CLAUDE COWORK PLUGINS

The Complete Builder's Guide

From architecture to deployment — a practitioner's playbook for building intelligent AI workflow plugins for Experience Design and beyond.

Mayur Chaudhary

Design Leader • Founder, RethinkingUX • Author, UX Mastery

February 2026

Author: Mayur Chaudhary • LinkedIn / Instagram @mayurchaudhary

Contents

CHAPTER 01

What Are Cowork Plugins?

On January 30, 2026, Anthropic released 11 open-source plugins for Claude Cowork. Within 48 hours, software stocks lost $285 billion in market value. The event was dubbed the "SaaSpocalypse" — not because the plugins were revolutionary technology, but because they revealed an uncomfortable truth: a few kilobytes of structured expertise can replace workflows that companies have been charging millions for.

A Cowork plugin is not software in the traditional sense. It is codified expertise — markdown files and JSON configurations that teach an AI agent how to perform your specific work, using your methodology, at a level of consistency no team can match.

Why This Matters for Design Leaders

Every design practice has institutional knowledge trapped in people's heads. How you assess a client's digital maturity. How you structure discovery. How you position against competitors. How you estimate effort. This knowledge works when the right people are in the room. It fails when they are not. It degrades every time someone leaves. It does not scale.

> **The core insight:** Building a Cowork plugin is not a technical exercise — it is a leadership exercise. The hardest part is not the tool. It is deciding what "good" looks like and writing it down with enough precision for an AI to follow.

What a Plugin Contains

- **Skills** — Markdown files with domain expertise that auto-activate when Claude recognises relevant context
- **Commands** — Slash commands (`/research`, `/pitch-deck`) that trigger structured multi-step workflows
- **Connectors** — MCP configurations that wire Claude to external tools and data sources
- **References** — Knowledge base files (playbooks, templates, methodology docs) that skills and commands draw upon

No compiled code. No infrastructure. No deployment pipeline. The entire system runs on structured thinking expressed as text files.

CHAPTER 02

Plugin Anatomy

Every Cowork plugin follows the same folder structure. Understanding this anatomy is essential before you start building.

The Folder Structure

Path	Purpose
`your-plugin/`	Root folder (plugin name)
`.claude-plugin/plugin.json`	Manifest — name, version, description, tags
`.mcp.json`	MCP connectors — external tool wiring
`commands/`	Slash command definitions (one .md per command)
`skills/`	Domain expertise directories (each contains SKILL.md)
`references/`	Knowledge base — playbooks, templates, guides
`.claude/settings.local.json`	Local personalisation (names, rates, prefs)

The Plugin Manifest

The manifest file (`plugin.json`) is the identity card of your plugin. It tells Claude what the plugin does and when to use it.

```
{   "name": "your-plugin-name",   "version": "1.0.0",   "description": "Clear description of what     this plugin does and WHEN to use it.",   "author": {     "name": "Your Name"   },   "license": "proprietary" }
```

Critical: The description field determines when Claude activates your plugin. Write it like explaining to a smart colleague when to reach for this tool. The more precise the trigger context, the better the activation accuracy.

Manifest validation rules: `author` must be an object (`{"name": "..."}`), not a plain string. Do not include `tags` — it is not a recognised field and will cause the plugin loader to reject the manifest on startup.

How Many Files Do You Need?

Plugin Size	Files	Best For
Minimal	3–5	Single workflow (e.g., a research process)
Focused	10–15	One practice area with multiple workflows
Comprehensive	25–40	Full practice intelligence system
Enterprise	40+	Multi-vertical, multi-team shared knowledge

CHAPTER 03

Skills: Teaching AI Your Expertise

Skills are the brain of your plugin. Each skill is a directory inside `skills/` containing a single `SKILL.md` file with domain expertise. Claude loads them automatically when it recognises relevant context. Think of them as: "if the conversation is about X, here is everything I know about X."

