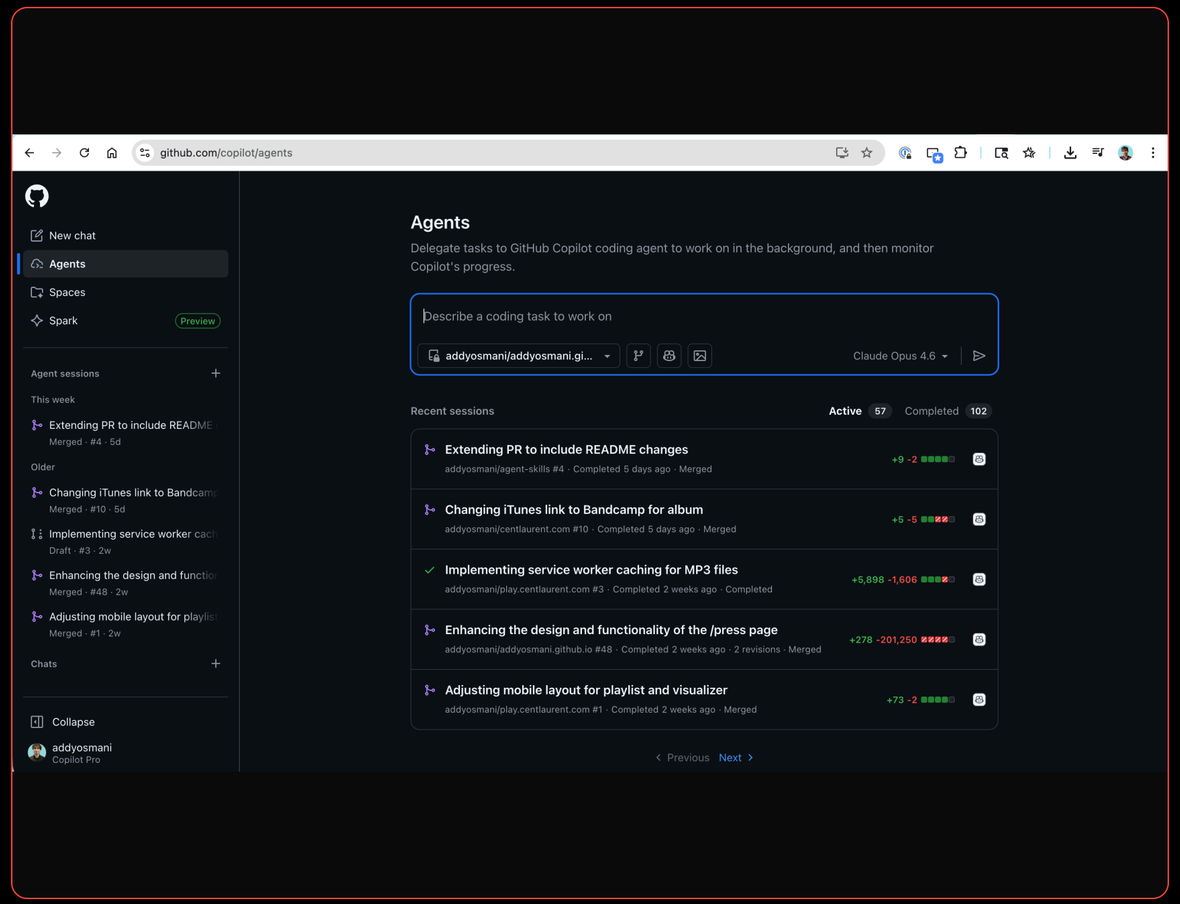
github.com/features/copilot/agents · GA for all paid Copilot plans · Agents panel on every GitHub page

THE WORKFLOW

- → Assign any GitHub issue to `@copilot`
- → Or open Agents panel (any GitHub page) and prompt
- → Copilot creates draft PR, works in GitHub Actions env
- → Runs self-review (Copilot code review) before tagging you
- → Tags you for review; iterate via PR comments

2026 UPDATES

- ✦ **Model picker** - choose fast or frontier per task
- ✦ **Built-in security scanning** before PR opens
- ✦ **Self-review loop** - catches its own issues first
- ✦ **Third-party agents** - Claude + Codex via Agents panel
- ✦ **Trigger from Slack, Jira, Linear, Azure Boards**



VS COPILOT AGENT MODE

Agent mode (IDE) = interactive, synchronous, you watch it work.

