

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

March 24, 2026

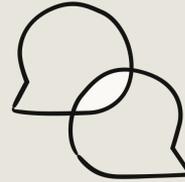
# Housekeeping



**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

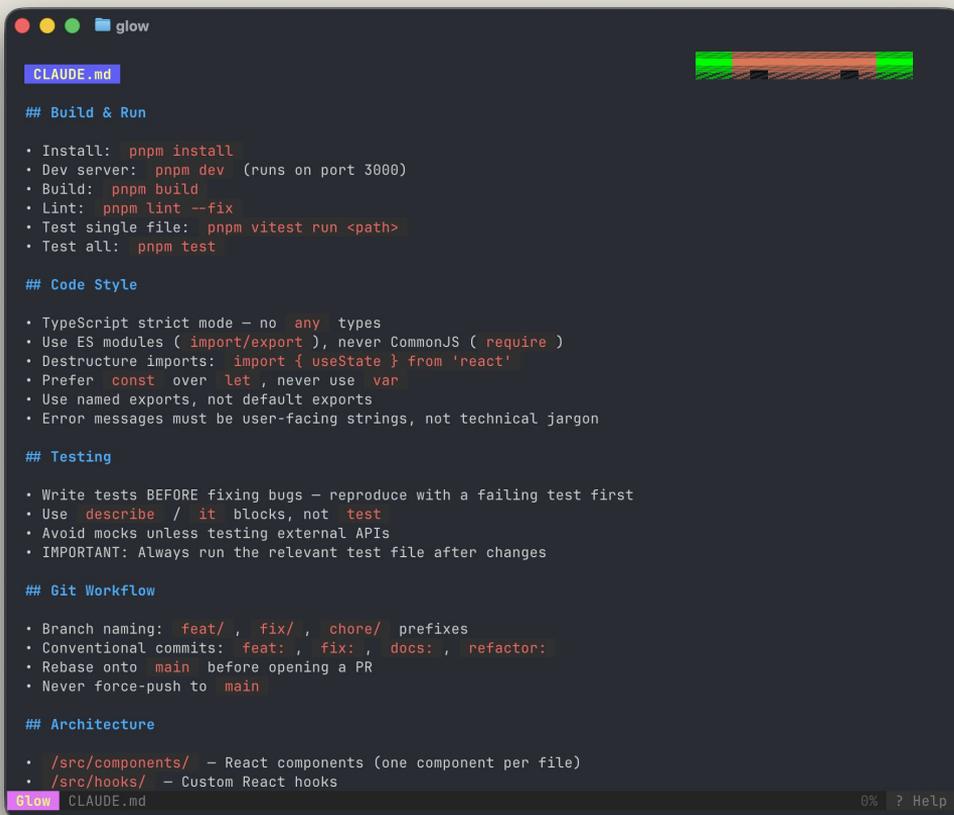
- 1.** Control Claude's behavior with CLAUDE.md and Hooks
- 2.** Parallelize Claude for major productivity gains
- 3.** Embed Claude Code across your SDLC, from feature research to CI/CD
- 4.** Form a mental model on when tool creation is worth the cycles
- 5.** Imagine what an advanced Claude Code implementation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

# Customizing Claude

ANTHROPIC

# 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

## Build & Run

• Install: pnpm install
• Dev server: pnpm dev (runs on port 3000)
• Build: pnpm build
• Lint: pnpm lint --fix
• Test single file: pnpm vitest run <path>
• Test all: pnpm test

## Code Style

• TypeScript strict mode – no any types
• Use ES modules ( import/export ), never CommonJS ( require )
• Destructure imports: import { useState } from 'react'
• Prefer const over let, never use var
• Use named exports, not default exports
• Error messages must be user-facing strings, not technical jargon

## Testing

• Write tests BEFORE fixing bugs – reproduce with a failing test first
• Use describe / it blocks, not test
• Avoid mocks unless testing external APIs
• IMPORTANT: Always run the relevant test file after changes

