Claude Code
Features Guide

Part 4 of the Claude Code Series

Claude Code Custom Subagents: Build Specialised AI Assistants for Any Task

Learn how to create custom subagents in Claude Code with specialised prompts and tool access. Build SEO auditors, content writers, and more. Complete guide for developers and agencies.

9 February 2026 · 12 min read

A general-purpose AI assistant is useful. A specialised AI assistant that knows your framework, your coding standards, your client's brand guidelines, and the exact format you need for deliverables is transformative.

Claude Code's custom subagents let you create exactly that. You define a system prompt, restrict which tools the subagent can access, and deploy it as a focused worker that handles specific tasks with precision that a general assistant cannot match.

This is not just a developer tool. For agencies and teams managing multiple projects, subagents represent a way to encode institutional knowledge into reusable, specialised AI workers that any team member can invoke.

What Are Subagents?

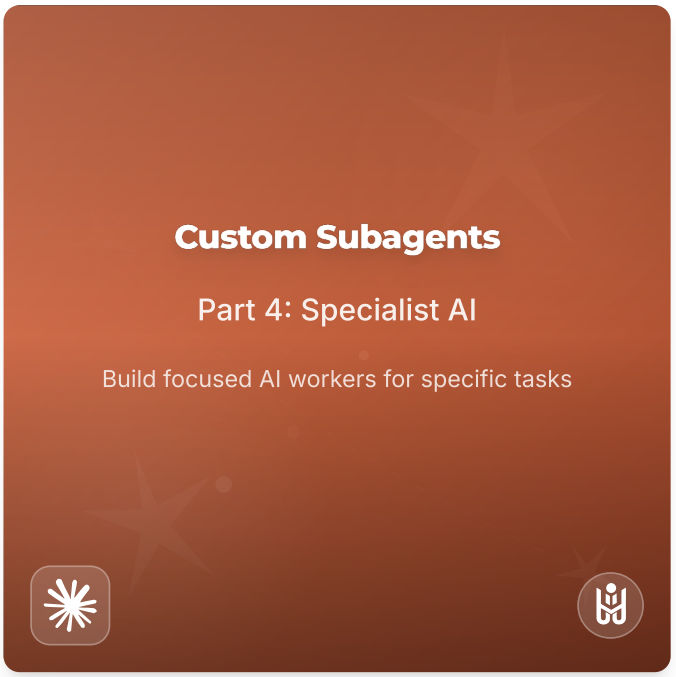
Subagents are specialised AI assistants that operate within Claude Code. Each subagent has its own custom system prompt and can be restricted to specific tools, making it purpose-built for a particular type of task.

Key characteristics:

- **Custom system prompts** - You define exactly what the subagent knows, how it should behave, and what output format it should produce
- **Tool restrictions** - Limit which tools the subagent can access (file reading, web search, terminal commands, etc.)

- **Scoped execution** - Subagents complete their task and report results back to the main Claude Code session
- **Reusable definitions** - Create once, use across every session and every team member

When you invoke a subagent, Claude Code spawns a focused worker that executes its task independently. The subagent reads the files it needs, performs its analysis or generation, and returns the result to your main session. You get specialised output without having to provide lengthy instructions every time.



Subagents vs Teammates: When to Use Each

Claude Code offers two models for delegating work: subagents and teammates (covered in our Agent Teams guide). Understanding the difference is important.

Subagents

- Report results **back to the main agent only**
- No peer-to-peer communication with other subagents
- Best for **focused tasks within a session**: "analyse this file", "generate this component", "audit this page"
- Lightweight and fast to invoke
- Think of them as specialised tools you call on demand

Teammates

- Can **communicate with each other** and coordinate work
- Operate as parallel, independent workers
- Best for **complex workstreams** that require collaboration: "rebuild this entire module while another

