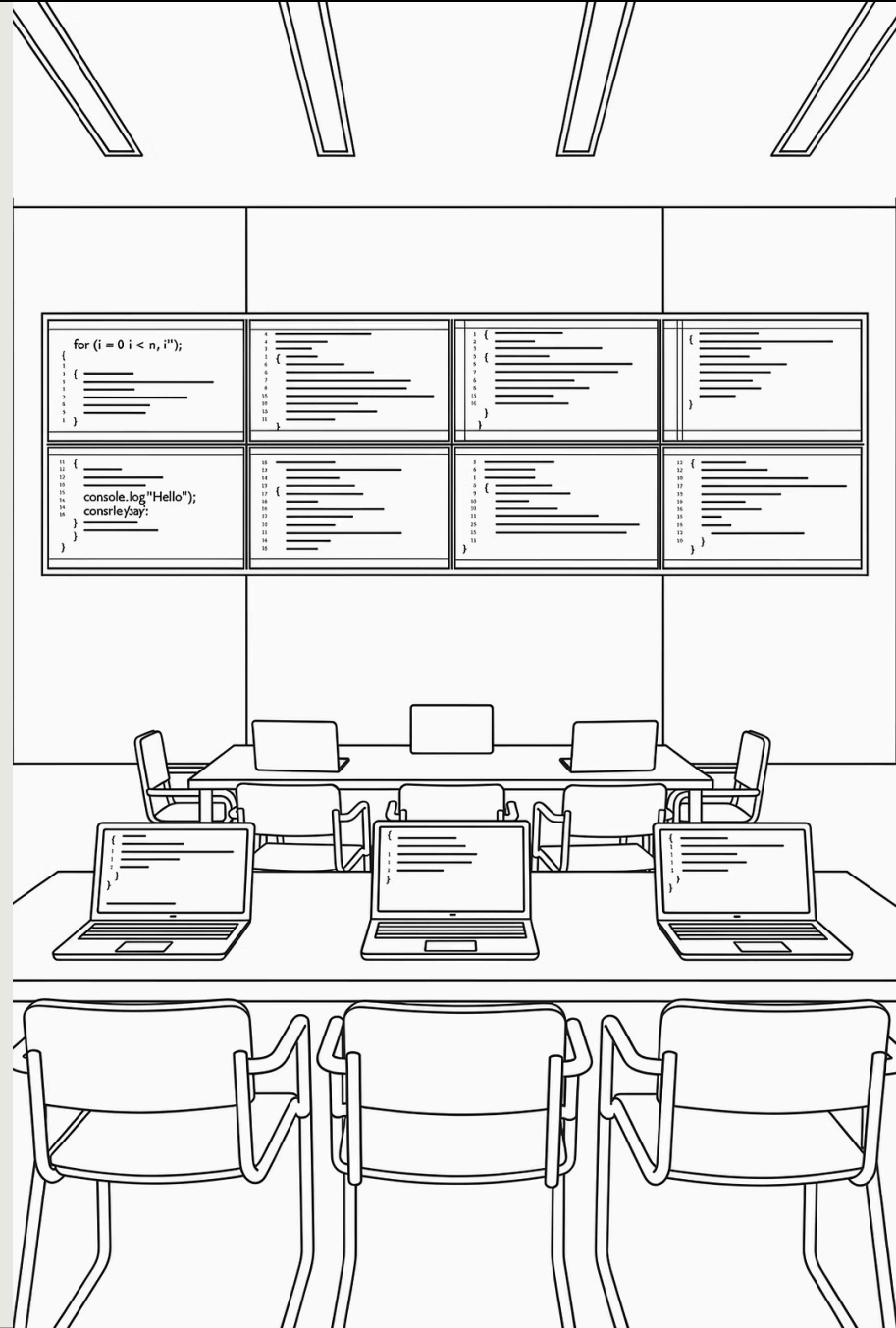
CLAUDE CODE CO-BUILDER: LESSON AND WORKSHOP

AI progression: chat > co-builder > autonomous agents

CLAUDE CODE CO-BUILDER

LESSON AND WORKSHOP

A 90-minute workshop for entrepreneurs and builders. Go beyond chat AI to set up Claude Code as a co-builder with persistent memory, task management, MCP, and session protocols. Includes a live demo of a 38-session setup and a hands-on workshop to configure your CLAUDE.md and run your first session.



THE GAP

Tools advance quickly. Adoption, trust, and behavior change lag, creating a significant gap.