## Git Workflow

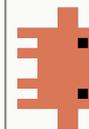
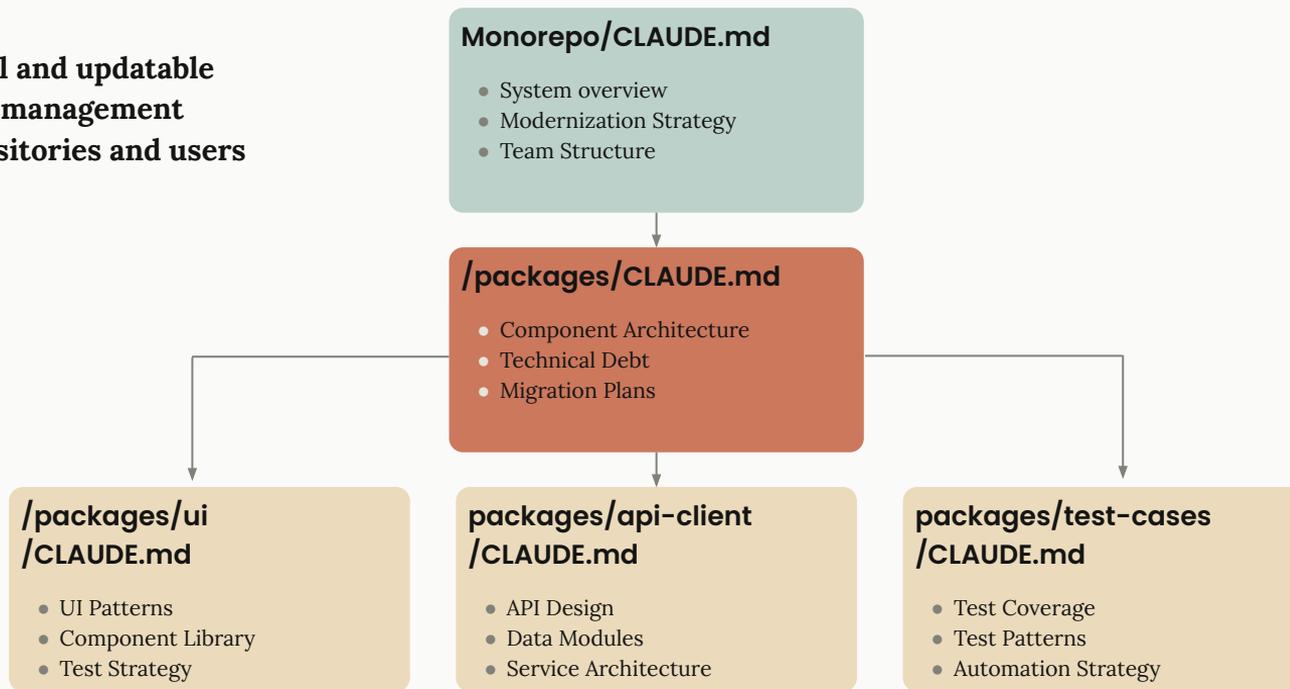
• Branch naming: feat/, fix/, chore/ prefixes
• Conventional commits: feat:, fix:, docs:, refactor:
• Rebase onto main before opening a PR
• Never force-push to main