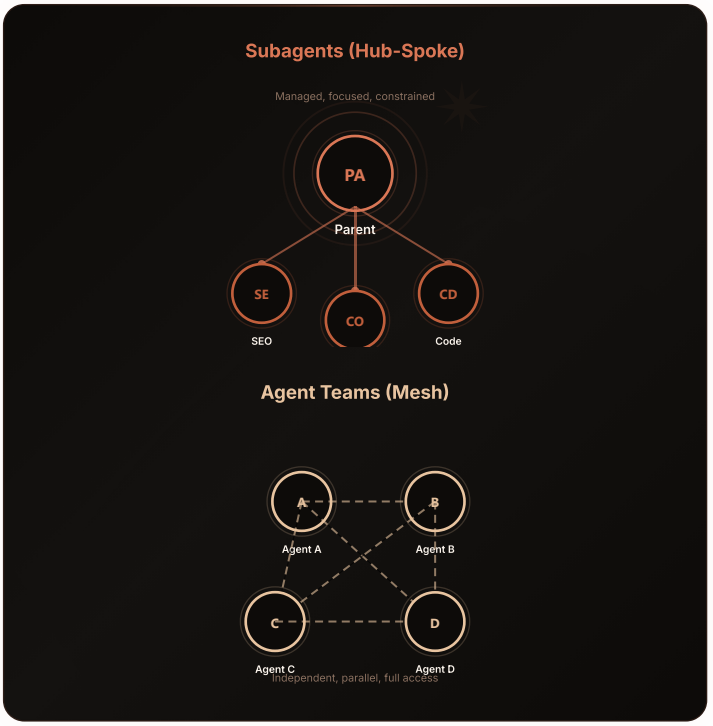
teammate updates the tests"

- Heavier orchestration overhead
- Think of them as team members working alongside you

The simple rule: If a task can be completed by a single focused worker with no need to coordinate with others, use a subagent. If the work requires multiple workers to communicate and divide responsibilities, use teammates.

In practice, subagents handle 80% of delegation needs. You only escalate to the full teammate model when the work genuinely requires parallel collaboration.

Subagents vs Agent Teams: Two Topologies



How to Create Custom Subagents

Creating a subagent is straightforward. You define it as a Markdown file in your project's `.claude/agents/` directory.

Directory Structure

```
your-project/
  .claude/
    agents/
      seo-auditor.md
      content-writer.md
      code-reviewer.md
      test-generator.md
```

Anatomy of a Subagent Definition

Each `.md` file in the agents directory becomes an available subagent. The file contains the system prompt that defines the subagent's behaviour, knowledge, and constraints.

Here is a simplified example:

```
# SEO Auditor

You are a specialised SEO audit agent. Your role is to analyse web
and content files for SEO issues and opportunities.
```

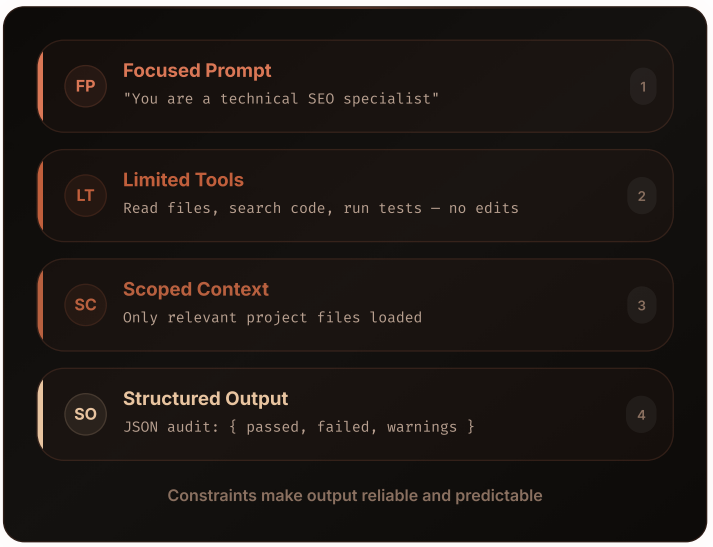
What Are Subagents?

Subagents are specialised AI workers with focused prompts and limited tool access. Unlike general-purpose agents, they stay on task and produce consistent, predictable output.