AI tools launch weekly. Team adoption is slow. Bottleneck: not tech, but understanding, trust, and integration.

It's not 'which tool is best,' but understanding each's function and your progression. Skipping steps costs teams.

The real opportunity: While others debate tools, teams grasping AI progression quietly gain advantage, workflow by workflow.

THE PROGRESSION

AI tool adoption involves three distinct stages. Many skipped Stage 2, drawn to autonomous agents. However, **Stage 2 offers immediate, untapped value**.

STAGE 1 · CHAT

ChatGPT, Claude.ai

Q&A, brainstorming, writing. Context resets each time.

STAGE 2 · CO-BUILDER

Claude Code

Interactive collaboration. AI maintains context, reads files, runs code, remembers sessions. It's building, not Q&A.

STAGE 3 · AUTONOMOUS AGENTS

SDK, OpenClaw

AI runs independently; you review results. Potent with a solid foundation, but premature without Stage 2 mastery.

SIDE-BY-SIDE COMPARISON

Each tool aligns with a different stage. Here's a comparison across key dimensions.

Dimension	Claude.ai / ChatGPT	Claude Code	Claude SDK	OpenClaw
What it is	AI chat	Interactive co-builder	Autonomous agent framework	Always-on background agent
Interface	Browser	Terminal	Custom app/service	WhatsApp, Telegram, Slack
Best at	Questions, writing, brainstorming	Collaborative building, thinking, coding	Deploying autonomous agents	Monitoring, automating, proactive nudges
Runs when	You're in browser	You're at keyboard	On-demand or always-on	24/7 (even while you sleep)
Memory	Resets per conversation	Persists across sessions	You build it	Persists on disk
Model	Claude or GPT	Claude only	Claude only	Any model
Security	Enterprise-grade	Enterprise-grade	Enterprise-grade	512 vulnerabilities (Jan audit)
Setup	Sign up	2 minutes	Code & deploy	Docker, firewall, Tailscale

REAL CAPABILITY VS. HYPE

The OpenClaw Reality Check

Despite **254K GitHub stars** and massive hype, OpenClaw has serious flaws: **512 vulnerabilities** (Jan audit), **820+ malicious plugins**, and **135K publicly exposed instances** with disabled authentication.

Claude SDK offers OpenClaw's capabilities, but you have to build it. OpenClaw gives you plumbing out of the box but with security tradeoffs.

OpenClaw's Genuine Strengths

OpenClaw's 'heartbeat' (proactive nudges) and multi-channel messaging (texting agents) are novel features.

Weak co-builder automation creates a poor foundation. Prioritize Stage 2 first.

WHAT MAKES A CO-BUILDER DIFFERENT FROM A CHATBOT?

	ChatGPT / Claude.ai	Claude Code Co-Builder
Instructions	System prompt (hidden, static)	CLAUDE.md (visible, version-controlled, evolving)
Memory	Unstructured	Structured: logs, decisions, task state, context
Projects	Uploads to conversation	Integrates with file system; reads/writes files
Knowledge	Context window only	Knowledge base with RAG on docs
Logging	No action record	Session logs: actions, decisions, next steps
Actions	Generates text	Runs code, emails, builds models, manages tasks

MCP — HOW CLAUDE CODE CONNECTS TO EVERYTHING

MCP (Model Context Protocol) is a standard for AI tools to access external services—think of them as Claude Code's plugins.

GMAIL

Read inbox, draft replies

GOOGLE DRIVE

Sync and access docs

DATABASES

Query and update directly

CUSTOM TOOLS

Anything you can build an API for

- Transforms Claude Code: From 'smart terminal' to 'operational co-builder'.
- OpenClaw offers 50+ integrations. MCP lets you build custom solutions.

WHAT IS CLAUDE CODE?

Claude Code: an **interactive AI agent** in your terminal. As a co-builder, it understands your project, remembers decisions, and works with you.

READS & WRITES FILES

Navigates project structure. Reads, edits, creates files in your dev environment.

RUNS CODE DIRECTLY

Executes scripts, tests, manages dependencies, debugs. Operates your tools.

PERSISTENT MEMORY

`CLAUDE.md` ensures context persists. Remembers preferences, project state, and past decisions across sessions.

TRUE CO-BUILDER

Not a Q&A tool, but a collaborative partner. It thinks, challenges assumptions, and builds in real time.

MY SETUP — THE CO-BUILDER STACK

After **38 sessions**, this workflow matured through compounding changes.