## Architecture

• /src/components/ – React components (one component per file)
• /src/hooks/ – Custom React hooks
```

# 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.***

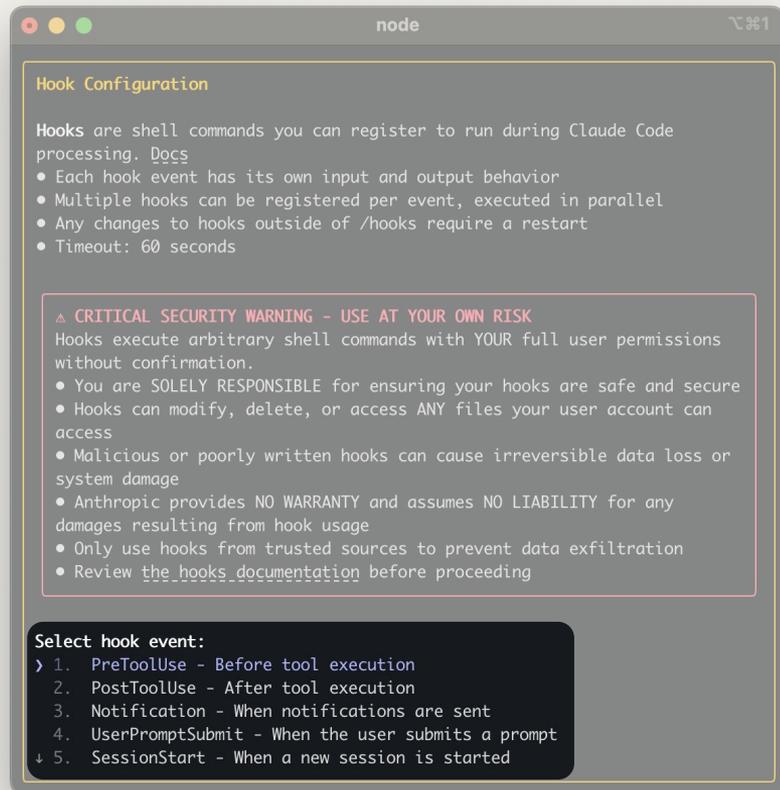
# Hooks

## `/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

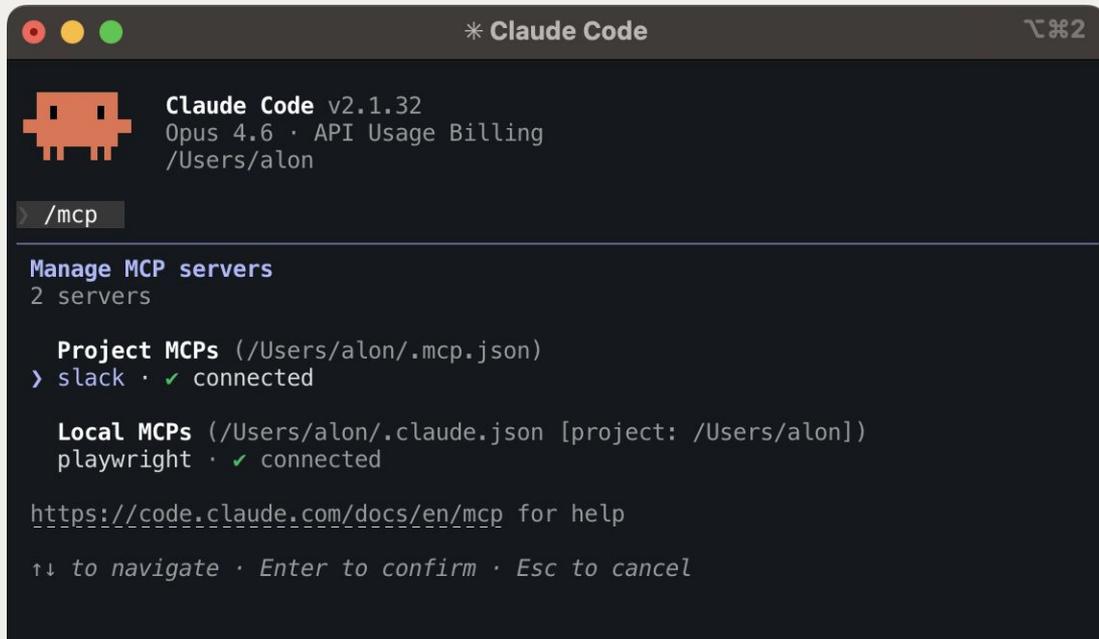
A standard protocol for giving Claude access to external systems: databases, ticket trackers, browsers, internal APIs

## 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.”



```
* Claude Code
Claude Code v2.1.32
Opus 4.6 · API Usage Billing
/Users/alon

/mcp

Manage MCP servers
2 servers

Project MCPs (/Users/alon/.mcp.json)
> slack · ✓ connected

Local MCPs (/Users/alon/.claude.json [project: /Users/alon])
playwright · ✓ connected