Your Expertise

- Technical SEO (meta tags, heading structure, schema markup)
- Content optimisation (keyword density, readability, internal linking)
- Core Web Vitals implications
- Australian English spelling conventions

Anatomy of a Subagent



Output Format

Always produce your audit as a structured JSON report with:

- score (0-100)
- issues (array of findings with severity)
- recommendations (prioritised list)

- quick_wins (changes that take under 5 minutes)

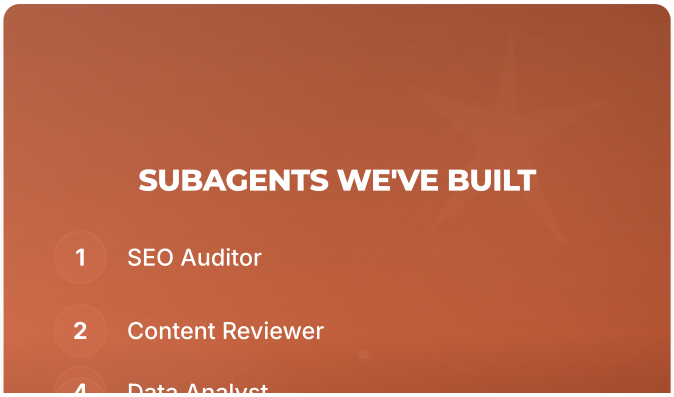
Constraints

- Never modify files directly. Report findings only.
- Flag but do not fix accessibility issues.
- Use Australian English in all commentary.

```
### Invoking a Subagent

Once defined, you can invoke subagents during any Claude Cod
```

Run the seo-auditor agent on the homepage template

```
Claude Code reads the subagent definition, spawns a focused

You can also restrict which tools a subagent can access. An

---
```



The Community Ecosystem

One of the most compelling aspects of Claude Code's subagent system is the growing community ecosystem. Over 100 pre-built subagent definitions are available from the community, covering common development tasks.

Popular community subagents include:

- **Code reviewers** that check for security vulnerabilities, performance issues, and style violations
- **Documentation generators** that produce consistent API docs from source code
- **Test writers** that generate unit and integration tests following specific frameworks
- **Migration assistants** that help upgrade dependencies or refactor for new API versions
- **Accessibility auditors** that check HTML output against WCAG guidelines

You can use community subagents as-is, fork and customise them for your needs, or use them as templates for building your own. The `.claude/agents/` directory is just Markdown files, so sharing and collaborating on subagent definitions is as simple as sharing text files.

Subagent Capability Matrix

Capability	SEO Auditor	Content Reviewer	Code Reviewer	Data Analyst
Read Files	Y	Y	Y	Y
Edit Files	-	-	-	-
Run Tests	-	-	Y	-
Search Code	Y	Y	Y	Y
Web Fetch	Y	-	-	Y

Limited tools = focused output. No subagent can edit files — they analyse and report.

Agency Use Cases: Where Subagents Shine

For agencies managing multiple clients and projects, subagents solve a persistent problem: how do you encode

the knowledge, standards, and preferences of each client into a reusable system?

Client-Specific Content Writers

Create a subagent for each client that knows their:

- Brand voice and tone guidelines
- Preferred terminology and phrases to avoid
- Target audience demographics
- Content structure preferences
- SEO keyword targets

When any team member needs to write content for that client, they invoke the client's content subagent. The output is on-brand from the first draft, regardless of who on your team is running it.

```
# Acme Corp Content Writer

You write marketing content for Acme Corp, a B2B SaaS company
targeting mid-market CFOs in Australia.
```

Brand Voice

- Professional but not corporate
- Data-driven, always cite specifics
- Australian English spelling
- Never use: "leverage", "synergy", "disrupt"
- Preferred: "improve", "streamline", "accelerate"

Subagent Prompt Structure

terminal

```
# SEO Auditor Subagent

## Your Expertise
You are a technical SEO specialist.

## Output Format
Return a JSON audit with: passed, failed,
warnings.

## Constraints
- Only check technical factors
- Do not modify any files
- Flag issues, don't fix them
```

The prompt defines expertise, output format, and constraints. These boundaries make the subagent reliable — it always produces the same structured output.

