Claude Code Advanced Patterns: Subagents, MCP, and Scaling to Real Codebases

March 24, 2026

ANTHROP\C

Housekeeping

A recording of this session will be distributed via email within 24 hours

Questions can be submitted at any time using the Q&A tab in the webinar portal

Give us feedback!
Rate this webinar by completing our survey that will be shared near the end of the webinar

Learning Outcomes

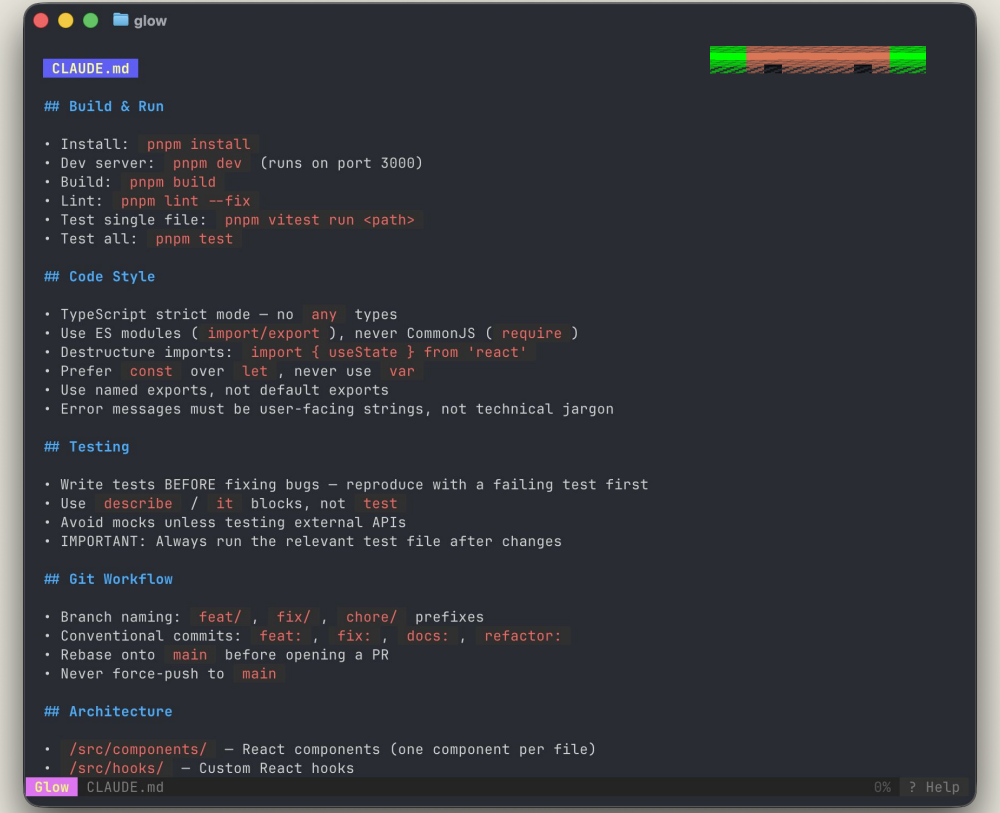
1. Control Claude's behavior with CLAUDE.md and Hooks
2. Parallelize Claude for major productivity gains
3. Embed Claude Code across your SDLC, from feature research to CI/CD
4. Form a mental model on when tool creation is worth the cycles
5. Imagine what an advanced Claude Code implementation could look like for your organization