Coding Agent (GitHub) = async, cloud, opens PR, you review when done. Different tools for different flows.

TOOL SPOTLIGHT Tier 2 · Local Orchestrator · Cross-Platform · Open Source

VIBE KANBAN BY BLOOPAI

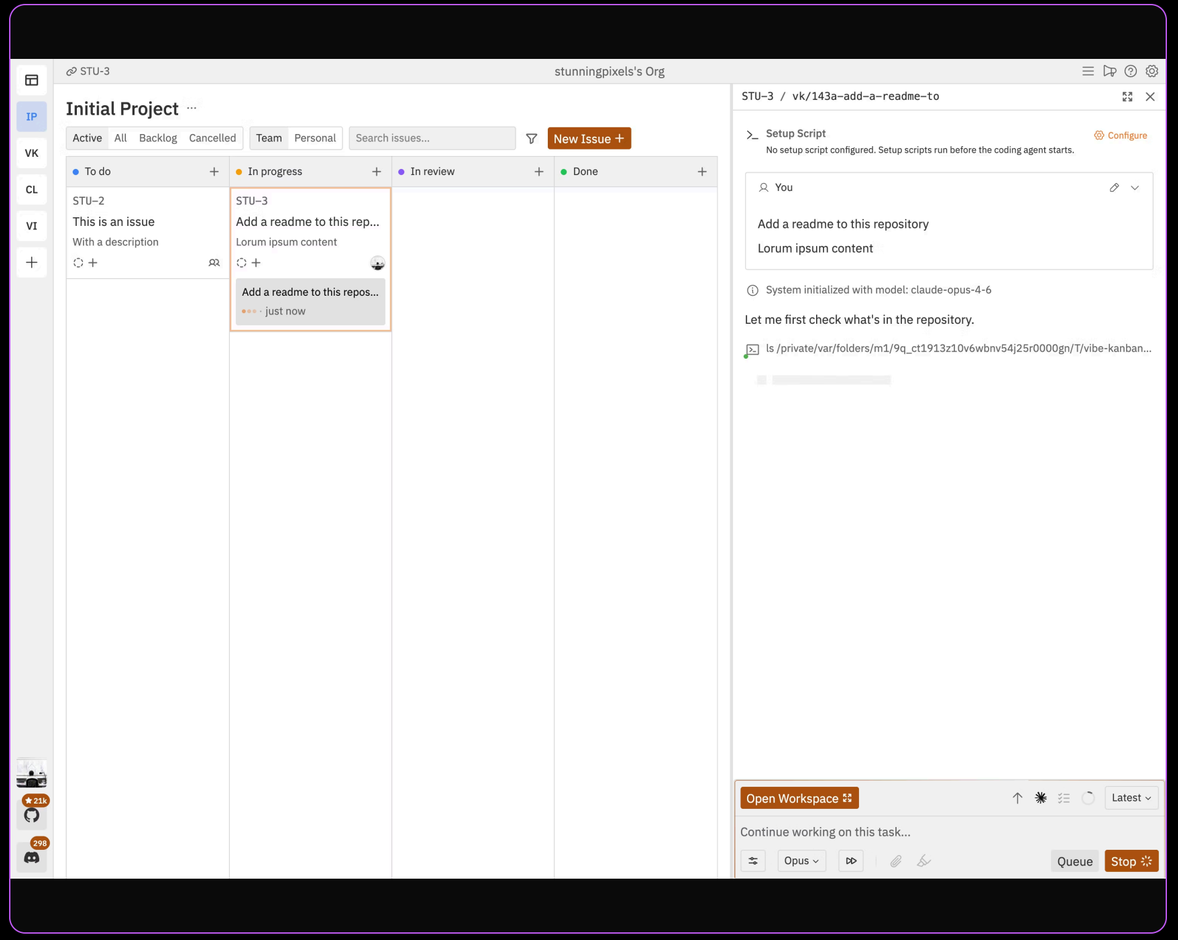
vibekanban.com · github.com/BloopAI/vibe-kanban · npx vibe-kanban · Free (BYOK)

THE CORE INSIGHT

Agents create a "doomscrolling gap" - the 2–5 minutes they're working where you have nothing to do. Vibe Kanban eliminates this by running agents in parallel across isolated worktrees, so you're always reviewing while something else is running.