https://code.claude.com/docs/en/mcp for help
↑↓ to navigate · Enter to confirm · Esc to cancel
```

**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...	<p>Project-related context &amp; instructions to prevent repeating instructions.</p> <p><i>e.g. “use pnpm, not npm. Run tests with pytest. Follow PEP8.”</i></p>	<p>Deterministic automation that must always run at specific lifecycle events.</p> <p><i>e.g. auto-format on save, run tests after edits, send notifications on completion.</i></p>	<p>Access to external tools, databases, and APIs through a standardized protocol.</p> <p><i>e.g. query database, fetch from GitHub, send Slack messages, access Google Drive.</i></p>

# Parallelizing Claude

ANTHROPIC

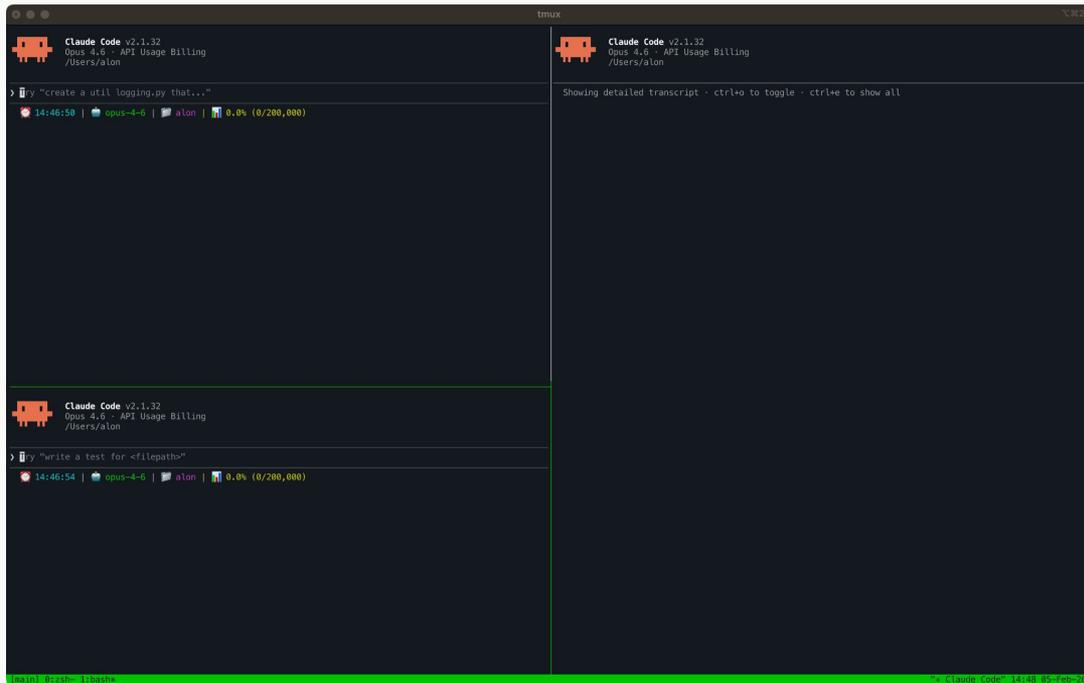
# Parallel Claude

## 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

```
my-project/  
├── CLAUDE.md  
├── README.md  
├── agents/  
│   ├── code-reviewer.md  
│   ├── researcher.md  
│   └── writer.md  
├── package.json  
└── src/  
    ├── app.ts  
    └── utils.ts
```

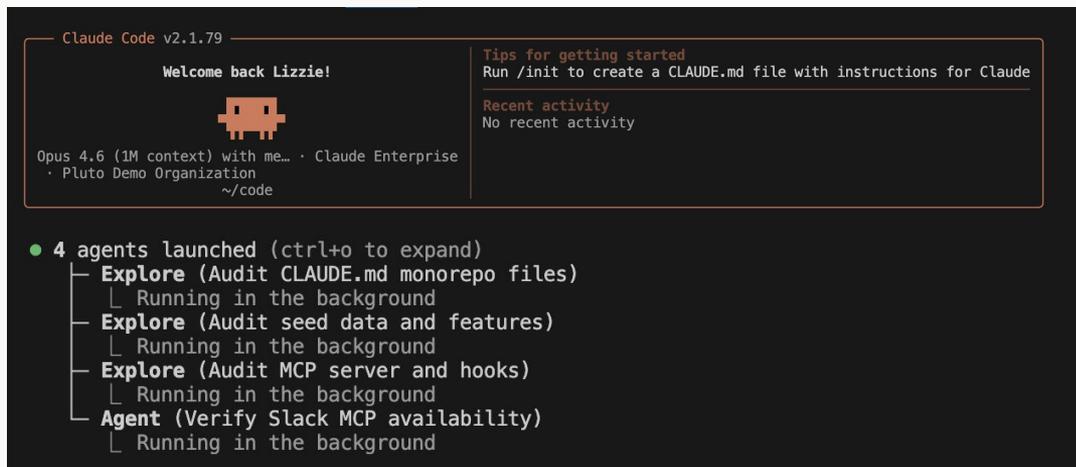
**Ideal Use Case:** Parallel experimentation or investigation of the codebase, only requiring a lighter amount of information to be returned to the main agent

# 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.