Customizing Claude

ANTHROP\C

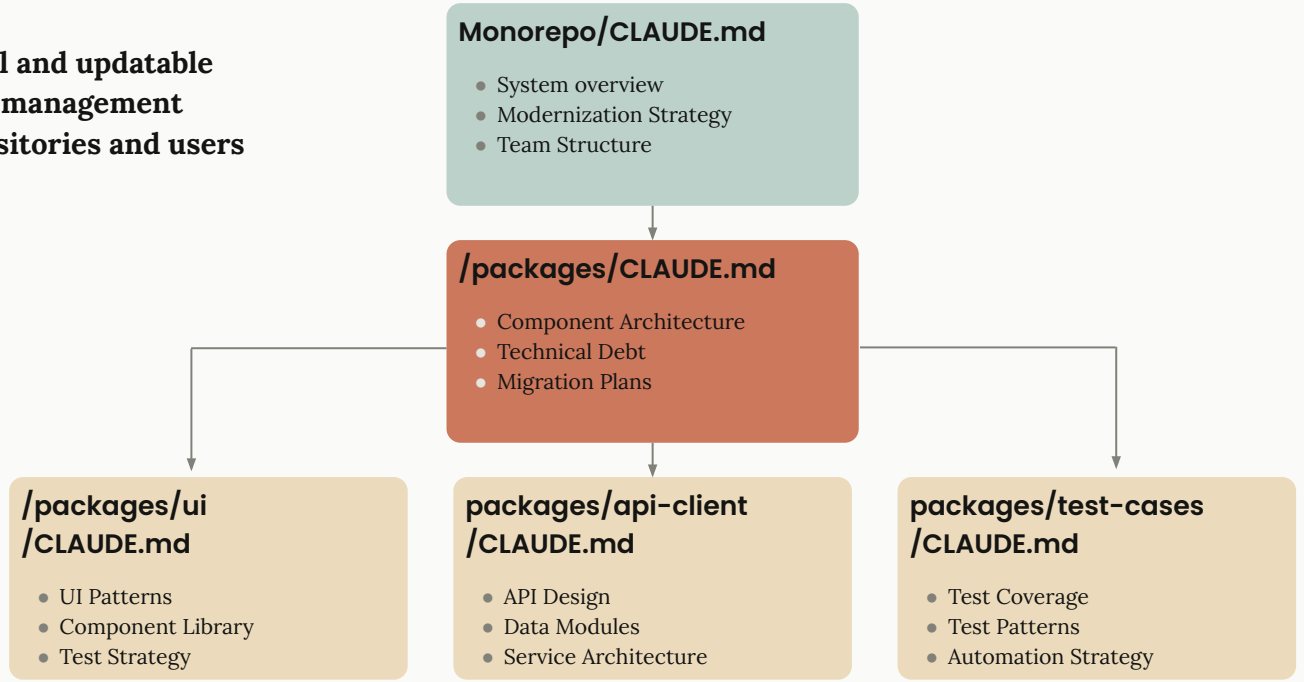
CLAUDE.md

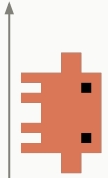
Similar to a README, but for Claude
Forced README file to give Claude instructions on project structure, common commands, personal styling tips, etc



CLAUDE.md for large codebases

Hierarchical and updatable instruction management across repositories and users





Claude walks up your directory tree to discover CLAUDE.md files

CLAUDE.md Best Practices

How should you structure CLAUDE.md files in large codebases?

Keep files < 200 lines

Longer files consume more context and can negatively affect instruction adherence.

.claude/rules/

Organize instructions into multiple files using the **.claude/rules/** directory.

Rules can also be scoped to specific file paths by providing a “paths” field in the frontmatter.

claudeMdExcludes setting

Exclude CLAUDE.md files irrelevant to your project to prevent contradictory instructions.