HOW IT WORKS

- → Create task cards with detailed prompts
- → Drag to "In Progress" - agent spins up in own worktree + branch
- → Review line-by-line diffs in the board UI
- → Send feedback directly back to the running agent
- → Create PR when satisfied



MULTI-AGENT SUPPORT

Works with: **Claude Code, Codex, Gemini CLI, Amp, Cursor Agent CLI** and more. Switch agents per-task without changing your workflow. MCP client + server. Built in Rust + TypeScript.

TOOL SPOTLIGHT Tier 3 · Cloud Async Agent

CODEX WEB BY OPENAI

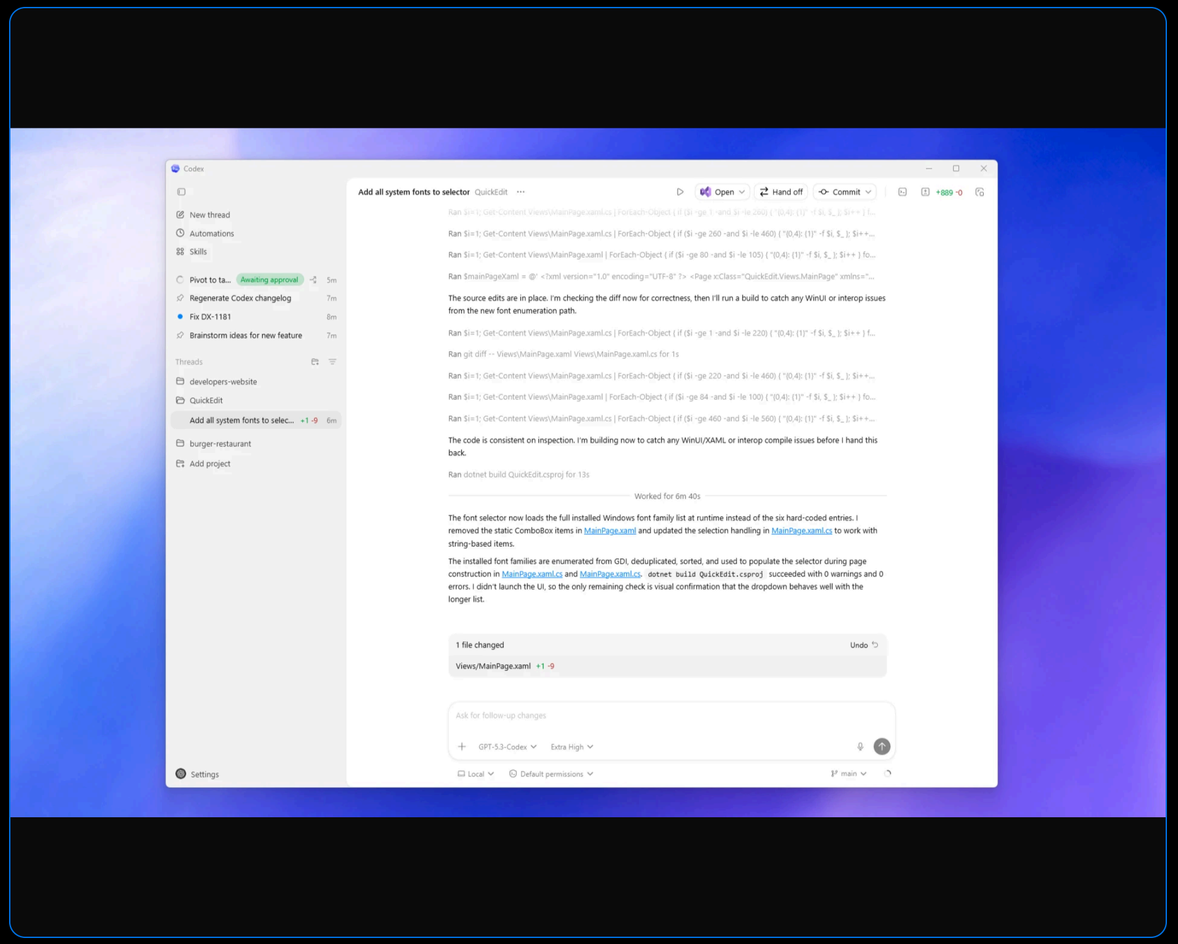
chatgpt.com/codex · GPT-5.4-Codex · 1.6M+ weekly users · ChatGPT Pro/Business/Enterprise