1	**CLAUDE.MD** Memory & behavior contract. Guides Claude on preferences, context, coding standards, and decision style.
2	**TASK SYSTEM** Tracks tasks across projects. Persists between sessions, ensuring no work is missed.
3	**SESSION LOGS** Records session events, decisions, and next steps. Provides an audit trail and speeds context recall.
4	**KNOWLEDGE BASE** RAG over docs, research, and archives. Claude references all written and collected info.
5	**SLASH COMMANDS** Custom shortcuts (e.g., `/todos`, `/reply`) reduce friction and enforce patterns.
6	**HOOKS** Auto-ingest files, session cleanup, co-pilot triggers. The glue for a self-maintaining system.

VOICE INPUT — TALK TO YOUR CO-BUILDER

Voice input transforms interaction. Speak your thoughts, dictate instructions, brainstorm verbally.

SUPERWHISPER

macOS native, local processing, fast, system-wide.

WHISPR FLOW

Similar, cloud-based transcription.

Not required, but once you start, you won't go back.

LIVE DEMO

LIVE DEMO

REAL SETUP DEMO

No canned demo. See my actual working environment, warts and all. Look for:

SESSION START

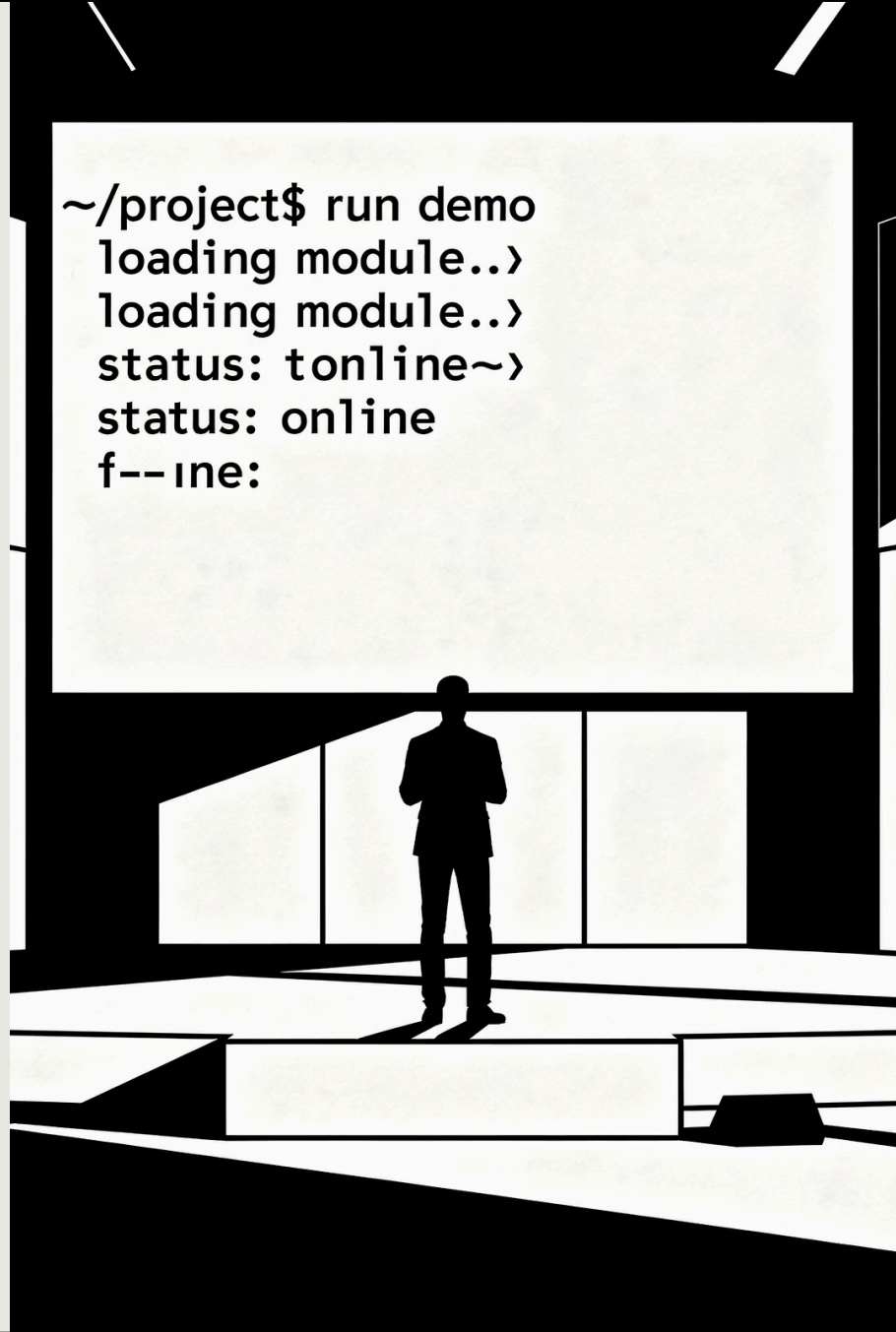
Memory loads, tasks surface, context rebuilds. See Claude get up to speed.