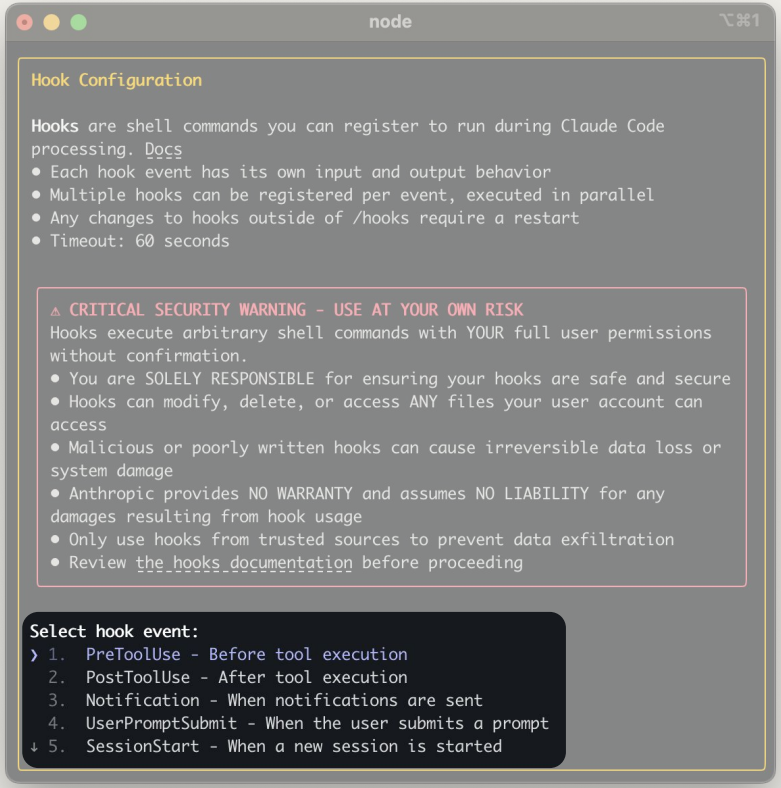
Managed policy CLAUDE.md files cannot be excluded.

Hooks

/hooks
Customize and extend Claude Code behavior by registering shell commands at various points in Claude's lifecycle.

Example Usage

- Notifications: Customize how you get notified.
- Automatic formatting
- Logging
- Correcting Claude behavior



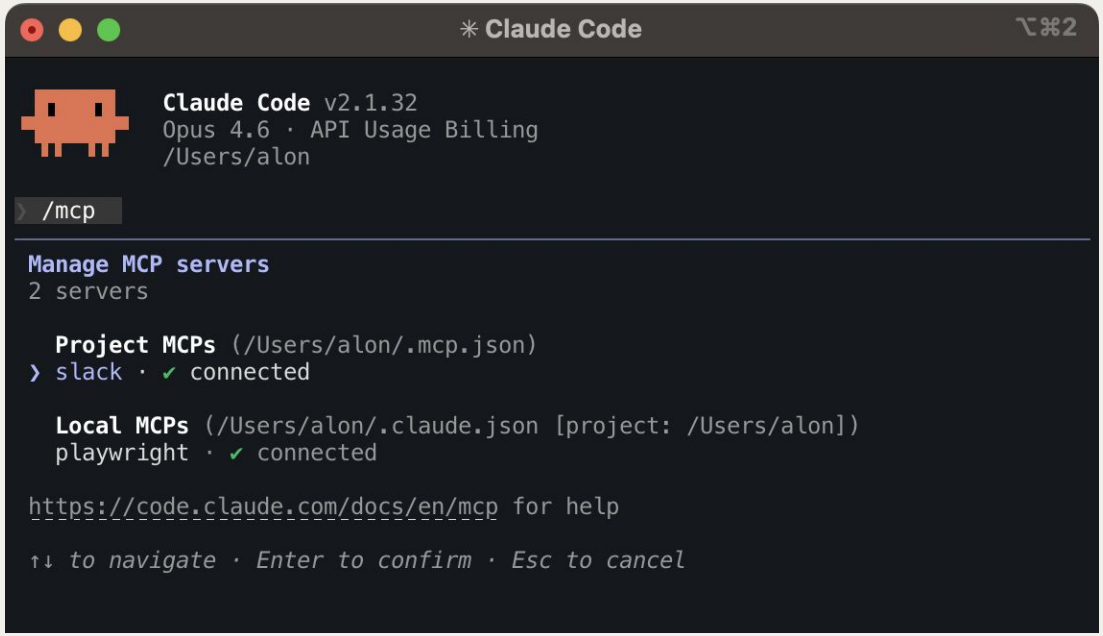
Customize with MCP

A standard protocol for giving Claude access to external systems: databases, ticket trackers, browsers, internal APIs

Add a server
Using mcp add <server-name>

Example Usage

- **Analyze monitoring data:** "Check Sentry and Statsig to understand the usage of the feature described in ENG-4521."