Content Structure

- Lead with the business problem
- Follow with quantified impact
- Present the solution third
- Close with a specific next step

PRO TIP

The Subagent Ecosystem

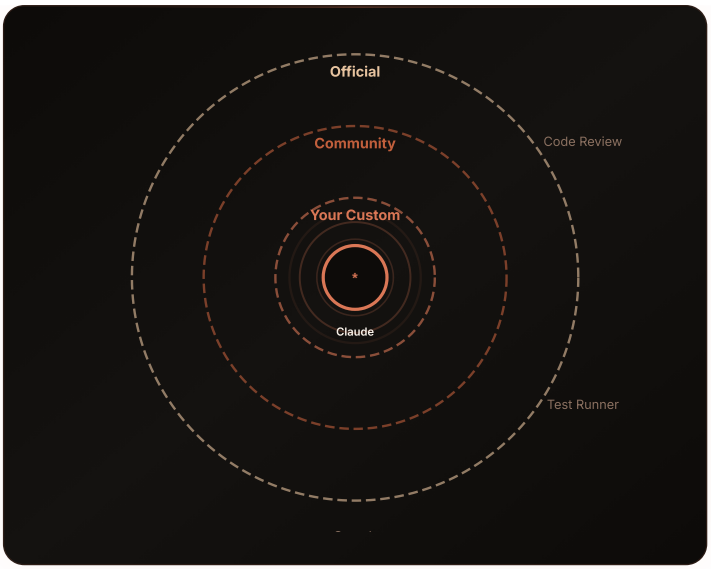
Anthropic is building a community marketplace for subagent prompts. Soon you'll be able to share and discover specialist agents — an SEO auditor, a content grader, a security scanner — all plug-and-play.

SEO Requirements

- Include primary keyword in H1 and first paragraph
- Use secondary keywords in H2 headings
- Internal link to /platform/ and /case-studies/ where natural

```
### Framework-Aware Auditors

Build subagents that know your agency's audit frameworks int

We use this pattern at Jordan James Media. Our audit subagen

### Deployment Validators

Create subagents that check code before it goes to productio
- Verify that all environment variables are set
- Check that database migrations are in order
- Confirm that tests pass
- Validate that no debug code remains
- Ensure client-specific configurations are correct

This catches errors that general code review misses because

---
```

The Subagent Ecosystem



Building Effective Subagents: Best Practices

1. Be Specific About Output Format

The most effective subagents produce consistent, structured output. Define the exact format you expect: JSON for machine-readable results, Markdown for documentation, specific heading structures for reports. Vague instructions produce vague output.

2. Include Examples

Show the subagent what good output looks like. Include a sample in the system prompt. One concrete example communicates more than a paragraph of abstract instructions.

3. Define Boundaries Clearly

State what the subagent should and should not do. "Analyse and report, but never modify files" prevents unintended side effects. "Only examine TypeScript files in the /src directory" prevents scope creep.

4. Layer Knowledge Progressively

Start with a general-purpose subagent and refine it over time. After each use, note what worked and what the subagent got wrong. Update the system prompt. Subagent definitions are living documents that improve with use.

5. Keep System Prompts Focused

A subagent that tries to do everything does nothing well. Better to have five focused subagents than one overloaded one. An SEO auditor should audit. A content writer should write. A code reviewer should review. Specialisation is the entire point.

6. Version Control Your Agents

Because subagent definitions live in `` `.claude/agents/` `` as Markdown files, they are automatically version-controlled

with your project. This means you can track how your subagents evolve, roll back changes that reduce quality, and share improvements across the team.