REAL INTERACTION

Genuine working session: decisions, file edits, code. Unscripted.

WHERE IT BREAKS

Every tool has limits. See Claude Code's struggles, setting expectations.



THE SDK — WHEN TO USE IT

The SDK creates **standalone, autonomous agents**. Powerful, but rarely the ideal starting point for most individuals.

✅ SDK USE CASES

- Event-driven services
- Autonomous webhooks/APIs
- Scheduled automation (cron)
- Enterprise phone interfaces

❌ WHEN TO AVOID SDK

- Interactive, keyboard workflows
- No established co-builder workflow
- Autonomy over collaboration

My experiment: An SDK agent for task prioritization, with a Slack interface now, and phone access next.

THE OPERATING MODEL

Effective tool usage is crucial. Speed comes from **alignment and fast learning loops**, not brute force. These core principles guide success.

CONSTRAINTS CLARIFY

Limitations are design tools, not blockers. Defined scope improves output over open prompts. Clearly state AI boundaries.

SMALL CORRECT CHANGES COMPOUND

Master fundamentals, define KPIs. Incremental improvements compound; 38 sessions show dramatic results.

DON'T CHASE THE SHINIEST TOOL

Prioritize adoption over novelty. The best tool is effective team use, not just hype. Start with fundamentals.

10 BEST PRACTICES FROM MY SETUP

INSIDE CLAUDE CODE

1. **CLAUDE.md:** Visible, version-controlled behavior contract, evolves with you.
2. **Session protocol:** Context auto-rebuilds, logs update, no data lost.
3. **Structured memory:** Always-loaded context, on-demand queries, RAG over docs.
4. **Task system:** Source of truth for status, priority, dependencies, triggers. Claude manages tasks.
5. **Session logs:** Records all actions, decisions, and next steps.
6. **Slash commands:** Repeatable workflows in one command (/todos, /reply, /ship).
7. **Privacy by design:** PII rules, pre-commit hooks, auto-cleanup of transcripts.

AROUND CLAUDE CODE

1. **Git backbone:** Version-controlled setup, evolves, revertible.
2. **Monorepo:** Codified conventions; CLAUDE.md defines placement.
3. **MCP servers + automation:** Gmail, digests, co-pilot analysis via launchd.

Inside Claude Code = how it thinks and interacts. · **Around Claude Code** = the infrastructure making it an OS.

WORKING WITH YOUR CO-BUILDER — HABITS THAT COMPOUND

CORRECTIONS STICK

- "Don't do that again" – Claude learns, avoids repeats.
- "Remember this" – preferences persist.
- "Always read before edit" – a single correction, lasting improvement.

SHAPE HOW IT THINKS

- Use "plan mode" for complex tasks.
- "Lead with the answer" – teaches communication style.
- "Push back" – empowers Claude to challenge.

STAY SAFE

- Confirm irreversible actions (e.g., trash, not rm).
- Label uncertainty – "don't present inference as fact."
- Privacy rules in CLAUDE.md protect sensitive data.

THE KEY INSIGHT

- Every correction improves future sessions.
- You're training a co-builder, not just using a tool.
- Session 1 vs 38 offers a different experience.

WORKSHOP

HANDS-ON

Time to build. Aim for a **compounding** setup, not a one-off demo.

INSTALL CLAUDE CODE

Terminal setup. 2-minute guide.

PICK SOMETHING REAL

Choose a real work task, not a toy. Something for this week.

PAIR UP

Collaborate, compare notes, share insights. Learning thrives in conversation.

LEAVE WITH A SYSTEM

Start `CLAUDE.md`, define a slash command, log your first session. Begin compounding.

WHAT'S NEXT?

ROUND THE ROOM

Your key takeaway.

What clicked? What will you try? Any surprises?

Next topic?

CLAUDE.md patterns? SDK deep dive? Security? Multi-agent orchestration? Your choice.

Close the adoption gap. One session, one change. Start compounding today.