Ideal Use Case: Claude needs to ***reason over external state***, such as fetching Figma designs or creating tickets, without relying on copy + paste.

When to use which feature – CLAUDE.md, Hooks, and MCP

Tools	CLAUDE.md	Hooks	MCP
Used when...	Project-related context & instructions to prevent repeating instructions. *e.g. "use pnpm, not npm. Run tests with pytest. Follow PEP8."*	Deterministic automation that must always run at specific lifecycle events. *e.g. auto-format on save, run tests after edits, send notifications on completion.*	Access to external tools, databases, and APIs through a standardized protocol. *e.g. query database, fetch from GitHub, send Slack messages, access Google Drive.*

Parallelizing Claude

ANTHROP\C

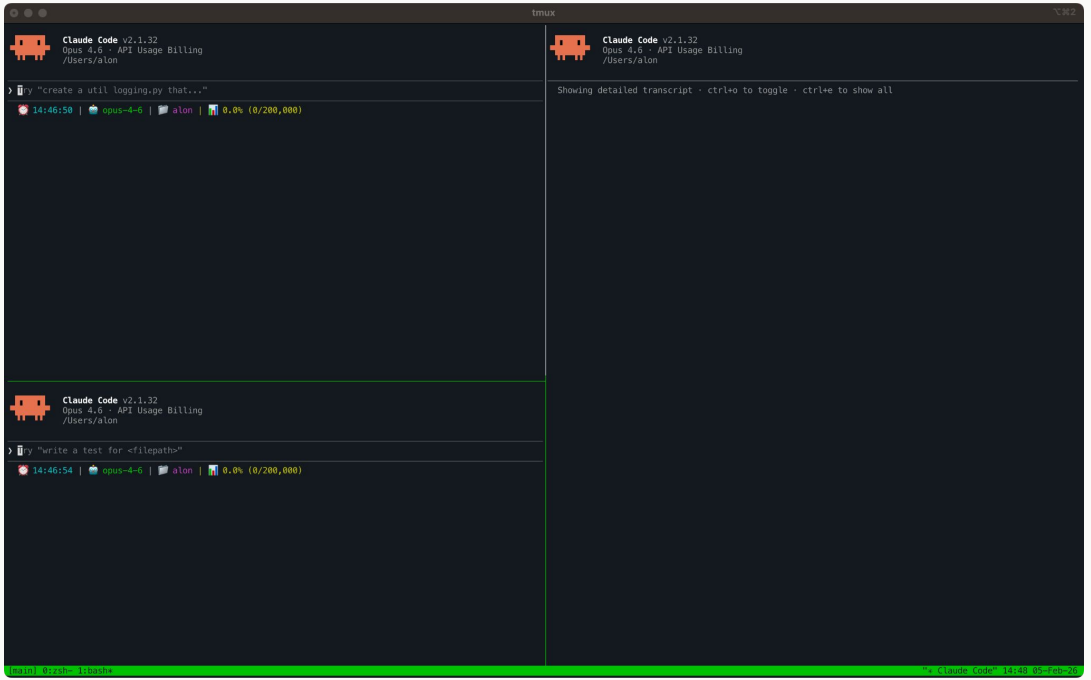
Parallel Claude

How does it work

- Run multiple Claude Code instances simultaneously, each in its own terminal
- Each instance works on a separate task independently with its own context

Git worktrees

- Create isolated working directories from the same repo: `claude --worktree`
- Each Claude instance operates on a different worktree, avoiding file conflict