> *Custom subagents make Claude Code feel less like a tool and more like an extensible platform. You're not limited to what it can do out of the box — you build what you need.*
>
> — Jordan James Media

Key Takeaways

1. **Subagents are specialised AI workers** with custom prompts and restricted tool access
2. **Different from teammates** - subagents report to the main agent; teammates collaborate with each other
3. **Defined as Markdown files** in `.claude/agents/`, making them easy to create, share, and version control
4. **100+ community subagents** available as starting points or ready-to-use tools
5. **Agency game-changer** - encode client knowledge, brand guidelines, and audit frameworks into reusable agents
6. **Best when focused** - one task per subagent produces better results than multi-purpose definitions
7. **Combine with other features** - Skills, Checkpoints, Plan Mode, and MCP multiply subagent effectiveness
8. **Start small** - build one subagent for a repeated task and expand from there

---

Build Your AI Workforce

Custom subagents represent a shift from using AI as a general assistant to deploying it as a team of specialists. Each subagent carries institutional knowledge, follows established standards, and produces consistent results regardless of who invokes it.

At Jordan James Media, our subagent library has become one of our most valuable assets. It encodes years of process refinement into reusable AI workers that make every team member more effective.

Ready to build specialised AI workflows for your business? Talk to our team at Jordan James Media about how custom subagents can accelerate your development and marketing operations.

Related Reading:

- Claude Code's Biggest Update: 7 Features That Changed How We Build
- Claude Code Agent Teams: Parallel AI Development
- Claude Code Skills System: Reusable AI Workflows

Key Takeaway

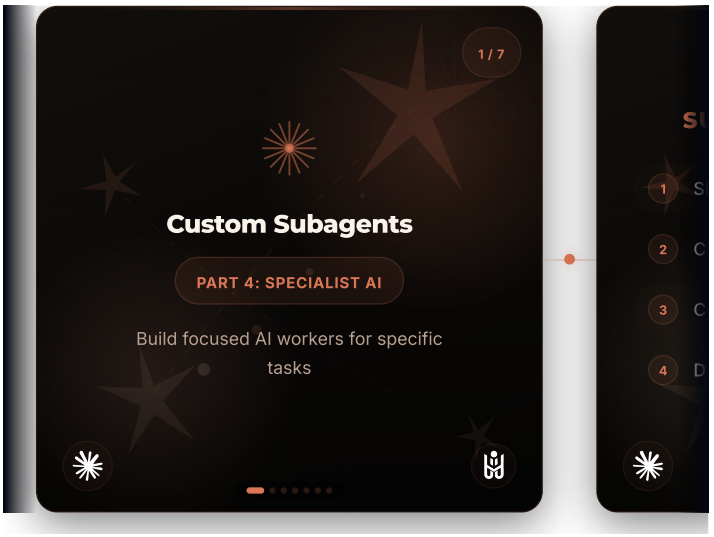
1. Subagents are specialist AI workers with focused prompts and constrained tools
2. Different from Agent Teams: subagents are managed by a parent, teams are independent
3. Structured prompts (expertise + output format + constraints) produce reliable results
4. Community ecosystem emerging for shareable subagent definitions
5. Start by identifying your most repeated analysis task — that's your first subagent

Social Media Carousel

7 cards — Scroll to browse, click any card to download

PDF for LinkedIn

ZIP Images



Share This Article

Spread the knowledge

Twitter LinkedIn Copy

Build With Claude Code

We use Claude Code to ship faster, smarter, and at scale. Let us build your next project.

Get Your Free AI Strategy →

App Development

Claude Code

8 parts

View All

Quick Links

→ Series Home
→ App Development
→ Get In Touch

Claude Code Stats

Model	Opus 4.6
Features	7
Series Parts	8

Agency Tested
Real projects, real results

Ready to outperform your competition?

Get your free AI strategy consultation today.

Start Your Free Consultation →

Jordan James Media
GROWTH STUDIO

Australia's AI-first digital marketing agency. We build intelligent systems that drive exponential growth.

info@jordanjamesmedia.com

0475 897 737

Sydney, Australia

SERVICES

- SEO Services
- Google Ads
- Website Design
- Local SEO
- Content Marketing
- App Development

COMPANY

- About Us