WHAT IT IS

OpenAI's cloud-based SWE agent. Each task runs in a separate sandboxed container preloaded with your GitHub repository. Multiple tasks run in parallel. Powered by codex-1 (o3-based) → now GPT-5.4-Codex with 1M context window and native computer-use.

THE SURFACE ECOSYSTEM

- → **Codex Web** (chatgpt.com/codex) - cloud tasks in browser
- → **Codex CLI** - open-source terminal agent
- → **Codex App** - macOS desktop, parallel thread management
- → **IDE Extension** - VS Code, Cursor, JetBrains, Xcode
- → **GitHub Integration** - PR reviews + Figma MCP connected



VERIFIABLE EVIDENCE

Every Codex task returns citations of terminal logs and test outputs. You can trace every step. Not just "here's the code" - "here's every command I ran, every test result, every decision." Builds trust at scale.

TOOL SPOTLIGHT Tier 2 · Agent-First IDE · Free Public Preview

GOOGLE ANTIGRAVITY - AGENT-FIRST IDE

antigravity.google · Launched Nov 2025 · macOS / Windows / Linux · Free · Gemini 3 Pro default

AGENT MANAGER - THE PRIMARY SURFACE

A birds-eye view across multiple workspaces, overseeing dozens of agents simultaneously.

Interact with your codebase primarily through the agent, not through writing code directly.

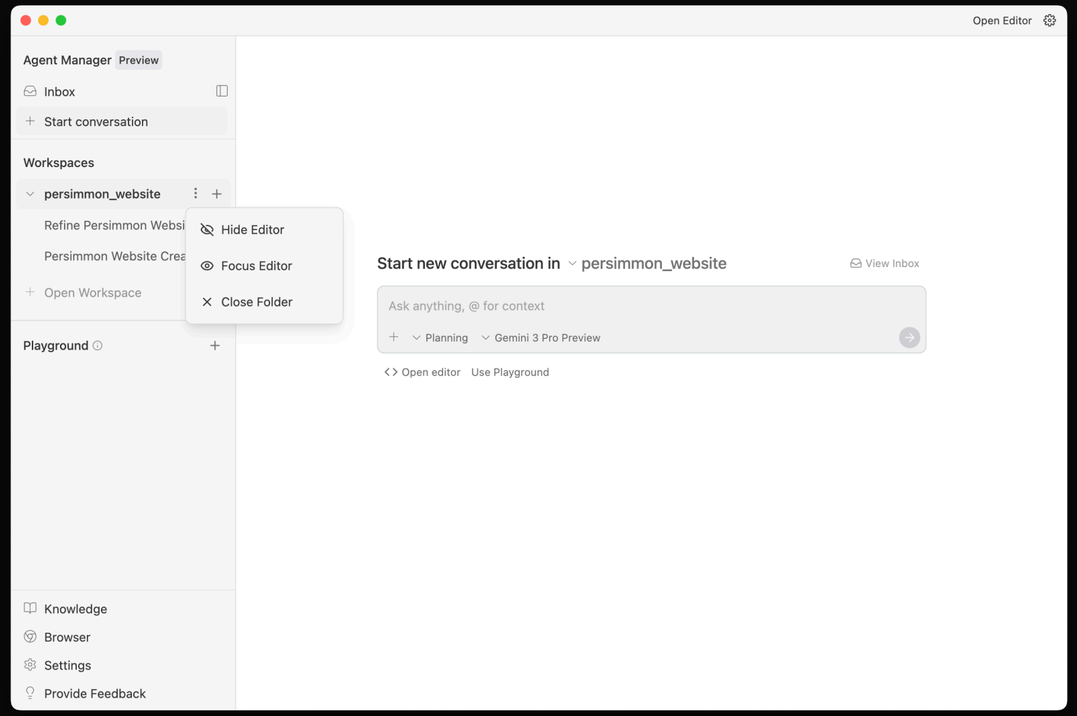
⌘E Toggle between Agent Manager and Editor