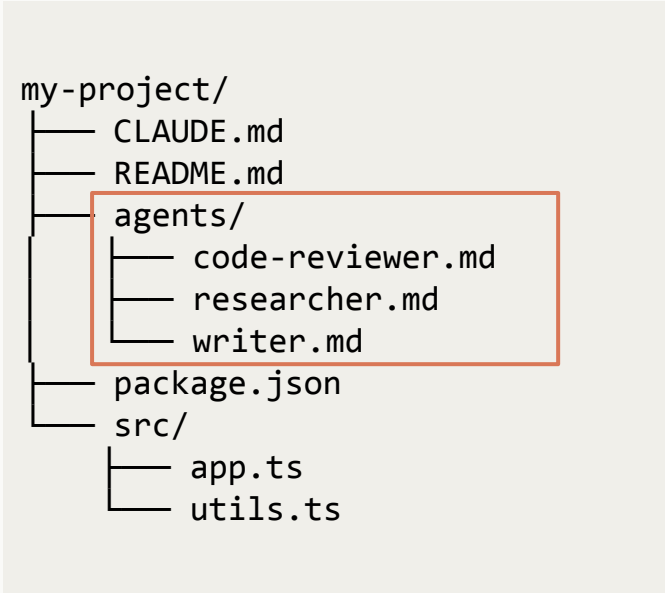


Subagents

1. **Well-defined roles**
   When you can give the subagent a clear and relatively specialized role with a defined **tool access levels**, and criteria for success or completion

2. **Hands-off delegation**
   When inspecting or interacting with the subagent's work is not a priority for you and only need a task to complete and return a few conclusions

3. **Enhance context management**
   Subagents can be a great strategy to manage your context, under the right conditions



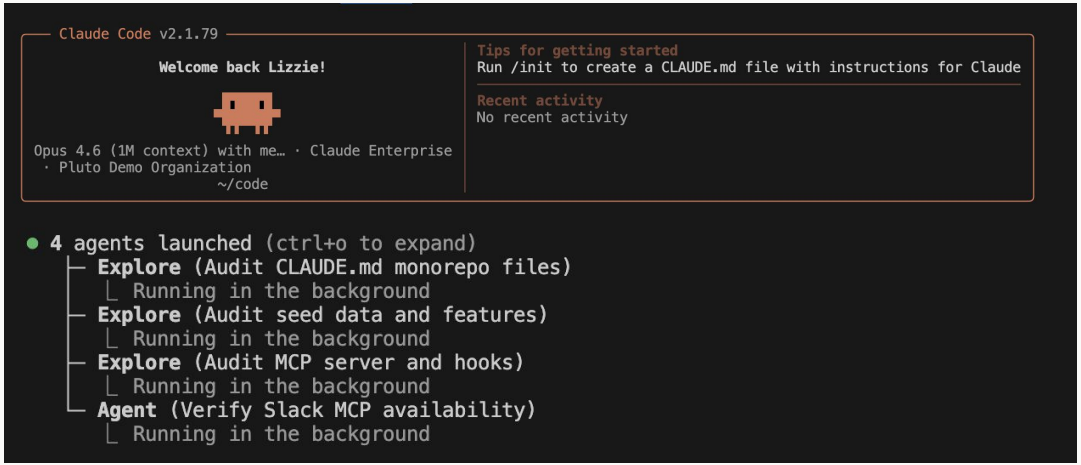
Ideal Use Case: Parallel experimentation or investigation of the codebase, only requiring a lighter amount of information to be returned to the main agent

Agent Teams

Orchestrate teams of Claude Code Sessions

Spin up multiple agents that communicate, coordinate, and divide-and-conquer parallelized work, now in research preview.

Agent teams shine when a task can be broken into independent workstreams, so each agent can own a slice without blocking others.