Anatomy of a Skill Directory

Every skill lives in its own directory inside `skills/`, with a single `SKILL.md` file containing two parts: a YAML frontmatter header (name + description for triggering) and a markdown body (the actual expertise). Keep each under 500 lines — if longer, have the skill reference files in the `references/` folder.

```
skills/ client-intelligence/ # directory SKILL.md # expertise file digital-maturity/ SKILL.md --- (inside SKILL.md) --- --- name: client-intelligence description: "Triggers when researching a company,  preparing for a client meeting, or starting  any engagement with a named organisation." --- # Client Intelligence Framework ## Research Dimensions ### 1. Company Overview & Strategy Search for earnings calls, annual reports...
```

Skill Design Principles

- **Be specific, not generic.** Don't write "research the company." Write "Search for recent earnings calls, annual reports, CEO statements. Extract strategic priorities, M&A activity, leadership changes."
- **Include the So What.** Every finding should connect to action. Don't just describe a framework — explain why each dimension matters and how to use the output.
- **Use structured headings.** Claude navigates your skill like a reference manual. Clear H2/H3 structure helps it find the right section fast.
- **Encode judgment, not just process.** "Score maturity 1–5" is process. "A score of 3 means competent mobile web but no native app strategy" is judgment.
- **Write for precision, not brevity.** Vague skills produce inconsistent results. Detailed skills produce reliable results. Err on specificity.

Skill Categories for Experience Design

Skill Type	Example	Triggers When
Research	Client Intelligence	Researching any company or client
Assessment	Digital Maturity Model	Auditing a client's digital experience

Skill Type	Example	Triggers When
Positioning	Competitive Landscape	Comparing against competitors
Estimation	Deal Sizing Framework	Estimating effort, team, investment
Methodology	Delivery Approach	Describing how you work
Qualification	Go/No-Go Assessment	Evaluating whether to pursue an opportunity
Matching	Case Study Mapper	Finding relevant proof points
Generation	Brand Intelligence	Extracting visual identity for outputs

Pro tip: The skill description in YAML frontmatter is the single most important line in your plugin. It determines triggering accuracy. Test by asking: "If I said these words in conversation, would I want this skill to activate?" Refine until the answer is always yes.

Warning: frontmatter is mandatory. Without the YAML block containing `name` and `description`, your skills will register with empty descriptions and never auto-trigger. The plugin system reads the trigger description exclusively from frontmatter — not from your markdown headings or body text.

CHAPTER 04

Slash Commands: Orchestrating Workflows

While skills activate automatically, slash commands are explicit — the user types a command and Claude executes a structured multi-step workflow. Commands are your power tools.

Command File Structure

Each command is a single markdown file in `commands/`. The filename becomes the command name.

```
# /xd:client-research Full client intelligence workflow. ## Usage /xd:client-research <company name> ## What This Does 1. Identifies industry, loads playbook 2. Runs client intelligence skill 3. Conducts digital presence audit 4. Extracts brand identity 5. Maps competitive landscape 6. Identifies strategic entry points ## Output Structured intelligence dossier (markdown)
```

Command Design Patterns

Research Commands — Gather and synthesise information from multiple sources into a structured deliverable.

Analysis Commands — Apply structured frameworks to produce scored assessments and benchmarks.

Generation Commands — Produce finished outputs: presentations, documents, emails, one-pagers.

Qualification Commands — Help with go/no-go decisions using weighted scoring models.

Example Command Set

Command	Type	Output
`/xd:research <company>`	Research	7-dimension intelligence dossier
`/xd:gap-analysis`	Analysis	10-dimension maturity scorecard + roadmap
`/xd:position`	Positioning	Deal-specific competitive battlecard
`/xd:estimate`	Estimation	Team plan, timeline, investment range
`/xd:pitch-deck`	Generation	25-slide branded deck with speaker notes
`/xd:go-nogo`	Qualification	7-dimension scorecard with recommendation
`/xd:followup`	Generation	Email, one-pager, FAQ package

Command	Type	Output
`/xd:case-match`	Matching	Tiered relevant proof points

> **Design principle:** A command should do one complete workflow and produce one clear output. If you are tempted to make a command that does "everything," break it into focused commands instead. Composability beats monoliths.

CHAPTER 05

MCP Connectors: Wiring External Tools

The Model Context Protocol (MCP) lets your plugin reach beyond local files to connect with external tools. Connectors are defined in `.mcp.json` at the plugin root.

Essential Connectors

Connector	Type	Use Case
Web Search	builtin	Client research, competitor analysis, news, brand extraction
Google Drive	http	Past proposals, templates, brand guidelines, case studies
Slack	http	Deal context, team availability, relationship signals
Notion / Confluence	http	Internal knowledge base, methodology docs, team profiles
Salesforce / HubSpot	http	Pipeline data, account history, existing relationships
Figma	http	Design system assets, component libraries, project files

Configuration Format

```
{   "mcpServers": {     "web-search": {       "type": "builtin",       "description": "Web search"     },     "slack": {       "type": "http",       "url": "https://mcp.slack.com/sse",       "description": "Deal context"     }   } }
```

Critical: Use `"type": "builtin"` for web search and `"type": "http"` with a real URL for all other connectors. Never use `"type": "mcp"` with placeholder notes — the plugin loader will fail trying to initialise servers without valid endpoints. Only include connectors you can actually configure.