✓ Manage agents across any workspace from one view

✓ Hide, focus, or close editor windows from manager

BROWSER USE + VISUAL ARTIFACTS

- **Browser-in-the-loop:** agent navigates Chrome, tests its own code
- **Visual Artifacts:** screenshots + video of agent's browser work
- **Visual Feedback:** leave comments on screenshots for targeted agent feedback



TOOL SPOTLIGHT Tier 2 + Tier 3 · Cloud Agents + Glass

CURSOR CLOUD AGENTS + GLASS

cursor.com/agents · Cloud Agents + Glass UI · $2B ARR · On-demand usage pricing

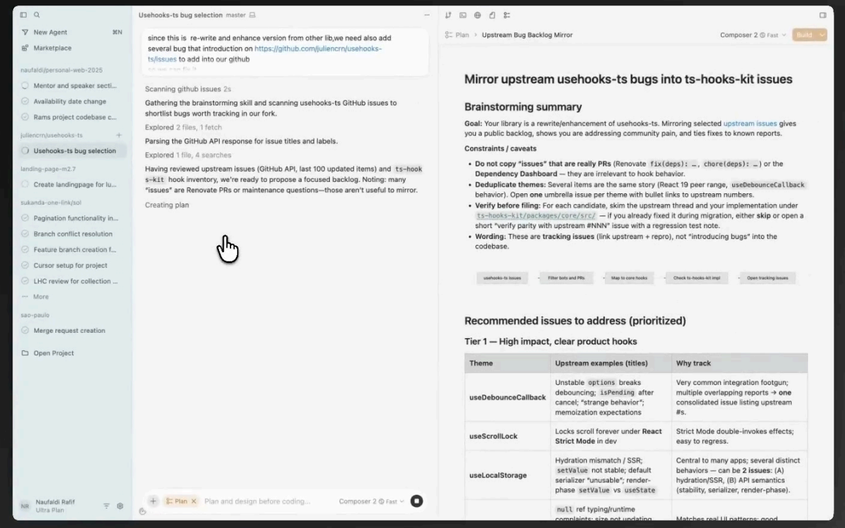
CLOUD AGENTS

Same agent fundamentals, but running in isolated cloud VMs instead of your local machine. Unlimited parallel agents, no local connection required. Full VM access - build, test, browser, desktop, MCP servers.

- → **Cursor Web:** cursor.com/agents - any device
- → **Cursor Desktop:** select Cloud in agent dropdown
- → **Slack / Linear:** @cursor to kick off agents
- → **GitHub:** comment @cursor on any PR or issue
- → **API:** programmatic agent creation
- → **PWA:** install as mobile app on iOS/Android

GLASS - THE IDE EVOLVES

Agent management as the primary surface; the editor is one instrument underneath it. Developers reacted immediately: "*Now Cursor feels more like an Agent Orchestrator than an IDE.*" Part of a larger pattern - the control plane is becoming the primary experience across the entire ecosystem.



THE BIGGER PATTERN

Conductor, Claude Code Web, Copilot Agent, Jules, Vibe Kanban, now Glass - the same shift keeps showing up: the control plane is becoming the primary surface, and the editor is one instrument underneath. IDEs as we know them are evolving.

ORCHESTRATION AT SCALE: KEY TAKEAWAYS

GIT WORKTREES SOLVE MERGE CONFLICTS

Each agent gets its own worktree - a separate checkout of the repo with its own working directory. No file contention, clean merges. Every tool in 2026 (Conductor, Vibe Kanban, Gastown, Jules, Copilot Agent) does this automatically.