When to use which feature – Parallel, Subagents, and Agent Teams

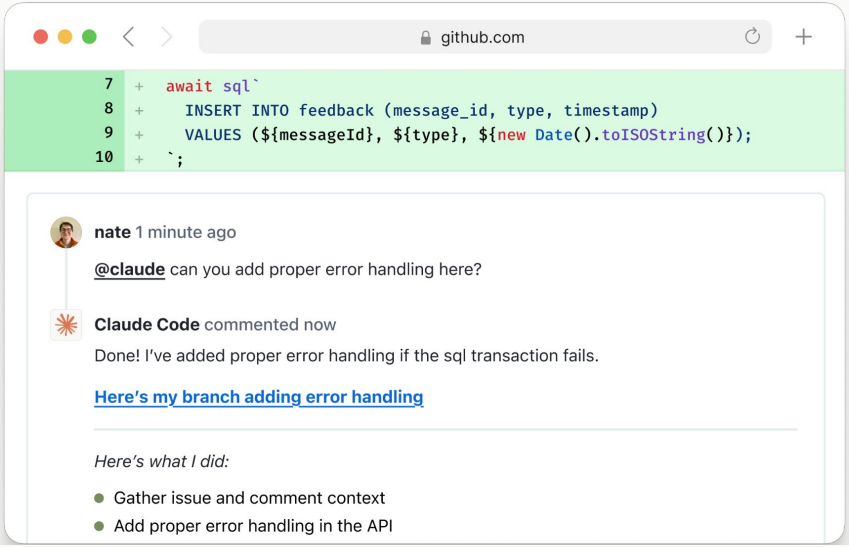
Tools	Parallel Claude	Subagents	Agent Teams
Used when... 	Working on multiple unrelated tasks at once, each in its own terminal and worktree. *e.g. fix a bug in one worktree while building a feature in another.*	Delegating focused subtasks from the main session with isolated context. *e.g. spawn a reviewer or researcher that returns a summary.*	Splitting a large task into independent workstreams that coordinate. *e.g. multi-service refactor where each agent owns a slice and syncs progress.*

Embedding Claude

ANTHROP\C

Claude Code in GitHub Actions - powered by the Claude Agent SDK

- Run Claude Code in GitHub Actions
- Remotely trigger edits to pull requests and issues by tagging @claude
- Easily add Claude to any part of your CI/CD process

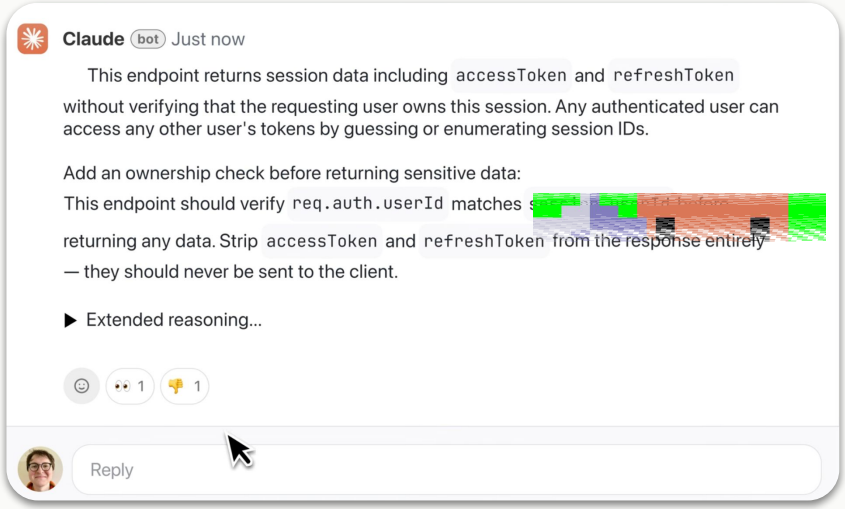


Getting started

- Run /install-github-app from within Claude Code
- Customize and deploy your own workflows using Claude Code

Code Review – thorough, agent team-based review system

- Specialized agents with context on your full codebase, capable of finding easy-to-miss edge cases and regressions
- Post inline comments on bugs, security gaps, and more, all rated by severity



Getting started

- **For admins:** Enable Code Review in your Claude Code settings, install the GitHub App, and opt in repositories.

Live Demo

Let’s tackle three time-consuming tasks simultaneously

1. Adding an @mention feature to our ticketing tool
2. Deciding on priorities and assigning tickets for my team’s next sprint
3. Getting through an initial code review for a major addition