Phased Connector Strategy

- **Phase 1:** Web search only (built-in, works immediately with zero config)
- **Phase 2:** Add Google Drive or Notion for internal knowledge access
- **Phase 3:** Add Slack, CRM if your workflows need real-time team or client data

Start minimal. Add connectors only when you can clearly articulate what external data source will improve output quality for a specific skill or command.

CHAPTER 06

Reference Files: Your Knowledge Base

Reference files are the knowledge library that skills and commands draw upon. They contain detailed information that would make skill files too long: industry playbooks, methodology details, templates, and guides.

Organising References

```
references/ ■■ industries/ # One file per vertical ■ ■■ financial-services.md ■ ■■
healthcare.md ■ ■■ retail.md ■■ methodology/ # Delivery approach ■ ■■
delivery-approach.md ■■ templates/ # Output templates ■■ ppt-generation-guide.md ■■
executive-summary.md
```

What Goes in an Industry Playbook

- **Industry overview** — Current landscape and forces shaping the sector
- **Key challenges** — 5–7 specific pain points design can address
- **XD opportunities** — Concrete project types relevant to this vertical
- **Metrics that resonate** — KPIs that matter to buyers in this industry
- **Positioning guidance** — How to lead conversations in this sector
- **Key players** — Major companies and competitors for context

Presentation Generation Guide

If your plugin generates presentations, include a detailed PPT guide covering:

- Brand colours (exact hex values), typography, visual style characterisation
- Dual-branding strategy: your brand as primary (70%), client brand as accent (30%)
- Slide sequence template with section purposes and slide counts
- Speaker notes requirements: key message, talking points, transition cues
- Quality checklist: minimum font sizes, contrast, margins, file size limits

> **Scaling tip:** Start with 2–3 industries you work in most. An empty playbook is worse than no playbook — it produces generic output.

CHAPTER 07

Building a Plugin: Step by Step

This chapter walks through building a complete plugin from scratch. Substitute your own domain expertise for the XD-specific examples.

Step 1: Map Your Expertise

- What are the 5–7 core workflows in your practice?
- For each workflow, what separates a senior practitioner's output from a junior's?
- What institutional knowledge exists only in people's heads?
- Where does quality vary depending on who is doing the work?
- What industry-specific knowledge do you draw upon repeatedly?

> **This is the hardest step.** Most leaders discover their "methodology" is actually strong intuitions, not a structured system. The plugin-building process forces you to convert intuition into structure — and that is where the real value lies.

Step 2: Define Your Skills (6–10 files)

Skill	What It Encodes	Lines
Client Intelligence	How you research a prospective client across multiple dimensions	80–150
Digital Maturity Assessment	Your scoring model for evaluating current state	100–180
Competitive Landscape	How you position against specific competitors	120–200
Offerings Catalog	Complete capability set, delivery models, differentiators	100–160
Deal Sizing	Team composition templates, rate structures, timelines	80–150
Go/No-Go Assessment	Qualification criteria for pursuing opportunities	80–120
Case Study Matcher	Logic for mapping past work to new challenges	60–90

Skill	What It Encodes	Lines
Response Framework	Narrative arc and section structure of proposals	120–200

Step 3: Design Your Commands

Map commands to the deliverables your team produces. Each command orchestrates multiple skills into one end-to-end workflow.

Step 4: Build Your Knowledge Base

Populate industry playbooks (start with 2–3), document your delivery methodology, create a PPT generation guide if needed, and include output templates.

Step 5: Wire Connectors Incrementally

Start with web search. Add internal knowledge sources in Phase 2. Only add real-time connectors (Slack, CRM) when specific workflows demand them.

Step 6: Test Against Real Work

Run your plugin against past projects where you know what good looks like. Compare AI output to what your team actually produced. The gaps reveal where skills need more specificity.

CHAPTER 08

Presentation Generation Layer

One of the most powerful plugin capabilities is auto-generating branded presentations from gathered intelligence. This requires a detailed PPT generation guide.