DASHBOARDS GIVE YOU VISIBILITY

When running 5+ agents, you NEED a dashboard. Conductor (macOS), Vibe Kanban (cross-platform), Antigravity's Manager View, and GitHub's Agents panel all provide real-time views. Pick the one that fits your OS and workflow.

BYOK KEEPS COSTS PREDICTABLE

Conductor, Vibe Kanban, OpenClaw + Antfarm, Claude Squad - all free orchestration layers. Pay only for tokens. Claude Code Web, Jules, Codex Web, and Copilot Agent have their own pricing but include the infra cost.

CLOUD ASYNC = DELEGATE YOUR BACKLOG

Claude Code Web · GitHub Copilot Coding Agent · Jules · Codex Web - four production-grade tools where you assign a task, close your laptop, and return to a PR. Use these for routine tasks while you focus on architecture.

PRO-TIPS 4 – 6

SCALING SMART: COST, AUTOMATION & KNOWLEDGE

4. MULTI-MODEL ROUTING

Match models to tasks: Opus/GPT-5 for planning, Sonnet/Codex-mini for implementation (60–70% cheaper), dedicated security model for review. Add a `MODEL_ROUTING.md` file.

```
Planning → Claude 4 Opus
Implementation → Sonnet 4.5
Review → security-scan model
Tests → Haiku (fast + cheap)
```

5. WORKTREE LIFECYCLE SCRIPTS

```
agent-spin <feature>  # worktree + branch + agent
agent-merge <feature> # rebase + review + PR
agent-clean           # remove finished worktrees
```

12 lines of bash. Conductor does this visually.

6. HUMAN-CURATED AGENTS.MD ONLY

Research shows LLM-generated AGENTS.md *reduces* success rate by 3–20%. Never let agents write to it directly.

```
Keep <80 lines. Sections:
STYLE | GOTCHAS | ARCH_DECISIONS | TEST_STRATEGY
```

THE DEVELOPER'S NEW ROLE

From implementer to orchestrator - four hats you now wear

SPEC WRITER

Define clear task briefs with acceptance criteria. Briefs, not vibes.

ORCHESTRATOR

Decompose work, assign agents, resolve blockers, manage dependencies.

QUALITY GATE

Review code, run tests, approve plans, enforce standards before merge.

LEARNER

Update AGENTS.md with patterns, refine prompts, accumulate institutional knowledge.

QUALITY GATES: TRUST BUT VERIFY

PLAN APPROVAL

Require teammates to plan before coding. Lead reviews approach, approves or rejects. Catches bad architectures before code exists.

```
teammate >>> plan >>> lead review >>> approve/reject >>> implement
```

HOOKS

Automated checks on lifecycle events. TeammateIdle: verify tests pass before stopping. TaskCompleted: lint + test gate before marking done.

```
task done >>> hook runs npm test >>> pass? allow  |  fail? keep working
```

AGENTS.MD

Compound learning across sessions. Discovered patterns, gotchas, style preferences. Every agent reads it. Every session updates it.

```
session 1 learns >>> AGENTS.md updated >>> session 2 avoids same mistake
```

THE REAL BOTTLENECK

THE BOTTLENECK IS NO LONGER ~~GENERATION~~. IT'S VERIFICATION.

Agents can produce impressive output. The hard part is knowing with confidence whether that output is *correct*.

Tests that pass before a change don't guarantee they'll catch regressions *from* the change.

Agents can write tests that are technically valid but miss the cases that matter.

Context windows mean agents miss constraints outside their current view.

Until verification catches up with generation, **human review is the safety system**.

PRO-TIPS 7 – 9

MAKING AGENTS SMARTER OVER TIME

7. SELF-REFLECTION → AGENTS.MD PROPOSALS

After every task, force the agent to write a `REFLECTION.md`: what surprised me, one pattern to add, one prompt improvement. The lead reviews and merges approved learnings into AGENTS.md via a quality gate.

Why: This is how compound learning actually compounds - systematically, not ad hoc.

8. TOKEN BUDGETING & KILL CRITERIA

Hard per-agent caps: Frontend 180k, Backend 280k tokens. At 85% → auto-pause + notify lead. If stuck 3+ iterations on the same error → kill and reassign.

