EVOKNOW.AI · AI Super Power Skills for Claude

The Claude Guide to Building AI Agent Teams

Getting started with Metaskill -- one skill to create them all

Metaskill turns a single slash command into a complete, research-driven AI agent team: a tech lead, a handful of specialists, a code reviewer, ready-to-run skills, coding standards, and a pre-wired set of MCP servers -- generated in minutes instead of assembled by hand over hours.

This guide walks you through what Metaskill is, how to install it, and how to generate your first team, single agent, or custom skill. No prior multi-agent experience required -- if you can run Claude Code, you can run Metaskill.

Repository: github.com/xvirobotics/metaskill
Author: xviRobotics · Published by EVOKNOW.AI
Guide version 1.0

1. What is Metaskill?

Metaskill is a skill for Claude Code whose entire job is to build other AI infrastructure for you. Its tagline says it plainly: "one skill to create them all." From a single /metaskill command it can generate a full multi-agent team, a single expert agent, or a standalone workflow skill -- whichever your request implies.

Standing up a multi-agent system by hand usually means hours of work: designing roles, writing system prompts, choosing which MCP servers to wire in, and defining how the agents hand work to one another. Metaskill removes that setup tax. It researches current best practices for your stack, then writes all of the configuration for you.

2. What you need

- Claude Code CLI, installed and authenticated. Metaskill runs inside Claude Code.
- A terminal and about five minutes for installation and your first generation.
- Optional: API keys or tokens for any MCP servers your team ends up using. Metaskill can collect these during setup, or you can add them later.

3. Installing Metaskill

Automated (recommended)

Run the one-line installer. It places the skill and its flow files under your ~/.claude/skills/metaskill directory:

```
curl -fsSL https://raw.githubusercontent.com/xvirobotics/metaskill/main/install.sh | bash
```

Manual

Prefer to see exactly what lands where? Create the folders and pull the four files yourself:

```
mkdir -p ~/.claude/skills/metaskill/flows

curl -fsSL .../skill/SKILL.md -o ~/.claude/skills/metaskill/SKILL.md
curl -fsSL .../skill/flows/team.md -o ~/.claude/skills/metaskill/flows/team.md
curl -fsSL .../skill/flows/agent.md -o ~/.claude/skills/metaskill/flows/agent.md
curl -fsSL .../skill/flows/skill.md -o ~/.claude/skills/metaskill/flows/skill.md
```

(Replace "..." with https://raw.githubusercontent.com/xvirobotics/metaskill/main -- shortened here to fit the page.)

4. Generating your first team

Open Claude Code in an empty project folder and describe what you want to build. Metaskill detects your intent and picks the right flow automatically:

```
/metaskill fullstack web app with React and PostgreSQL
/metaskill a code reviewer agent
/metaskill a slash command that summarizes my git diff
```

The first request builds a whole team. The second creates a single agent. The third produces one workflow skill. You do not choose the mode -- your wording does.

The three modes

Team (default): a coordinated group -- tech lead, specialists, and a code reviewer -- plus skills, rules, and MCP configuration.

Single agent: one focused expert persona with its own system prompt, created when your request names a specific role.

Single skill: one automation triggered by a slash command, created when you ask for a command-style workflow.

5. How generation works

For a team, Metaskill runs a four-phase process so the result reflects current practice rather than a frozen template:

Phase 1 -- Research: web searches for team structures, existing Claude Code setups, relevant MCP servers, and best practices for your stack.

Phase 2 -- Build: writes CLAUDE.md and .mcp.json, generates four to six agents, two to four automation skills, and language-specific coding standards.

Phase 3 -- Credentials: prompts for any API keys the chosen MCP servers need, with a skip option so you can configure them later.

Phase 4 -- Verify: lists every file created, validates the .mcp.json syntax, and prints a summary of what was generated.

6. What gets created

Metaskill lays down a self-contained project structure that Claude Code can use immediately:

```
<project-name>/
CLAUDE.md routing table + orchestration protocol
.mcp.json MCP server configuration
.claude/
agents/
tech-lead.md
<specialist-1..3>.md
code-reviewer.md
skills/
<skill-1>/SKILL.md
<skill-2>/SKILL.md
rules/
<language>-standards.md
```

7. Bundled MCP servers

Teams are wired with a set of verified MCP servers, chosen to match your stack. Common ones include:

- context7 -- up-to-date library documentation.
- playwright -- browser automation and testing.
- filesystem -- safe file operations.
- postgres -- database queries.
- sequential-thinking -- structured reasoning.
- memory -- knowledge persistence across sessions.
- github -- repository and pull-request access.

8. Example projects

The repository ships ready-made teams you can copy and run to see the pattern in action:

- fullstack-web -- React + Node.js + PostgreSQL, with frontend, backend, and devops engineers.
- ios-app -- SwiftUI + Swift, with an iOS engineer, UI designer, and test engineer.
- data-science -- Python + PyTorch, with data and ML engineers plus an analyst.

To try one:

```
cp -r examples/fullstack-web my-project && cd my-project && claude
```

9. Agent architecture & discipline

Every generated team follows the same orchestration pattern. A tech lead (on the stronger Opus model) coordinates the work and delegates to specialists (on Sonnet), and a dedicated code reviewer enforces quality gates before anything is considered done.

All agents ship with built-in "Workflow Discipline": they plan before acting, re-plan when a step fails, and verify completion rather than assuming success. That is what keeps a multi-agent run from drifting off course.

10. Tips & troubleshooting

- Be specific about your stack. "React and PostgreSQL" yields a sharper team than "a web app" -- the research phase has more to work with.
- Skip credentials on the first pass if you just want to explore the generated files; you can fill in .mcp.json keys afterwards.
- If /metaskill is not recognized, confirm the skill lives at ~/.claude/skills/metaskill/SKILL.md and restart Claude Code.
- Start from an example project when you are new -- reading a working team is the fastest way to understand the routing table in CLAUDE.md.

Resources

- Repository & docs: github.com/xvirobotics/metaskill

More skills, reviews, and guides: evoknow.ai/ai-skills

Published by EVOKNOW.AI · Guide version 1.0 · Metaskill is an independent open-source project by xviRobotics.