```
Claude Code v2.1.79
Welcome back Lizzie!
Opus 4.6 (1M context) with me... · Claude Enterprise
· Pluto Demo Organization
~/code

Tips for getting started
Run /init to create a CLAUDE.md file with instructions for Claude

Recent activity
No recent activity

● 4 agents launched (ctrl+o to expand)
├─ Explore (Audit CLAUDE.md monorepo files)
│   └─ Running in the background
├─ Explore (Audit seed data and features)
│   └─ Running in the background
├─ Explore (Audit MCP server and hooks)
│   └─ Running in the background
└─ Agent (Verify Slack MCP availability)
    └─ Running in the background
```

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

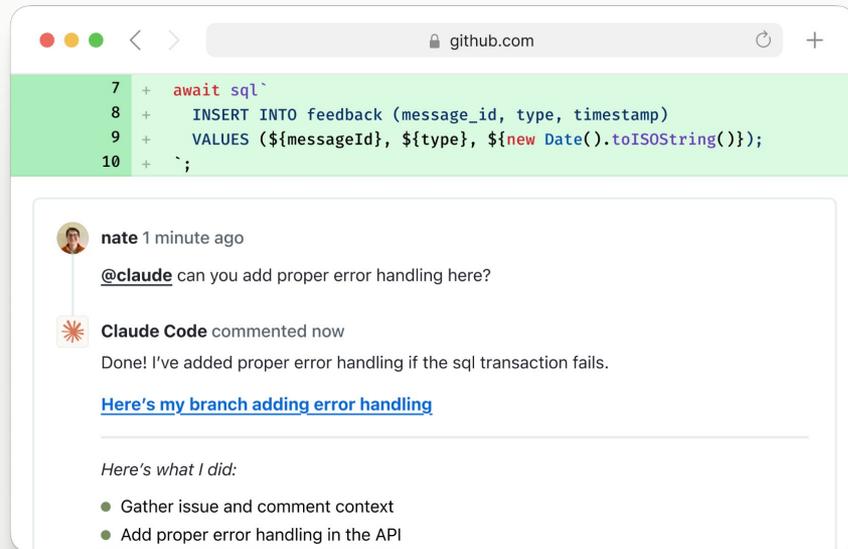
 Tools	Parallel Claude	Subagents	Agent Teams
<p data-bbox="324 543 548 576">Used when...</p> 	<p data-bbox="639 418 923 587">Working on multiple unrelated tasks at once, each in its own terminal and worktree.</p> <p data-bbox="639 634 913 765"><i>e.g. fix a bug in one worktree while building a feature in another.</i></p>	<p data-bbox="987 434 1306 568">Delegating focused subtasks from the main session with isolated context.</p> <p data-bbox="987 645 1271 743"><i>e.g. spawn a reviewer or researcher that returns a summary.</i></p>	<p data-bbox="1348 434 1628 568">Splitting a large task into independent workstreams that coordinate.</p> <p data-bbox="1348 645 1647 776"><i>e.g. multi-service refactor where each agent owns a slice and syncs progress.</i></p>

# Embedding Claude

ANTHROPIC

# 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

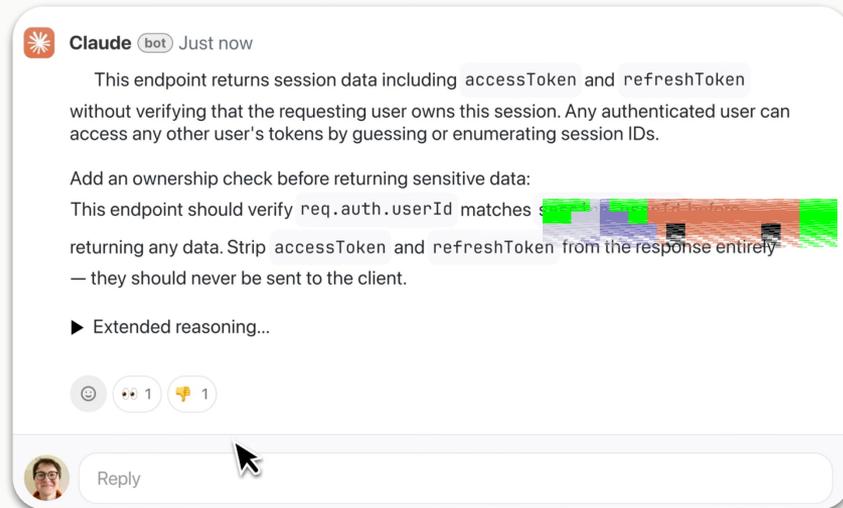


## Getting started

- 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

ANTHROPIC

# 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