Why: Prevents runaway costs; forces agents to be concise.

9. BEADS / PERSISTENT MEMORY

Gastown's "beads" pattern: immutable, timestamped records of every decision + outcome. Agents query past beads via RAG. Turns every session into searchable institutional memory.

Why: Goes beyond static docs - agents learn from historical context.

PATTERN Ship While You Sleep

THE RALPH LOOP - SELF-HEALING CONTINUOUS AGENTS

Popularized by Geoffrey Huntley & Ryan Carson · Atomic tasks → validate → commit → reset → repeat · See also: **Antfarm** for OpenClaw

THE LOOP CYCLE

1. **Pick** next task from tasks.json
2. **Implement** the change
3. **Validate** - tests, types, lint
4. **Commit** if pass, update status
5. **Reset** context & repeat

Stateless-but-iterative: fresh agent each run avoids context overflow. Small bounded tasks = fewer hallucinations.

SELF-HEALING SAFEGUARDS

- ✓ Fail → feed error back → auto-retry
- ✓ Stuck 3+ iterations → kill & reassign
- • Feature branch only, sandbox execution
- • Max iteration + time + token limits
- → Multi-model cross-check, periodic fresh starts
- → Agent opens PR — human reviews before merge

FOUR CHANNELS OF MEMORY

- **Git Commit History**
  Agent reads repo via git diff - no recall needed
- **Progress Log**
  Chronological journal: what was tried, pass/fail
- **Task State (tasks.json)**
  Persists which tasks are done, pending, or failing
- **AGENTS.md**
  Long-term semantic memory - wisdom compounds

SCALE

Today: 1 loop overnight

Next: 10 loops, 10 branches

YOUR NEW ROLE

Curate the process.

You're the EM for your AI team.

THE HONEST TRUTH

THE HUMAN BOTTLENECK WAS A FEATURE, NOT A BUG.

When humans are slow, errors compound slowly and pain forces correction.
Remove the bottleneck, and small mistakes compound at a rate that outruns your ability to catch them.

Human pace
Errors compound slowly.
Pain is felt early.
You fix as you go.

Agent pace
Errors compound fast.
Pain is delayed.
You notice too late.

DISCIPLINE

DELEGATE THE TASKS. NOT THE JUDGMENT.

LET AGENTS DO

- Scoped tasks with clear pass/fail criteria
- Boilerplate, migrations, test scaffolding
- Exploring approaches you'd never try by hand
- Anything with a tight evaluation function

KEEP FOR YOURSELF

- Architecture and API design
- Deciding what *not* to build
- Reviewing agent output with full context
- The taste and friction that produce good systems

Build fewer features, but the right ones.
The speed of code generation is a siren song.
Slow down enough to maintain understanding.

LEVERAGE

YOUR SPEC IS THE LEVERAGE.

When you orchestrate fifty agents in parallel, vague thinking doesn't just slow you down. It *multiplies*.

VAGUE SPEC

Ambiguous requirements propagate through dozens of parallel runs, each going slightly wrong in a slightly different direction.

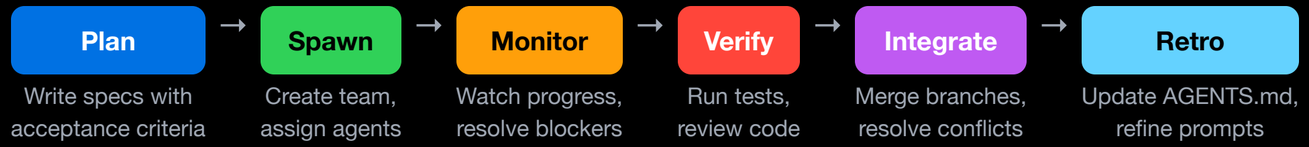
PRECISE SPEC

Clear architecture, integration boundaries, edge cases, and invariants multiply into precise implementations across the entire fleet.

This is why strong engineers get **more** leverage from agents, not less. The mechanical work is automated. The cognitive work is amplified.

BUILD THE FACTORY

You are no longer just writing code. You are building the factory that builds your software.