Dual-Branding Strategy

Element	Your Brand (Primary 70%)	Client Brand (Accent 30%)
Template	Your colours, logo, layout	Client logo on title slide only
Section dividers	Your brand background	Client colour as accent stripe
Charts	—	Client brand colours (shows empathy)
Typography	Your brand fonts throughout	—
Footer	Your logo + confidentiality	Client name in title

Standard Slide Sequence (20–25 slides)

#	Section	Slides	Purpose
1	Title	1	First impression, dual-branded
2	Agenda	1	Sets expectations
3	Understanding Their World	2–3	Deep client and industry knowledge
4	Current State Assessment	2–3	Maturity scorecard, productive urgency
5	Vision & North Star	2–3	Future state, experience principles
6	Gap Analysis	2–3	Priority matrix, horizon planning
7	Approach & Methodology	3–4	Execution plan, phases, governance
8	Why Us	2–3	Win themes, case studies
9	Team & Delivery	2–3	Proposed team, milestones

#	Section	Slides	Purpose
10	Investment & Timeline	1–2	Phase-wise investment
11	Getting Started	1	Clear next step

> **Speaker notes are non-negotiable.** Every slide needs: key message (one sentence), 3–5 talking points, transition cue to next slide, and likely objections. The deck without notes is only half the deliverable.

CHAPTER 09

Testing, Iteration & Deployment

The Build Cycle

Phase	Focus	Duration
1. Draft	Write all skill files and commands from expertise map	1–2 days
2. Test	Run real scenarios, compare to known-good work	1–2 days
3. Refine	Tighten descriptions, add dimensions, fix triggering	1–2 days
4. Extend	Add playbooks, references, presentation layer	2–4 days
5. Deploy	Package, document, distribute to team	1 day

Common Pitfalls

- **Missing YAML frontmatter.** Without the --- block with name and description at the top of each SKILL.md, skills register with empty descriptions and never auto-trigger. This is the most common plugin-breaking mistake.
- **Invalid MCP server types.** Using "type": "mcp" with placeholder notes instead of "type": "http" with real endpoint URLs causes the plugin loader to fail on initialisation.
- **Manifest validation errors.** The `plugin.json` author field must be an object (`{"name": "..."}`) not a string, and `tags` is not a recognised field. Either mistake crashes the plugin loader on startup.
- **Skills too vague.** "Research the company" = generic output. "Search for earnings calls, job postings, Glassdoor reviews" = insight.
- **Descriptions too broad.** If a skill triggers on everything, it is useful for nothing. Narrow the trigger context.
- **Missing the So What.** Skills that describe frameworks without explaining how to use output produce analysis without recommendations.
- **Inconsistent structures.** If your maturity model uses 1–5 in one skill and High/Medium/Low in another, output will be confused.
- **No test against real work.** The only way to validate is to test against projects where you know what good looks like.

Packaging for Distribution

- Zip the entire plugin folder
- Include `README.md` with setup, commands, and customisation guide
- Include `settings.local.json` template with placeholder values
- Document which MCP connectors need configuration

CHAPTER 10

What to Build Next

The plugin architecture applies far beyond experience design. Here are proven patterns for other domains:

Domain	Plugin Concept	Core Skills
Thought Leadership	Content engine	Trend research, voice calibration, post architecture
Product Management	PRD builder	Market research, user stories, prioritisation, roadmap
Design Operations	Practice manager	Utilisation, hiring, capacity, skill gap analysis
Sales Engineering	Solution architect	Discovery, architecture patterns, estimation, demos
Content Strategy	Content intelligence	Audience research, auditing, editorial calendar
UX Research	Research ops	Study planning, recruitment, analysis, synthesis
Engineering	Technical proposals	Architecture review, tech selection, risk, migration
Community	Community manager	Events, speakers, engagement, content repurposing

The Bigger Picture

The most valuable plugins do not just automate tasks. They capture institutional expertise that would otherwise be trapped in individuals' heads, inconsistent across teams, and lost when people move on.

The real question is not "what can I automate?" It is: "what expertise in my practice is too valuable to exist only as tribal knowledge?"

> **The builder's moat:** Your craft skills can be learned in months. Your ability to structure a discovery process, define quality criteria, and encode institutional expertise into scalable systems? That compounds over a career. And now, for the first time, you can package it into a system that scales beyond you.

Mayur Chaudhary
Design Leader • Founder, RethinkingUX • Author, UX Mastery
LinkedIn / Instagram: @mayurchaudhary

Built with Claude Cowork • February 2026