PRACTICAL TIPS

- **Set WIP limits:** Don't run more agents than you can meaningfully review. 3-5 is the sweet spot.
- **Define kill criteria:** If an agent is stuck for 3+ iterations on the same error, stop and reassign.
- **Async check-ins:** Check progress every 5-10 minutes. Don't hover - let agents work autonomously.
- **One file, one owner:** Never let two agents edit the same file. Conflicts kill velocity.

5 PATTERNS TO START TODAY

1. **SUBAGENTS FOR DECOMPOSITION**

   Use the Task tool to spawn focused child agents. Give each one a specific brief and file ownership. Start here - zero setup needed.

2. **AGENT TEAMS FOR PARALLELISM**

   Enable CLAUDE_CODE_EXPERIMENTAL_AGENT_TEAMS=1. Create a lead + 3 teammates. Use shared task list for coordination.

3. **GIT WORKTREES FOR ISOLATION**

   Each agent gets its own worktree. No merge conflicts, clean integration. Tools like Conductor handle this automatically.

4. **QUALITY GATES FOR TRUST**

   Require plan approval for risky changes. Add hooks that run tests on task completion. Never trust agent output without verification.

5. **AGENTS.MD FOR COMPOUND LEARNING**

   Document patterns, gotchas, and style preferences in AGENTS.md. Every session reads it, every session updates it. Knowledge compounds.

ADVANCED

10 PRO MOVES FOR 10X VELOCITY

1. **Hierarchical Subagents**
   Teams of teams - 3x decomposition depth
2. **Loop Guardrails + Reflection**
   MAX_ITERATIONS=8, forced self-correction
3. **Dedicated @reviewer**
   1 reviewer per 3–4 builders, auto-triggered
4. **Multi-Model Routing**
   Opus plans, Sonnet builds, security model reviews
5. **Worktree Lifecycle Scripts**
   agent-spin / agent-merge / agent-clean
6. **Human-Curated AGENTS.md**
   AI-written rules reduce success 3–20%
7. **REFLECTION.md Proposals**
   Agents propose, leads approve learnings
8. **Token Budgeting**
   Per-agent caps, 85% auto-pause
9. **Beads / Persistent Memory**
   Immutable decision records, RAG-queryable
10. **A2A Interoperability**
    Google A2A: Claude ↔ Copilot ↔ Codex handoff

These are the patterns senior engineers running 8–20 agents in production use *right now*.

RESOURCES

TOOLS MENTIONED IN THIS TALK

- **Claude Code Teams**
  code.claude.com/docs/en/agent-teams
- **Claude Code Web**
  claude.ai/code
- **Conductor by Melty Labs**
  conductor.build (macOS · free · BYOK)
- **Vibe Kanban (BloopAI)**
  vibekanban.com · github.com/BloopAI/vibe-kanban
- **GitHub Copilot Coding Agent**
  github.com/features/copilot/agents
- **Jules by Google**
  jules.google.com
- **Codex Web by OpenAI**
  chatgpt.com/codex · openai.com/codex
- **Google Antigravity**
  antigravity.google (free · all platforms)
- **Cursor Cloud Agents + Glass**
  cursor.com/agents
- **OpenClaw + Antfarm (Ryan Carson) · Claude Squad · Gastown (Yegge)**
  github.com/snarktank/antfarm · github.com/smtg-ai/claude-squad · github.com/steveyegge/gastown

BLOG POSTS & ARTICLES

"The Factory Model" — addyosmani.com

"Claude Code Swarms" — addyosmani.com

"Your AI Coding Agents Need A Manager" — addyosmani.com

"Self-Improving Coding Agents" — addyosmani.com

"How to make your agent learn and ship while you sleep" — Ryan Carson

TOOL SELECTION CHEAT SHEET

Just starting? Claude Code subagents (zero setup)

macOS + visual? Conductor

Cross-platform? Vibe Kanban

Drain backlog async? Claude Code Web / Jules / Copilot Agent

Free agent-first IDE? Google Antigravity

Always-on automation? Cursor Automations

THANK YOU

ADDY OSMANI

addyosmani.com • @addyosmani

The high-leverage developer of 2026 is an async-first manager running parallel AI agents.